INTERNATIONAL COMPUTERS AND TABULATORS LTD.

1900 SERIES
SYSTEM DEVELOPMENT ORGANISATION
REFERENCE MANUAL

Internal use only

Technical Information Branch
Programming Services Division

October 1965

1900 PDRM/MBC
AL 7

B. apps

I.C.T. SYSTEM DEVELOPMENT ORGANISATION

1900 SERIES REFERENCE MANUAL

1st September 1966

Instructions to recipients of the reprint up to and including Amendments 1-7

1. Amendments 1 - 6 are incorporated complete in this reprint.

2. In the case of Amendment 7, all the 'handwritten' corrections have been made, and all the pages to be replaced have been removed. It remains only for the recipient to insert the new pages, which are to be found in the accompanying Amendment List No. 7. Disregard and destroy pages 1, 2, and 3 of that list.

3. The following is a complete list of the pages to be inserted in the reprinted SDORM. (These pages replace those already removed from it).

Part	Section	Page	Part	Section	Page
Title Page					
0	5	1	2	4.5	2
0	6	1-3	2	4.5	5
1	4	3	2	4.6	2-3
1	13	1-5	2	4.6	5
1	15	5-7	3	0	7
1	15	9	4	0	Contents page
1	16	4	4	0	1-11
2	4.4	3	5	1	3
2	4.4	5-6			

Please insert the attached separator sheets and distribute the parts of the manual as follows:-

Volume 1 to contain Part 0 and Part 1

Volume 2 to contain Part 2

Volume 3 to contain Parts 3, 4, 5 and 6.

Backing slips for the three volumes are also attached.

I.C.T. SYSTEM DEVELOPMENT ORGANISATION
1900 SERIES PROGRAMMING DIVISION REFERENCE MANUAL

AMENDMENT LIST NO. 7 — 27th July, 1966.

1. Line 1 is the first line of text following the ruled line.

2. Negative line numbers count upwards from the last line of text (which is '-1') ignoring '1900 PDRM.....'

3. After a written amendment to an existing sheet, add the identity of the Amendment List in the right-hand margin, thus: AL2.

4. Vertical lines in the right-hand margin of a re-issued page indicate where amendments have been made. If there are no lines then the greater part, or the whole, of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page).

5. Update the Amendment Record.

CHANGE	PART	SECT.	PAGE	LINE	
1					Transfer parts 3 - 6 to new Vol. 3, and Part 2 to Vol. 2. Insert new spine labels. Replace title page by amended page.
2	0				Add to Contents page '5 Modification' and '6 Reserved locations'
3	0	5			Insert new section (1 page)
4	0	6			Insert new section (3 pages)
5	1	4	3		Replace by amended page
6	1	13	1-4		Replace by amended pages 1 - 5
7	1	13	5-9		Re-number as pages 6 - 10
8	1	15	1	14	Add 'except as described under instruction 156'
9	1	15	4	17	Alter '1900.....item 53' to 'Part o, Section 6'
10	1	15	5-7		Replace by amended pages
11	1	15	9		Replace by amended page

12	1	16	1	-3	Alter '8' to '10'
13	1	16	1		Add to Notes '5. N(M) must not be an accumulator. The result of doing so cannot be predicted, as it may vary for different processors, and for different runs on the same processor.'
14	1	16	3	-5	Delete whole para. '.3 162 SUSMA etc.'
15	1	16	4		Replace by amended page
16	2	1	1	-4	Alter '00000000' to '00010000'
17	2	1	1	-1	Alter '01100000' to '01110000'
18	2	1	3	6	Alter 'Emedec' to 'Emidec'
19	2	1	3	7	Alter 'Rool' to'Roll'
20	2	4.3	5	14	Underline 'Mode o'
21	2	4.3	5	-5	Underline 'Mode 2'
22	2	4.3	10	2	Delete 'and 2401' and 's' in Printers
23	2	4.4	3		Replace by amended page
24	2	4.4	5-6		Insert new pages
25	2	4.5	2		Replace by amended page
26	2	4.5	5		Replace by amended page
27	2	4.5	7	-1	Delete line '2201 not yet defined'
28	2	4.6	2-3		Replace by amended pages
29	2	4.6	5		Replace by amended pages
30	2	4.6	7	-2	Alter '01100000' to '01110000'
31	2	4.6	7	-1	Delete '2201: not yet defined' and replace by '1922: 00000000'
32	2	4.7	15	-9	Add 'and must consist only of letters, spaces, the character '-', and must begin with a letter.'

CHANGE	PART	SECT.	PAGE	LINE	
33	2	5	1	4	Alter 'distinguished' to 'distinguishes'
34	3	0			Add to Contents page '0.7 RESERVED LOCATIONS 7'
35	3	0	7		Replace by amended page
36	4	0			Replace Contents page by amended page
37	4	0	1-3		Replace by amended pages 1 - 11
38	5	1	2	4	Alter 'diagrametic' to 'diagrammatic'
39	5	1	3		Replace by amended page
40	5	2	2	10	Alter 'End of File mark' to 'Tape Mark'

I.C.T. SYSTEM DEVELOPMENT ORGANISATION
1900 SERIES REFERENCE MANUAL

AMENDMENT LIST NO.8 — 18th August 1966

---

1. Line 1 is the first line of text following the ruled line.

2. Negative line numbers count upwards from the last line of text (which is '-1') ignoring '1900 PDRM.....'

3. After a written amendment to an existing sheet, add the identity of the Amendment List in the right-hand margin, thus: AL2.

4. Vertical lines in the right-hand margin of a re-issued page indicate where amendments have been made. If there are no lines then the greater part, or the whole, of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page).

5. Update the Amendment Record.

CHANGE	PART	SECT.	PAGE	LINE	
1	1	13	1	-1	Alter 'V is not set.' to 'V does not get set at this stage.'
2	1	13	2	-12	Alter 'NO' to 'BO'
3	1	13	2	-9	Alter 'but' to 'bit'
4	1	13	3	10	Alter 'of' to 'to'
5	1	13	4	2	Alter 'b' to 'B'
6	1	15	5		Replace by amended page
7	2	4.2	7		In heading, alter 'PART 1' to 'PART 2'
8	2	4.3	10	2	After '1930', restore 'and 2401' and alter'Printer' to 'Printers'
9	2	4.4	5		Against 0-8-2 in first table, under 1401, alter '#' to '≠'
10	2	4.5	7	-4	Against 1911, alter '00101000' to '01001000', and '00100000' to '01000000
11	2	4.6	2	4	Delete sentence 'Some punches ... 1401).'
12	2	4.6	3		Delete last line of table 'non-1900...' Codes
13	2	4.6	5	4	Alter 'Mode #0...1900 code)' to 'Mode = 0'
14	2	4.6	5	5	Delete line '#40 Punch... Decoder E)'
15	2	4.7	15	-8	In AL7 amendment, after 'letters' add 'figures,'

I.C.T. SYSTEM DEVELOPMENT ORGANISATION

1900 SERIES PROGRAMMING DIVISION REFERENCE MANUAL

Amendment List No. 9 — 3rd October 1966

1. Line 1 is the first line of text following the ruled line on each page.
2. Negative line numbers count upwards from the last line of text on each page (which is '-1'), ignoring '1900 PDRM etc.' or 'SDORM etc.'
3. After making a written amendment to a page, add the identity of the Amendment List and the change number, in the right-hand margin, thus : AL9/1.
4. Vertical lines in the right-hand margin of a re-issued page indicate where amendments have been made. If there are no lines, then the greater part or the whole of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page).
5. Update the Amendment Record at the front of Volume 1 of the manual.

Change	Part	Sect.	Page	Line	
1	0	0	2	9	Insert 'd' after 'x' to read 'x_d'
2	1	2	2	19	Change 'n_e' to 'x_e'
3	1	3	1	17	Change '.1' to '.2'
4	1	4	3	-4	Change " r = nx*' " to " r = x - nx*' "
5	1	10	2	8	Change '006' and '007' to '106' and '107'
6	1	13	3	14	Change '*' to 'V'
7	2	4.3	11	4	Change 'B10 : Free ' to 'B10 : undefined '
8	2	4.3	11	8	Change 'B16 : Free ' to 'B16 : undefined '
9	2	4.4 ICT 64-Character Card Code			For card-punching 10/6/8, symbol ', change Internal value (octal) '26' to '27'
10	6	1	2	3	Change figure one ('1') in 'F1' to upper-case (capital) 'I' to read 'FI'
11	6	1	2	10	Change figure one ('1') in 'F1' to upper-case (capital) 'I' to read 'FI'
12	6	2	4	15	Change 'LTD' to 'LDT'
13					Insert separator card for Part 7, attached, after Table C in Volume 3, Part 6.
14					Insert new pages for Part 7 attached.
15	Manual Contents Page, Volume 1				Insert 'Part 7 Instruction Times'

I.C.T. SYSTEM DEVELOPMENT ORGANISATION

1900 SERIES PROGRAMMING DIVISION REFERENCE MANUAL

Amendment List No. 10 — 25th October 1966

1. Line 1 is the first line of text following the ruled line on each page.
2. Negative line numbers count upwards from the last line of text on each page (which is '-1'), ignoring '1900 PDRM etc.' or '1900 SDORM etc.'.
3. After making a written amendment to a page, add the identity of the Amendment List and the change number in the right-hand margin, thus : AL 10/1
4. Vertical lines in the right-hand margin of a re-issued page indicate where amendments have been made. If there are no lines, then the greater part or the whole of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page).
5. Update the Amendment Record at the front of Volume 1 of the manual.

Change	Part	Sect.	Page	Line	
1	1	12	2	5	Between lines 5 and 6, insert 'This order does not clear 'C'.'
2	1	13	5		Insert 'AL7' under '1900 PDRM/MBC' at bottom left-hand corner of page.
3	2	4.0 Table A			Insert 'Section 4.0 Page 6 1.10.65' in top right-hand corner of page.
4	2	4.0 Table B			Insert 'Section 4.0 Page 7 1.10.65' in top right-hand corner of page.
5	2	4.0 Table C			Insert 'Section 4.0 Page 8 1.10.65' in top right-hand corner of page.
6	2	4.0 Table D			Insert 'Section 4.0 Page 9 1.10.65' in top right-hand corner of page.
7	2	4.7	11	11	Change 'No' to 'A' to read 'A tape mark precedes the request block.'
8	2	4.8	2	13	Change '5' to '6' to read 'page 6'.
9	4	0	3	17	Delete '1900 series..... item 53' and insert 'Part 0, Section 6' to read '(see also Part 0, Section 6)'.
10	6	2 Table A			Insert 'Section 2 Page 9 1.12.65' in top right-hand corner of page. Insert 'Part 6' in top centre of page.
11	6	2 Table B			Insert 'Section 2 Page 10 1.12.65' in top right-hand corner of page. Insert 'Part 6' in top centre of page.

Change	Part	Sect.	Page	Line	
12	6	2	Table C		Insert 'Section 2 Page 11 1.12.65' in top right-hand corner of page. Insert 'Part 6' in top centre of page.
13	4	0	1 to 11		Remove the whole of this section on Sub-programming and substitute the new pages 1 to 11 attached.
14	4	0	Contents		Against the heading '0.4 General' change page number '2' to '3'. Against the heading '0.6 Reference to common storage areas' change page number '3' to '4'.

NOTE

IF USERS OF THE MANUAL FIND THAT THEY ARE UNABLE TO OBTAIN INFORMATION THEY NEED IN THE MANUAL, WOULD THEY PLEASE WRITE TO: J.M.S. FERGUSSON, EDITOR, SDORM, TECHNICAL INFORMATION BRANCH, ROOM 1A, BHN, GIVING PARTICULARS OF THE INFORMATION THEY REQUIRED.

I.C.T. SYSTEM DEVELOPMENT ORGANISATION

1900 SERIES REFERENCE MANUAL

Amendment List No. 11 — 22 December 1966

1. Line 1 is the first line of text following the ruled line on each page.
2. Negative line numbers count upwards from the last line of text on each page (which is '-1'), ignoring '1900 PDRM etc.'
3. After making a written amendment to a page, add the identity of the Amendment List and the change number, in the right-hand margin, thus: AL9/1
4. Vertical lines in the right-hand margin of a re-issued page indicate where amendments have been made. If there are no lines, then the greater part or the whole of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page).
5. Update the Amendment Record at the front of Volume 1 of the manual.

Change	Part	Sect.	Page	Line	
1	0	6	1 to 3		Remove these pages and insert the new Section 6, pages 1 to 4 attached
2	1	4	1 to 5		Remove these pages and insert the new Section 4, pages 1 to 4 attached
3	1	11	3	-1	After '...sign bit is propagated.' insert 'If N_t = 3, but N_s = 0, V will be unchanged.'
4	1	11	4	18	After 'There is no rounding of the result.'insert 'If N_t = 3, but N_s = 0, V will be unchanged.'
5	2	4.8	11	-2	Change 'W has values 1 - 3' to read 'W has values of 0 to 3.'
6	7	0	2	-1	Delete the footnote following '**' and insert 'These abbreviations are for Compact Store Mode and Extended Store Mode and refer to the operating modes, not the physical nature of the store.'
7	7	0	2	column 2	In the sentence 'MULTIPLY 1902 BY $2\frac{1}{2}$ (EXTRACODE)' change '$2\frac{1}{2}$' to 'APPROX. 3'
8	7	0	3	-1	In the footnote following '*' change 'successful' to 'unsuccessful'
9	7	0	4	column 2	In the sentence 'MULTIPLY....(EXTRACODE)' change '$2\frac{1}{2}$' to 'APPROX. 3'
10	7	0	5	column 2	In the sentence 'MULTIPLY....(EXTRACODE)' change '$2\frac{1}{2}$' to 'APPROX. 3'

I.C.T. SYSTEM DEVELOPMENT ORGANISATION

1900 SERIES REFERENCE MANUAL

Amendment list No. 12 February, 1967

---

1. Line 1 is the first line of text following the ruled line on each page.
2. Negative line numbers count upwards from the last line of text, on each page (which is '-1'), ignoring '1900 PDRM etc.' or 'SDORM etc.'
3. After making a written amendment to a page, add the identity of the Amendment List and the change number, in the right-hand margin, thus: AL 12/1
4. Vertical lines in the right-hand margin of a re-issued page indicate where amendments have been made. If there are no lines, then the greater part or the whole of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page).
5. Update the Amendment Record at the front of Volume 1 of the manual.

Change	Part	Sect.	Page	Line	
1	1	13	6	-4	Alter '.4' to '.5'
2	1	15	1		Replace with new page 1 attached.
3	1	15	5		Replace with new page 5 attached.
4	1	15	8		Replace with new page 8 attached.
5	1	16	2 to 4		Replace with new pages 2 to 4 attached.
6	1	16	7		Replace with new page 7 attached.
7	1	16	10		Alter "1" to read "'" (i.e. single quote mark)
8	2	0	2		Replace with new page 2 attached.
9	2	3	3	-2	Delete "when reading or 3 when writing"
10	2	4.3	2	5	Alter "1903" to read "1930"
11	2	4.3	6		Replace with new page 6 attached.
12	2	4.5	7		Replace with new page 7 attached.
13	2	4.6	7		Replace with new page 7 attached.
14	2	4.7	1 to 19		Replace whole section 4.7 with new 4.7 attached.
15	2	4.8	4		Replace with new page 4 attached.
16	2	4.8	8		Replace with new page 8 attached.

Amendment list No. 12 (cont.)

Change	Part	Sect.	Page	Line	
17	2	4.8	9		Replace with new page 9 attached.
18	3	0	3	8	Alter '01' to read '00'
19	5	1	1,2,3,7,8,		Replace with new pages 1,2,3,7 and 8 attached.
20	5	2	2		Replace with new page 2 attached.
21	5	2	3	-1	Alter" end of file mark" to read "tape mark".
22	5	4	1	-7	Alter "type 2" to read "type 4".
23	5	4	1	-4 to -1	Delete from "If" to "allowed".
24	5	4	6		Replace with new page 6 attached.
25	6	1	2	-5, -4	Delete "only" in line -5 and add "or magnetic tape unit" in line -4.
26	7	0	1,2,3,4,5.		At the foot of each page, in the "For Modification Add" row of the first column, under "1901" change "6" to read "2".

Amendment list No. 13

1. Line 1 is the first line of text following the ruled line on each page.
2. Negative line numbers count upwards from the last line of text on each page (which is '-1'), ignoring '1900 PDRM etc.' or 'SDORM etc.'
3. After making a written amendment to a page, add the identity of the Amendment List and the change number, in the right-hand margin, thus: AL 13/1
4. Vertical lines in the right-hand margin of a re-issued page indicate where amendments have been made. If there are no lines, then the greater part or the whole of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page).
5. Update the Amendment Record at the front of Volume 1 of the manual.

Change	Part	Sect.	Page	Line	
1	0		Contents Page		Insert '7 Instruction Classification.'
2	0		Contents Page		Insert '8 Overflow Indicator.'
3	0	7			Insert new pages 1 to 6 attached.
4	0	8			Insert new page 1 attached.
5	1	13	1	-7	Add 'Upon first activation of a program, the contents of A are indeterminate.'.
6	1	13	7	6	Delete 'X is always zero.' and insert 'X may be 0 or 4; the results conform to the same definition in both cases, but need not necessarily be identical, see (e) above.'.
7	1	13	7	-11	As for Change 6 above.
8	1	15	2 and 3		Substitute new pages attached.
9	1	15	4a	10	Delete 'certain File Storage Devices', and insert ', Magnetic Tape, but not cassette tape.)'.
10	1	15	5	9	Delete 'Bits 18, 19 must always be zero.'.
11	1	15	6	14	After 'type N.', insert 'This does not apply to Direct Access devices,(EDS,MCF,FDS, etc.)
12	1	16	2		Substitute new page attached.

Change	Part	Sect.	Page	Line	
~~13~~	1	16	5	12	Alter '1st January 1900' to '31st December 1899'
~~14~~	1	16	5	13	Alter 'day 0' to 'day 1'
~~15~~	1	16	7		Delete lines 1 to 10
~~16~~	1	16	9,10 & 11		Remove page 9 and insert new pages 9, 10 and 11.
~~17~~	2		Contents	Page	Insert '4.9 Direct Access Devices'
~~18~~	2		Contents	Page	Insert '6 Own Monitoring Facilities'
~~19~~	2	1			Substitute new pages 1 to 5 attached.
~~20~~	2	3	4	3	Delete '0' after 'FAIL'.
~~21~~	2	3	4	-6	Delete 'RPTD' after 'Unit q:-', and insert 'ERR.'.
~~22~~	2	3	7		Substitute new page attached.
~~23~~	2	4.8	9	-10	Change '32768' to read '8191'
~~24~~	2	4.9			Insert New section, pages 1 to 24 attached.
~~25~~	2	6			Insert new section, pages 1 to 5 attached.
~~26~~	3		Contents	page	Insert 'Section 1 Executives'
~~27~~	3	1			Insert new section, pages 1 to 12
~~28~~	5	1	2	-6	Change 'word 11' to read 'word 1'.
~~29~~	5	2	2	-3	Delete from 'If the DUmp.....' to end of page.
~~30~~	5	2	3	1	Delete from 'block type 3' to 'tape'.
~~31~~	5	4	3	9	Delete 'only'.
~~32~~	6	1	2	-5	Delete 'or magnetic tape unit.'.
~~33~~	6	2	10,11 & 12		Substitute new pages attached.

I.C.T. SYSTEM DEVELOPMENT ORGANISATION

1900 SERIES REFERENCE MANUAL

Amendment List No.14 July 1967

---

1. Line 1 is the first line of text following the ruled line on each page.
2. Negative line numbers count upwards from the last line of text on each page (which is '-1'), ignoring '1900 PDRM etc.' or 'SDORM etc.'
3. After making a written amendment to a page, add the identity of the Amendment List and the change number, in the right-hand margin, thus: AL9/1
4. Vertical lines in the right-hand margin of a re-issued page indicate where amendments have been made. If there are no lines, then the greater part or the whole of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page.)
5. Update the Amendment Record at the front of Volume 1 of the manual.

Change	Part	Sect.	Page	Line	
~~1~~	0	5	1	-12	Delete "DGH" and insert "DCH".
~~2~~	1	2	2		Substitute new page.
~~3~~	1	3	2		Substitute new page.
~~4~~	1	12	2		Substitute new page.
~~5~~	1	13	1,2		Substitute new pages 1, 1a, 2.
~~6~~	1	13	5		Lines 4,8,-3;Change "N:" to "N(M) and N*(M)".
~~7~~	1	13	7		Lines 4,9,-2,-7,-12; As for Change 6.
~~8~~	1	15	6		Substitute new page
~~9~~	1	15	9		Substitute new page.
~~10~~	1	16	3	-12	Change "8" to read "12".
~~11~~	1	16	12	~~11~~	Insert new page 12
~~12~~	2	0	2		Remove page 2 and insert new pages 2 and 3
~~13~~	2	3	1,2		Substitute new pages 1 & 2.
~~14~~	2	3	6	5	Change "eas" to read "was".
~~15~~	2	4.1	13	2	Change "Section 5, page 2" to read "Section 5, page 1.".
16	2	4.2	10	2	As for change 15.
~~17~~	2	4.3	11	2	As for change 15.

Change	Part	Sect.	Page	Line	
~~18~~	2	4.4			Substitute new section
~~19~~	2	4.5			Substitute new section
~~20~~	2	4.6			Substitute new section
~~21~~	2	4.7	4	-5	Delete "Undefined" and insert "Bits 0 to 5, geographical address of deck when header label was written. Bits 6 to 23 are undefined. Word 10 undefined".
~~22~~	2	4.7	10	7	Delete "overlay".
~~23~~	2	4.7	13	-1	Add to end of line: "See especially Sect. 1.3(g)."
~~24~~	2	4.7	20	3	Before "subfile" add "simple".
~~25~~	2	4.7	20	15	Before "subfile" add "simple".
~~26~~	2	4.7	23		Substitute new page.
~~27~~	2	4.8			Insert new section pages 1 to 15
~~28~~	4	0	4	-6	Delete "0.9 .10" and insert "sub-sections 0.9 and 0.10".
~~29~~	5	4	3	9.10	Re-insert "only" and delete the words from "issued" to "below".
~~30~~	6	0	4	10	Add "The name EXEC is reserved".

M. Barham

I.C.T. SYSTEM DEVELOPMENT ORGANISATION

1900 SERIES REFERENCE MANUAL

Amendment List Number 15 October, 1967

---

1. Line 1 is the first line of text following the ruled line on each page.
2. Negative line numbers count upwards from the last line of text on each page (which is -1), ignoring '1900 PDRM' etc. or '1900 SDORM' etc.
3. After making a written amendment to a page, add the identity of the Amendment List and the change number, in the right hand margin thus AL15/6.
4. Vertical lines in the right hand margin of a re-issued page indicate where amendments have been made. If there are no lines, then the greater part or whole of the page has been amended (except where an earlier insertion causes overflow of text onto an extra page).
5. Update the Amendment Record at the front of Volume 1 of the manual.

Change	Part	Sect.	Page	Line	
~~1~~	0	6	2	-1	Add "Note that bits 1 to 8 are indeterminate at all times".
~~2~~	0	8	1	-3	Add ", i.e. 070, 074/2, 074/4, 114, 115 and 112 or 113 with $N_t = 3$."
~~3~~	1	2	3	12	Add ", otherwise clear C."
~~4~~	1	2	3	14	Delete "+c" after "n" in both cases.
~~5~~	1	4	2	-8	Add "This instruction is undefined if X = N(M) or X+1 = N(M)."
~~6~~	1	4	4	-4	Add "This instruction is undefined if X = N(M) or X+1 = N(M)."
~~7~~	1	7	3	7	Delete lines 7 to 9 inclusive.
~~8~~	1	11	5	-1	Add "This applies if either or both are operand instructions of an 023 (OBEY), q.v."
~~9~~	1	13	5		Substitute new page.
~~10~~	1	15	1	11	Insert "1." between "where Y =" and "s. 6 bits"
~~11~~	1	15	2		Add after line -1 "4. With the 1901 Handswitch Executive this instruction acts in the same way as the 152 instruction."
~~12~~	1	15	4a		Withdraw existing page 4a, and insert new pages 4a and 4b.

Change	Part	Sect.	Page	Line	
~~13~~	1	15	6	5	Add "at all times".
~~14~~	1	16	1	-4	Delete the whole of note 5. (Lines -4 to -1).
~~15~~	1	16	2-4		Withdraw existing pages 2 to 4 and insert new pages 2, 2a, 3 and 4.
~~16~~	1	16	12		Change page 12 to read page 13.
~~17~~	1	16	11		Withdraw existing page 11 and insert new pages 11 and 12.
~~18~~	2				Substitute new contents page.
~~19~~	2	0	1		Substitute new page.
~~20~~	2	0	2	15	Insert between "disengaged," and "unless its disengagement" the words "(except by the 151 instruction)".
~~21~~	2	4.3	5&6		Substitute new pages.
~~22~~	2	4.5	10	2	Delete "turned offline" and insert "disengaged".
~~23~~	2	4.5	11	13	Change "1 - 20" to read "1 - 160".
~~24~~	2	4.5	11	-7	Change "2103" to read "2102".
~~25~~	2	4.7	1	-8	Add ".4.11 Composite Subfile".
~~26~~	2	4.7	4	-5	Delete "Word 10 - Undefined".
~~27~~	2	4.7	6	-5	Delete "reserved" add insert "user information".
~~28~~	2	4.7	7	-8	After "file" and "or the subfile sentinel in a simple subfile (see 4.7.3.3)."
~~29~~	2	4.7	7	-15	Change "Permanent Program Identifier Block" to "Pre-Request Block Sentinel".
~~30~~	2	4.7	9	-7	Change "listed" to "nested".
~~31~~	2	4.7	9	-6	Add "The rules governing the contents of simple files apply also to simple subfiles."
~~32~~	2	4.7	13		Substitute new page.
~~33~~	2	4.7	19&20		Substitute new pages 19 and 20.
~~34~~	2	4.7	21	1	Delete lines 1 to 8 inclusive.
~~35~~	2	4.8	14		Substitute new page.
~~36~~	2	4.9	20		Withdraw existing page 20 and insert new pages 20 and 20a.

Change	Part	Sect.	Page	Line	
~~37~~	2	4.10			Insert new section.
~~38~~	5	1	1-5		Substitute new pages 1 to 5a.
~~39~~	5	1	7	-7,-9	Delete "word" in both cases.
~~40~~	5	1	8	2,3,17,18	Delete "word" in all cases.
~~41~~	5	2	2	16	Delete "end of file" and insert "tape".
~~42~~	5	2	3	3	Delete "1902/3" and insert "1901/2/3".
~~43~~	5	4	3-6		Withdraw old pages 3 to 6 and insert new pages 3 to 8.
~~44~~	6	1	1	-4	Add ", but does not alter it".
~~45~~	6	2	11		Delete "H Illegal from sub-program".
~~46~~	7	1	1		Substitute new page 1.

I.C.T. SYSTEM DEVELOPMENT ORGANISATION

1900 SERIES REFERENCE MANUAL

Amendment List Number 16 January, 1968

---

1. Line 1 is the first line of text following the ruled line on each page.
2. Negative line numbers count upwards from the last line of text on each page (which is -1), ignoring '1900 PDRM' etc. or '1900 SDORM' etc.
3. After making a written amendment to a page, add the identity of the Amendment List and change number, in the right hand margin thus: AL16/5.
4. Vertical lines in the right-hand margin of a re-issued page indicate where the amendments have been made. If there are no lines, then the greater part or whole of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page).
5. Update the Amendment Record at the front of Volume 1 of this manual.

Change	Part	Sect	Page	Line	
1	0	Contents page			Remove and destroy existing Contents page and replace with new Contents page.
2	0	0	1&2		Remove and destroy pages 1 and 2 and replace with new pages 1 and 2.
3	0	4	1	8	Insert "F N(M)" in the "PLAN Format" column.
4	0	4	1	-13	Add "(except by the 117 (SMO) instruction)" after "... not modifiable".
5	0	4	1	-4	Delete "effect" and insert "be used to affect the".
6	0	8	1		Remove and destroy page 1 and replace with new page 1.
7	0	9			Insert new Section 9.
8	1	2	2		Remove and destroy page 2 and replace with new pages 2 and 2a.
9	1	6	1&2		Remove and destroy pages 1 and 2 and replace with new pages 1 and 2.
10	1	7	1		Remove and destroy page 1 to 4 and replace with new pages 1 to 4.

Change	Part	Sect	Page	Line	
~~11~~	1	11	4	-8	Delete "(Available on 1906 and 1907 only)" and insert "(For availability see Part 0, Section 7)".
~~12~~	1	11	5	13	Delete from line 13 to end of page.
~~13~~	1	11			Insert new page 6.
~~14~~	1	15	3	-7	Add ", or flag-setting devices."
~~15~~	1	16	13		Remove and destroy page 13, and change pages 11 and 12 to read pages 20 and 21.
~~16~~	1	16	1		Remove and destroy pages 1 to 10 and replace with new pages 1 to 19.
~~17~~	2	1	5		Remove and destroy page 5 and replace with new page 5.
~~18~~	2	4.3	8	-3	Change "0 - 15" to read "0 - 95".
~~19~~	2	4.5	9	19	Change "P.F.311" to read "2104".
~~20~~	2	4.5	11		Remove and destroy page 11 and replace with new page 11.
~~21~~	2	4.7	3		Remove and destroy pages 3 to 7 and replace with new pages 3 to 7a.
~~22~~	2	4.7	10	12	Delete lines 12 and 13; insert "Composite subfile 4.7.4.9".
~~23~~	2	4.7	10	26	Add "*" before "Date written".
~~24~~	2	4.7	11		Remove and destroy page 11 and replace with new page 11.
~~25~~	2	4.7	17	-12	Insert "subfile" between "The" and "sentinel".
~~26~~	2	4.7	17	-7	Delete "zero" and insert "Program name (optional on cassette tape)".
~~27~~	2	4.7	18	9	Delete "zero" and insert "Program name (optional on cassette tape)".
~~28~~	2	4.7	20		Remove and destroy pages 20 to 23 and replace with new pages 20 to 22.
29	2	4.9	8	4	Change "Bit 0 : Zero" to read "Bit 0 : Housekeeping Format".

Change	Part	Sect	Page	Line	
~~30~~	2	4.9	8	7	Change "Bits 3 to 5 : Zero" to read "Bits 4 to 5 : Reserved" and insert under "Bit 2 : Fixed ...." the words "Bit 3 : File Organisation Code (Extension)".
~~31~~	2	4.10	3	5	Change "Bit 8 to 15" to read "Bits 8 to 15"
~~32~~	2	4.10	4	6	Change "File Identity" to read "File Name" and insert "K" in the right hand column.
~~33~~	2	4.10	4	7	Delete "Not used" and insert "Logical bucket number of first bucket to store second level overflow, zero if none".
~~34~~	2	4.10	8		Remove and destroy page 8 and replace with new page 8.
~~35~~	2	4.10	11	14	Change "(64 Blocks)" to read "(24 Blocks)"
~~36~~	2	4.10	14	9	Delete "of the Card" and change "Area" to "Card".
~~37~~	2	4.10	15	7	Change "SYSTEMVFILE" to read "SYSTEMSVFILE"
~~38~~	2	4.10	22	15	Change "Index of MCF" to read "Index on MCF"
~~39~~	2	5	1	3	Delete "processors 1904 upwards" and insert "dual programming and multi-programming Executives".
~~40~~	3	0	3	-3	Change "1902/3" to read "1901/2/3".
~~41~~	4	0	1	-13	Insert "H" after "EX1".
~~42~~	5	Contents page			Remove and destroy existing Contents page and replace with new Contents page.
~~43~~	5	1	3		Remove and destroy pages 3 to 5 and replace with new pages 3 to 5.
~~44~~	5	3	2	7	Add "(see the 155 instruction)".
~~45~~	6	2	4	11	Delete "TAPE" which appears after "LDT file-name".
~~46~~	6	2	10	2	Delete "MK. 2" which appears after "..... EXECUTIVE".
~~47~~	7	0	2	-5	Change ".... the second one applies to successful jumps" to read".... the second one applies to unsuccessful jumps".

I.C.T. SYSTEM DEVELOPMENT ORGANISATION

1900 SERIES REFERENCE MANUAL

Amendment List Number 17 March, 1968

---

1. Line 1 is the first line of text following the ruled line on each page.
2. Negative line numbers count upwards from the last line of text on each page (which is -1), ignoring '1900 PDRM' etc. or '1900 SDORM' etc.
3. After making a written amendment to a page, add the identity of the Amendment List and change number, in the right hand margin thus: AL17/5.
4. Vertical lines in the right-hand margin of a re-issued page indicate where the amendments have been made. If there are no lines, then the greater part or whole of the page has been amended (except where an earlier insertion causes overflow of text on to an extra page).
5. Update the Amendment Record at the front of Volume 1 of this manual.

Change	Part	Sect	Page	Line	
1	1	16	20		Remove and destroy page 20 and replace with new page 20.
2	2	1	3		Remove and destroy page 3 and replace with new page 3.
3	2	4.7	6a	-4	Insert "*" at beginning of line.
4	2	4.7	7	12	Delete this line.
5	2	4.7	7	-9	Add "and Supplementary Request Blocks".
6	2	4.7	17	-7	Change "Program name (optional on cassette tape)" to "Program name (optional)".
7	2	4.7	18	9	Change "Program name (optional on cassette tape)" to "Program name (optional)".
8	2	4.7	20	3	Delete "either" and insert "the latter".
9	2	4.7	22		Remove and destroy page 22 and replace with new page 22.

Change	Part	Sect.	Page	Line	
~~10~~	2	4.9	5	9	Insert "(Additive Mode)" between "#7" and "Write a bucket ...", and change "#7" to "#40".
~~11~~	2	4.9	17	11	Delete lines 11 and 12 and insert "Type/#40 : Suppress check read (null on M.C.F.)".
~~12~~	2	4.9	17	15	Delete "and 7".
~~13~~	2	4.9	18		Remove and destroy page 18 and replace with new page 18.
~~14~~	2	4.10	10		Remove and destroy page 10 and replace with new page 10.
~~15~~	2	4.11			Insert new section 4.11 (Direct Access Formats).
~~16~~	3	1			Remove and destroy Section 1 (pages 1 to 12) and replace with new Section 1 (pages 1 to 28).
~~17~~	5	1	2	-4	Delete "alphanumeric characters," and insert "letters and numbers".
~~18~~	5	1	2	-2	Delete "alphanumeric characters," and insert "letters and numbers".
~~19~~	5	1	3	2	Delete lines 2 to 4 inclusive.
~~20~~	5	1	5a	-1	Change "0 to 6" to "1 to 6".
~~21~~	5	4	5		Remove and destroy page 5 and replace with new page 5.

PREFACE

This manual is intended to serve as an up-to-date, detailed work of reference for the 1900 series in all aspects relevant to programming. It is not intended as a training manual.

Considerable attention has been paid to the question of compatibility. The order code, for example, is defined as far as it is compatible throughout the series, to the exclusion of quirks peculiar to individual models. Hence, if a program is to work correctly on any appropriate 1900 configuration, it must not use any features not defined here.

Numbered amendment lists will be issued from time to time. Whenever such a list is incorporated, the Amendments Record at the beginning of the manual should be updated and the Amendment List filed behind the Amendments Record.

AMENDMENT RECORD

Whenever amendments to this manual are incorporated from a numbered Amendment list, details should be recorded in this section, and the Amendment List filed.

No. of list	Date of list	Incorporated By	Date
0	1-OCT-65		
1	1-NOV-65		
2	6-DEC-65		
3	14-MAR-66		
4	1-APR-66		
5	14-MAY-66		
6	1-JUNE-66		
7	27-JULY-66		
8	18-8-66	BJ	1-67
9	3-10-66	BJ	1-67
10	25-10-66	BJ	1-67
11	22-12-66	BJ	1-67
12	2-67	BJ	5-10-67
13	4-67	BJ	5-10-67
14	6-67	BJ	6-10-67
15	10-67	BJ	28-12-67
16	1-68	BJ	15-3-68
17	3-68	BJ	5-6-68

No. of List	Date of List	Incorporated By	Date

CONTENTS OF THE PRESENT ISSUE

PART 0	General
PART 1	Order Code
PART 2	Peripherals
PART 3	Executive
PART 4	Sub-programming
PART 5	Binary Programs
PART 6	Operator Communication
PART 7	Instruction Times

PART 0 CONTENTS

Section	0	Notation
	1	1900 Series word
	2	Word as a number
	3	Word as miscellaneous entities
	4	Word as an instruction
	5	Modification
	6	Reserved locations
	7	Instruction classification
	8	The Overflow Indicator and Carry Register
	9	Extended mode facilities

SYMBOL	MEANING
A	the floating point accumulator
a	the contents of A
Bi	the i^{th} bit of a word
C	the carry register
c	the contents of c (0 or 1)
e	the exponent of a floating point number
E	e + 256
F	a function
M	a modifier (registers 1 to 3)
m	the contents of the modifier register (zero if M=0)
N	a core store address or 12-bit number
N(M)	the modified core store address or 15-bit number
n	the contents of N after modification if applicable
n'	the contents of N (after modification if necessary) after an instruction has been obeyed
n:	the double-length contents of N and N + 1
N_{em}	a 22-bit address
N_m	a 15-bit address
N_r	a 14-bit address
N_t	the most significant 2-bits of the 12-bit N address
N_s	the least significant 10-bits of the 12-bit N address
R	the branch specifying bit in Extended Branch Mode

SYMBOL	MEANING
S	the sign bit
V	the overflow register
X	an accumulator (registers 0 to 7)
X*(X+1)	the accumulator adjacent to X (X7* = X0)
x	the contents of X
x*	the contents of X*
x'	the contents of X after an instruction has been obeyed
x:	the double-length contents of two consecutive accumulators
x:'	the double-length contents of two consecutive accumulators after an instruction has been obeyed
x_a	the least significant 12 bits of x
x_c	the 9-bit counter at the most significant end of x
x_d	the least significant 7 bits of x_c
x_e	the least significant 9 bits of x. The exponent of a floating-point number occupies this portion of the second word
x_{em}	the least significant 22-bits of x
x_j	any one of x0, x1, x2, x3 the four 6-bit characters of x
x_k	the most significant 2 bits of x
x_m	the least significant 15 bits of x

The last 14 symbols and their corresponding meaning could also apply to N or modified N.

1900 Series Word

1900 Series Word:

The 1900 Series word is 24 bits long. The bits of a word are referenced as B0, B1,....., B23 starting with the left most bit and progressing to the right.

A word (or words) might be interpreted as one of the following:

(1) A fixed point number

(2) A floating point number

(3) A string of characters.

(4) A logical expression

(5) A data-packed word

(6) An instruction

Convention for sign:

When a word is interpreted as a number, B0 is used to indicate its sign. The number is interpreted as negative if B0 = 1 and positive otherwise.

.1 Fixed Point Numbers

Throughout this manual while describing the arithmetic operations it has been assumed that the word holds a binary fraction.

When the ~~conceptual~~ binary point is assumed to be in a fixed position anywhere within the word, the word is said to hold a fixed point number.

(a) Fixed Point Single-length Numbers

Only one word is used to store the number.

According to the position of the binary point the numbers are classified as follows:

(i) Fixed Point Integer

If the binary point is immediately to the right of B23 in a word, the word holds a fixed point integer. Fixed point integers lie in the range -2^{23} to $2^{23}-1$.

(ii) Fixed Point Fraction

When the binary point is assumed to be between B0 and B1 of a word, the word represents a proper fraction. Fixed point fractions lie in the range -1.0 to $1.0 - 2^{-23}$

Note: Binary point could be assumed to lie anywhere within the word. In such a case the word would represent a mixed number. Under these circumstances the onus of keeping track of the binary point is on the user.

(b) Fixed Point Double-length Number

Two words are used to store a number.

B0 of the higher numbered word is not used for the purpose of storing a double-length number. B0 of the higher numbered word must be zero.

According to the assumed position of the binary point the numbers are classified as follows:

(i) Fixed Point Integer

If the binary point is immediately to the right of B23 in the less significant word, the two words together represent a fixed point double-length integer. The fixed point double-length integers lie in the range -2^{46} to $2^{46}-1$.

(ii) Fixed Point Fraction

When the binary point is assumed to lie between B0 and B1 of the lower numbered word, the double length word represents a proper fraction. The fixed point double-length fractions lie in the range -1.0 to $1.0 - 2^{-46}$.

(iii) Mid-point Number

A mixed number containing an integral and a fractional part can be stored in two words, the second word holding the fractional part. In this case the binary point could be considered to be between B23 of the first word and B1 of the second word. The range for mid-point numbers is -2^{23} to $2^{23} - 2^{-23}$.

.2 Floating Point Numbers

To obviate the difficulty of keeping track of the binary point while processing fixed-point numbers, floating-point arithmetic is used and numbers are stored in floating-point style (see 13.1).

Note: When a fixed-point number exceeds capacity overflow indicator is usually set. See glossary.

.1 4 - Character Word

A word might consist of 4 characters each of 6 bits. The layout of the characters is as follows:

Bit Nos.	Character
B0 - B5	First
B6 - B11	Second
B12- B17	Third
B18- B23	Fourth

.2 Logical Word

A word might also be interpreted as a configuration of 0's and 1's. In such a case the word could be considered as a logical expression. B0 in this case is not interpreted as a sign bit nor the content of the word as a numerical quantity.

.3 Counter Modifier

(i) Normal Counter Modifier:

A word might be used partly as a counter and partly as a modifier. 9 bits from B0 to B8 are used as counter and the rest of the bits B9 to B23 as modifier.

(ii) Character Counter Modifier:

When a character, rather than a word, has to be modified B0 & B1 are used as character modifier, B2 to ~~B6~~ B8 as counter and B9 to ~~B15~~ B23 as word modifier. AL 5

.4 Packed Data

The configuration of bits might be of interest in groups. Thus the word consists of several items of data packed into it.

Word as an instruction

When a computer word is interpreted as an instruction it is decoded according to a specific format. Different types of instructions have different formats as shown in the following table.

Function	Field Name	Field Symbol	Bits Range	No. of Bits	PLAN Format
Normal Order	Accumulator	X	B0-B2	3	
	Order	F	B3-B9	7	
	Modifier	M	B10-B11	2	F N(M) AL/16
	Address	N	B12-B23	12	F X N(M)
Jump Order	Accumulator or Condition*	X	B0-B2	3	
	Order	F	B3-B8	6	
	Address	N	B9-B23	15	F X N / F N

Note: Jump Orders are not modifiable, except by the 117(SMO) instruction. AL/16

Shift Order	Accumulator	X	B0-B2	3	
	Order	F	B3-B9	7	
	Modifier	M	B10 & B11	2	
	Type	N_t	B12 & B13	2	
	Shift Count	N_s	B14-B23	10	F X N_s(M)

Note: When the shift-order is modified the modification should not be used to affect N_t field. AL/16

* When the accumulator field is not used for normal purposes, it acts as function extension e.g. in order 074 and 076.

Word as an instruction

Function	Field Name	Field Symbol	Bits Range	No. of Bits	PLAN Format
Floating Point Order	Function extension (if any)	X	B0 - B2	3	
	Order	F	B3-B9	7	
	Modifier	M	B10 & B11	2	
	Address	N	B12 - B23	12	F X N(M)/ F N(M)

Note: Unless otherwise stated all instructions are modifiable.

This term refers here to the facility by which the N field in an instruction is modified (i.e. increased or decreased) at the time it is obeyed, the value of N in the stored instruction remaining unaltered. Modification may be done in 3 ways, of which (i) and (ii), or (ii) and (iii) can be used together on the same instruction:

(i) By specifying in the M field X1, X2, or X3 (not possible in a jump instruction, which has no M field). The effect of this is to add, without any carry beyond, the least significant 15 bits (22 in Extended Data Mode) of the modifier to the 12 bits of the N field of the instruction.

(ii) (On certain processors only) By a 117 (SMO) instruction, whose effect is to add the least significant 15 bits (22 in Extended Data Mode) of the word whose address is N(M) to the 12 or 15 bits of the N field of the following instruction (which must not itself be a 117 instruction).

Since a 117 instruction is itself modifiable by method (i), in 1906 and 1907 processors simultaneous triple modification is possible.

(iii) Modification of character manipulation in instructions 024 (LDCH), 034 (~~DCH~~ DCH), 043 (CDB), and 047 (CBD). [AL14]
This method operates with X1, X2, or X3 as a modifier as in (i), but in addition the character portion of this (viz bits 0-1) specifies absolutely the character to be loaded. (The 117 instruction ignores bits 0-1).
Note that the effect of these four instructions when unmodified is on character 3 (bits 18-23) of the operand, whereas when modified by a null character portion it is on character 0 (bits 0-5).

If the modifier in (i) (ii) or (iii) is negative (viz as a complement to 2^{-8} or 2^{-1}, in compact or extended mode respectively), its modifying effect will be to decrease the resultant N field.

Reserved Locations.

Certain locations at the low-address end of the object program area have special uses and their use by normal object programs is, therefore, restricted to particular purposes or prohibited entirely as stated in this section.

'Use' in this context includes, except where stated to the contrary, the following :

(a) use as operands (addressed directly or indirectly).
(b) use as control, flag or data areas for peripheral transfers.
(c) use as instructions (obeyed directly or indirectly).
(d) being overwritten as a result of a 154 (CONT) instruction.
(e) being the subject word of an ALTER directive.

Certain orders of the 1900 Series Order Code have further restrictions imposed, as stated in the definitions of the individual orders. Such further restrictions apply in addition to the general restrictions stated in this section.

The effect of violating the restrictions stated in this section is indeterminate. That is, the results cannot be predicted and are liable to differ between the various processor/executive environments, or even between two runs of the same program in the same processor/executive environment.

Any relaxation of these restrictions peculiar to individual orders will be stated in the definitions of these orders.

Unconditional Restrictions

The locations listed below are subject to the restrictions stated for all members of any normal object program.

Location address(es)	Purpose and/or restrictions
0 - 7	These are the program member's accumulators and can only be used as stated below : (a) as operands, in both X and N fields, except that, where the N operand of an instruction is interpreted as a double-length operand, as in Floating Point orders, N(M) must not address location 7. (b) as counter-modifiers. (c) as data areas for instructions 116 (MVCH), 126 (MOVE) and 127 (SUM), provided that the whole of such a data area lies within locations 0 to 7. (d) as the location containing the actual jump address for a Replaced (Indirect) jump, i.e. the location addressed by the N field of a Replaced (Indirect) jump order. This use is only applicable when the object program is working in Extended (Re-interpreted) Jump Mode. These locations must never be used as the ultimate address of a jump order.
8	This location is used by both Executive and the processor hardware for control purposes and must never be used by an object program in any way.
9	This location is used by Executive for the 'direct response' to certain orders and also, in part, by the hardware of certain processors for control purposes. Object programs must never alter the contents of this location, but may examine the location in accordance with the definitions of the appropriate direct responses. NOTE that bits 1 to 8 are indeterminate at all times. [A4/15]

Location Address(es)	Purposes and/or restrictions
9 (continued)	After an order which gives such a direct response, those parts of location 9 which are defined in that response will only remain unchanged until the object program member concerned issues either another order which gives a significant direct response, or any order in groups 15 and 16, regardless of whether or not that order has a significant direct response. The contents of those parts of location 9 which are not part of the defined response are liable to change at any time and must always be regarded as indeterminate.
10 and 11	These locations are reserved for communication between certain software packages and parts of the object program written by the user. User-written program segments may only use the locations for such communication in accordance with the specifications of the software packages.
12 and 13	These locations are reserved for Executive and must not be used by an object program in any way.
14 and 15	The same restrictions apply as for locations 10 and 11.
16 to 29	These locations are reserved for use by the various compilation systems. Object programs must only use them in the way defined by the compiler specifications.
30	This location contains the 24 'operator's switches'. The switches are numbered from 0 to 23 from the most significant end, thus switch 0 corresponds to Bit 0, switch 1 to Bit 1, etc. The 'ON' state of a switch is represented by the appropriate bit being in the 'one' state. Object programs may examine and/or alter the state of any bit or bits of this location. It is also permissible for an ALTER directive to change the contents of this location. Note: Certain operating system environments may impose restrictions on the use of these 'operator's switches'.
31	The same restrictions apply as for locations 16 to 29.

Note

Locations 0 to 15 inclusive are, in effect, individual to each member of a program, whereas locations 16 onwards are common to all members.

Conditional Restrictions

The locations listed below must not be used by any member of an object program for any purpose whatsoever when the conditions described exist.

Location Address(es)	Purpose and/or restrictions
32 to 63	The user of these locations by an object program is prohibited when that program has more than one member.
64 to 79	The use of these locations by an object program is prohibited when that program has three or more members.
80 onwards	In the case of object programs with more than three members, each extra member causes a further 16 words to be withdrawn from object program use. Such extra areas are arranged consecutively starting from location 80.

Note

In this context the number of members in an object program is deemed to be 'Z + 1', where 'Z' is the highest numbered member which 'exists' in accordance with the definition in the SDORM, Part 4, Section 0.3.3 (see also Part 5, Section 1.2).

Instruction Classification

.1 Functional Grouping of Instructions

The instructions in the 1900 Series Instruction Code may conveniently be grouped according to their functions as follows:-

a) Internal (instructions 000-137, 162-164), subdivided into:-
   i) Primary (instructions 000-127, except 076,114,115)
   ii) Floating Point (instructions 076, 114, 115, 130-137)
   iii) Subprogram Control (instructions 162-164)

b) Peripheral Device Control (instructions 150-157, 164)

c) Operating Environment Communication (instructions 160, 161, 165,166)

d) Privileged (to be defined)

The availability of instructions in various environments is defined according to the classifications described in the following paragraphs. An instruction which is not available will be monitored as illegal to the operating environment when an attempt is made to obey it.

.2 Instruction Code Levels

The following levels define the availability of primary instructions and of the addressing capabilities available to all instructions.

.2.1 Level A

Includes primary instructions other than 066, 116, 117. Addressing takes place in 15-bit address mode (15AM) and direct branch mode (DBM) only; 22-bit address mode (22AM) and extended branch mode facilities (EBM) are not available.

.2.2 Level B

Includes all primary instructions. Addressing is as for Level A. Level A is thus a subset of Level B, differing only in the non-availability of instructions 066, 116, and 117.

.2.3 Level C

Includes all primary instructions as for Level B. 22AM and EBM are available.

.3 Instruction Code Sub-levels

Levels A, B, and C are qualified by a sub-level numeral 1 or 2 to indicate the availability of Normalise instructions (114, 115) independently of the

.3 Cont'd

Floating Point option (see below)

.3.1 Levels A1, B1, and C1

Normalise instructions are not available unless the Floating Point option is available.

.3.2 Levels A2, B2, C2

Normalise instructions are available even though the Floating Point option may not be available.

.4 Instruction Code Options

.4.1 Floating Point Option

When this option is included, instruction 114, 115, 130-137 are always available. If the instruction code level is B or C, instruction 076 is also available. Implementation of the Floating Point option takes one of two formats as follows:-

a) Floating Point (Standard)

132-135 require operands to be in normalized form, and the values of x in the instruction are restricted to 0 and 4.

b) Generalized Floating Point

Instructions 132-135 will accept operands in normalized or unnormalized form, and the values of x in the instructions may range from 0 to 7.

.4.2 Sub-program Option

When this option is included,instructions 162 to 164 are always available. Implementation of this option takes one of two forms as follows:-

a) Sub-Programs (Standard)

Definitions for instructions 162 to 164 are in accordance with the "New" Symmetrical member systems only.

b) Combined Definition Sub-Programs

Combined "Old" and "New" definitions for instructions 162-164 are available.

.5 Privileged Instructions

Additional instructions will be available in certain environments to privileged or "Trusted" programs, to enable these programs to carry out some or all of the tasks regarded as being the responsibility of an Operating System. Series-compatible definitions, which may be optional, will become available at a later date.

.6 Operating Environment Communication

Instructions 160, 161, 165 and 166 are always available to non-privileged object programs. While their external effects are necessarily dependent upon the operating environment, the definition of their use by such programs is compatible between all environments. Any instructions monitored as illegal may also be regarded as (involuntarily) communicating with the Operating Environment.

Environments for which variations in external effects exist are:-

a) Handkey - Controlled Executives

b) Typewriter - Controlled Executives

c) George series of Operating Systems

.7 Peripheral Devices

Peripheral devices are classified, for the purpose of the instruction code, by the internal peripheral type numbers assigned to them. The instructions available for each internal type of peripheral, together with any variations due to differences in peripheral models within a single internal type, are defined so at to permit maximum compatibility to be achieved:-

a) between similar peripheral models with a single internal type

b) for a single peripheral model in all operating environments

A program's environment specification should indicate any restrictions that exist in respect of a) or b) above on account of the use of features not common to all peripheral models of a type, or to all operating environments.

.7.1 Instruction Code Options

Certain optional features are associated with the definitions of peripheral control instructions, for example:-

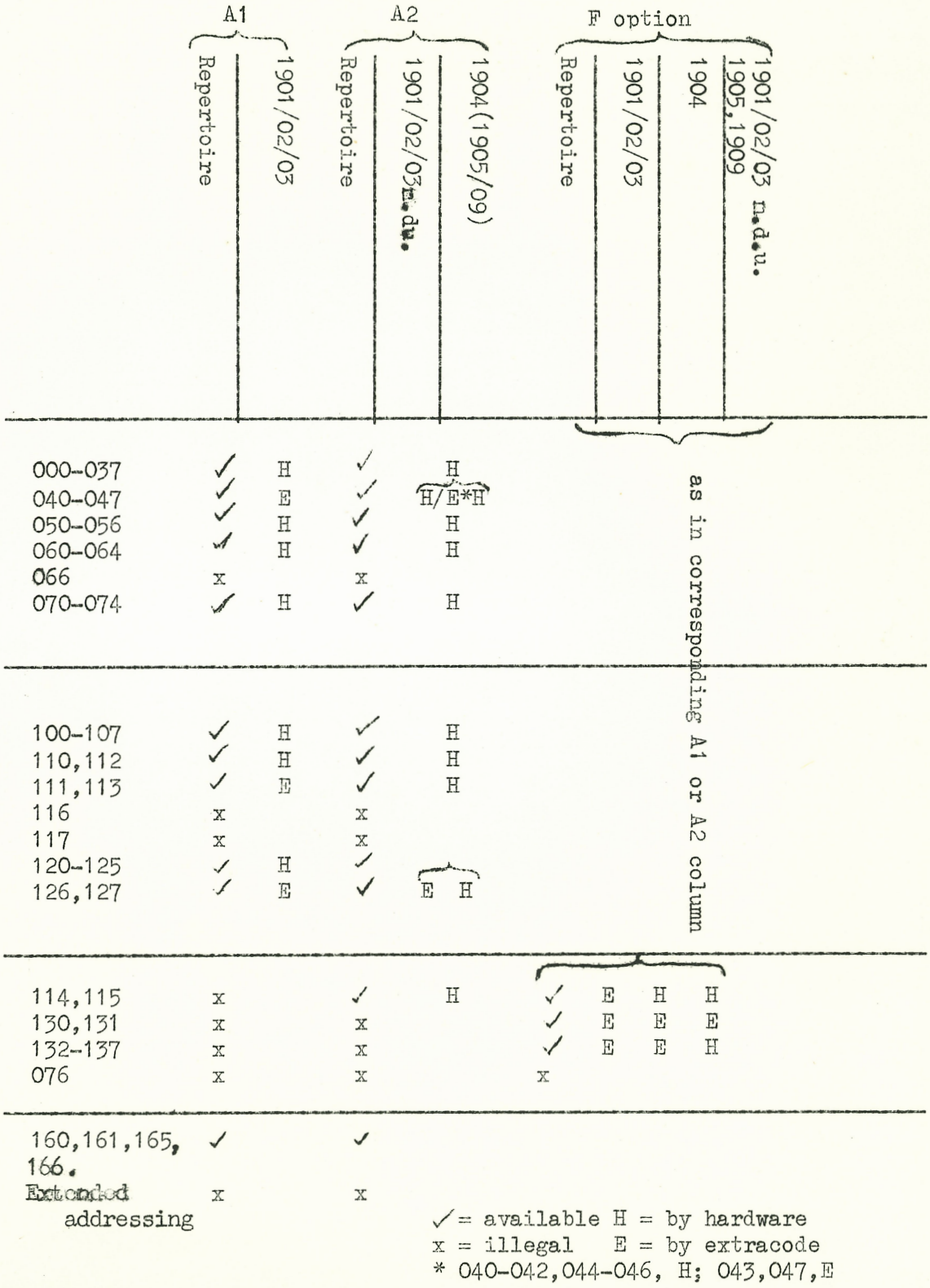
a) Own-Monitoring

b) Direct-Response Control

If such features are necessary for the proper operation of a program, this should be stated in its environment specification.

It should be noted that the definitions of peripheral control instructions available to privileged programs may differ in some respects from those available to other object programs.

Instruction Classification



✓ = available H = by hardware
x = illegal E = by extracode
* 040-042,044-046, H; 043,047,E

Instruction Classification

	B1 RESERVED FOR FUTURE PROCESSORS	C1 RESERVED FOR FUTURE PROCESSORS	C2 Repertoire	C2 1904 E/F, 1906	F option Repertoire	F option 1904E/F, 1906	F option 1905E	F option 1905F, 1907
000-037			✓	H	as in corresponding C2 column			
040-047			✓	H				
050-056			✓	H				
060-064			✓	H				
066			✓	H				
070-074			✓	H				
100-107			✓	H				
110,112			✓	H				
111,113			✓	H				
116			✓	H				
117			✓	H				
120-125			✓	H				
126,127			✓	H				
114,115			✓	H	✓	H	H	H
130,131			x		✓	E	E	H
132-137			x		✓	E	H	H
076			x		✓	E	E	H
160,161,165, 166.			✓					
Extended addressing			✓					

PF.281 is shown as 1904E and F, 1905E and F.

✓ = Available. H = by hardware. x = illegal. E = by extracode.

The Normal Overflow Indicator

1. Any condition defined as causing overflow will set V (the normal overflow indicator), unless V is already set, when it will remain set. (Effectively, a logical OR is performed between V and 1).
2. Once set, V will only be unset by instructions which are specifically defined as capable of unsetting it, i.e. 070, 074/2, 074/4, 112, 114 and 115 or 113 with $N_t = 3$.
3. The initial state of V when a program is first entered is undefined.

The Carry Register

1. The Carry or C register is a special one-bit register associated with, but outside, each member of a program. Carry is set (C = 1) when the result of carrying out an instruction in multi-length working exceeds the capacity.
2. The instructions which can set C or leave it set are 004 to 007, 014, to 017, 105 to 107, 026 and 027.
3. Instructions which always preserve C are 023, 123 and 117.
4. All other instructions clear C.

.1 Introduction.

Two enhancements are available for use on certain 1900 processors, which allow the attachment of core stores in excess of 32K words. These facilities are now available on the 1904E/F, 1905E/F and the 1906/7 processors.

The enhancements are 22-bit Address Mode and Extended Branch Mode. 22-bit Address Mode permits stores of more than 32K to be used as data areas, and Extended Branch Mode allows these stores to contain program instructions.

22-bit Address Mode and Extended Branch Mode are completely different facilities, and neither their availability or setting is dependent on the presence of the other.

.2 Terminology.

15AM - 15-bit Address Mode.
22AM - 22-bit Address Mode.
DBM - Direct Branch Mode.
EBM - Extended Branch Mode.
Relative Branch } The two types of branching employed in Extended Branch
Replaced Branch } Mode.

.3 Concept of Operating Modes.

The changes to the instruction code and addressing system on the 1904E/F, 1905E/F and 1906/7 to accomodate 22AM and EBM do not impair these processors' ability to run programs in 15AM and DBM.

Operating modes indicate the instruction codeand addressing features assumed by the program, and form part of the information stored by Executive for each program. When a program is activated, the machine is switched into the appropriate operating mode.

(.3 cont'd.)

The 1904E/F, 1905E/F and 1906/7 Executives store the operating mode when a program is loaded, and a means is provided to the program for changing its mode during running. The definition of the initial mode setting process ensures that existing programs are treated as being in 15AM and DBM, which are the only address and branch modes available on the smaller processors.

.4 22-bit Address Mode.

.1 In 22AM the store area for the program is treated as having a 22 bit address. (Direct branch addresses of 15 bits and unmodified data addresses of 12 bits restrict the usage of part of the total area). All modification, whether by premodify (117) orders or by an index register quoted in the instruction, is carried out using 22 bits from the modifier word.

Thus in 22AM a complete word is required for a character address, and only the most significant two bits are not used in a word address.

.2 Count instructions: because bits 2 to 8 form part of the address in 22AM working, the 9 bit/15 bit division of a word (count/modifier) cannot be used if the word is required as a modifier. The action of the count instructions 060(BUX), 062(BDX) and 064 (BCHX) is therefore different. In 22AM these instructions increment the address in X, and branch unconditionally to N.

Instruction 066 (BCT) may be used in conjunction with these 'increment only' instructions, the counting being carried out in a separate accumulactor.

.3 Subroutine Entry and Exit: the format of the link in X must allow for a 22 bit address. The setting of the zero-suppression mode is stored in Bit 1 on entry instead of Bit 8, and the remaining contents of the mode setting are lost. On exit the zero-suppression setting is restored from Bit 1.

(.4.3 cont'd.)

Note: It is therefore inadvisable, unless it is known that the zero suppression mode is not set, to enter a subroutine in 22AM and leave it in 15AM since the zero-suppression will be restored incorrectly; if entry is in 15AM and exit in 22AM the mode setting will be treated as part of the link address.

.4 Console displays: the 160 instruction (DISTY, SUSTY, DELTY) outputs a message on the console typewriter. In 15AM the address and length of the message is given by a control word in the count/modifier layout. In 22AM a word pair can be used, the first containing the number of characters in the message and the second the start address.

.5 Other instructions (clarification):-

i) 037(DLA) - this instructions continues to transfer a 15 bit field from X to N.

ii) 126(MOVE), 116(MVCH), 127(SUM) - the addresses in X and X+1 are in effect modifiers.

iii) 165(GIVE) with N(M) = 3 - the core store area requested is a 22 bit quantity.

iv) 157(PERI) - the address in the fourth word (word 3) of a 157 control area is in 22 bits. The reply word (word 1) in cases where a "next address" rather than an "unexpired count" is given (e.g. magnetic tape decks), uses only the least significant 18 bits. Because the maximum value of the count is limited, there is no ambiguity, but the true "next address" may have to be derived by a simple calculation.

.5 Extended Branch Mode (EBM).

.1 EBM is a development to permit program instructions to be obeyed in the extended area (i.e. above the first 32K words of the program's area).

(.5 cont'd.)

.2 EBM introduces two new types of branching, relative and replaced. In the former, jumps of limited extent are carried out relative to the position of the branch instruction (forwards or backwards). If the extent of the jump may be beyond the scope of the relative address, the replaced form allows reference to a location containing the full 22-bit address of the destination; this in effect "replaces" the address given in the instruction. (This technique is alternatively known as indirect addressing).

Bit 9 of the instruction is used to distinguish between the two cases:

Bit 9 = 0 Bits 10 to 23 are taken to be a signed relative address.

Bit 9 = 1 The address in bits 10 to 23 indicates the "replacer" which contains the address of the destination.

It will be observed that a section of code containing only relative branches (and which does not amend itself) may be obeyed in any part of the store, and indeed may be obeyed in different positions on different occasions or even moved during execution.

The PLAN assembly process translates branch instructions into relative branches whenever possible, as these give better efficiency both in space and time.

.3 023 (OBEY) and branch instructions: if a 023 addresses a relative branch instruction, the operation is performed relative to the 023, not the location of the branch instruction. Thus a branch to a labelled position, for example, will not be correctly performed.

Note: It is necessary for the programmer to be aware of this potential trouble, since the assembly process cannot discover it. A means is provided of indicating that a replaced branch is required instead. This is one of the principal points to beware of when

(.5.3 cont'd.)

existing PLAN coding is to be reassembled in EBM.

.4 Subroutine links: in 15AM and EBM, the subroutine link format is as for 22AM (see .4.3 above), and similar considerations apply.

.6 Setting and Switching of Modes.

.1 Mode word: this word contains two bits which convey the setting of address and branch modes:

Bit 21 = 0 indicates DBM
= 1 indicates EBM
Bit 23 = 0 indicates 15AM
= 1 indicates 22AM

The remaining bits are undefined (i.e.they should be set to zero when a mode word is produced, but should not be assumed to be zero when a mode word is examined).

.2 Mode setting on program loading: a Supplementary Request Block is used to hold the mode setting information for a program. It will be produced by the compiling/assembly process, or when a program is dumped, for any program not in 15AM or DBM. Being a new type of block, it will cause rejection of the program by processors not having 22AM and EBM facilities. (For further information on Supplementary Request Blocks see Part 5, Section 1, Page 5a).

.3 Mode setting during program execution: the 165(GIVE) instruction is provided for enquiring the current mode setting and for changing the mode setting (see Part 1, Section 16):

GIVE with N(M) = 8 Enquire mode setting. A reply is given in X, in the form of a mode word for the current setting.

GIVE with N(M) = 9 Change mode setting. A mode word with the required new setting must be in X, and a reply is given in X in the form of a mode for the setting actually achieved.

(.6.3 cont'd.)

On processors without 22AM or EBM capability, the reply to these instructions will always be zero. The action to be taken in the event of a change being unsuccessful is left to the program.

.7 Multi Member Programs.

Executive: the members of a multi-member program are treated independently, and may run in different operating modes. The instruction to enquire or to change mode setting (165 with N(M) = 8 or 9) applies only to the member which issues them.

The initial mode setting of member 0 is determined by the Supplementary Request Block (see .6 above). Members other than member 0 obtain their initial mode setting from the mode of member 0 at the time of first activation by 163(AUTO) X N(M) $\neq$ 0.

Use is made of the fact that when a multi-member program is to be dumped, the members other than member 0 must be inactive.

.8 Detailed Definitions.

.1 Notation

DN - operand address as derived in .2 below.

BN - branch address as derived in .3 below.

I - instruction address.

p - premodifying value specified by preceding 117 (SMO) instruction (0 if none).

q - the number of bits to which an address (inside parenthesis) is truncated. q is dependent on the Addressing Range Mode in which the program is running i.e. 15 in 15AM; 22 in 22AM.

→22← implies extension to 22 bits by propagating the most significant bit of the value concerned.

(.8.1 cont'd.)

(For the meaning of other notation used below see Part 0, Section 0)

.2 Operand Address DN (Normal instruction).

The operand DN (which is in most cases an address, but may be a literal N) is defined as follows:-

M = 0 DN = (N + p)q

M = 1, 2 or 3 DN = (N + m + p)q

Note: i) If the most significant bit of (m)q or (p)q is 1 the effect will be as though the value were negative. The sum (N + p) or (N + m + p) is truncated at 15 or 22 bits according to q.

ii) If the resultant DN is 'negative', then when used as an address it may cause a reservation violation, or when Datum has been added may lie beyond the lower end of store.

.3 Branch Address BN

The formula for BN depends upon the current setting of the branch mode, and in EBM on the value of R (Bit 9) of the instruction.

Note: The definitions below do not apply in the case of the 072(EXIT) instruction (see Part 1, Section 7, Page 1).

(a) Direct Branch (DBM)

$$BN = (N_m + (p)q\ )q$$

Note: i) p can be negative, with normal cyclic effect beyond q bits.

ii) BN must satisfy the reservation check. (On some processors, BN and I+1 must both always lie within reservations.

iii) q will be 15 or 22 depending on the Addressing Range Mode used.

(b) Relative Branch (EBM and R = 0)

$$BN = (I + N_{r \rightarrow 22} + (p)q_{\rightarrow 22})22$$

(8.3 cont'd.)

Note: i) Extension is to ensure correct subtraction effects if p or N_r are negative.

ii) BN must satisfy the reservation check.

iii) BN will always be truncated to 22 bits.

iv) The 14 bit value in N_r is regarded as a signed integer.

(c) Replaced Branch (EBM and R = 1)

$BN = (n_{em} + (p)q_{\rightarrow 22})22$ where n_{em} is the 22 bit content of address N_r.

Note: i) N_r and BN must satisfy the reservation check.

ii) BN will always be truncated to 22 bits.

.9 For unstructions particularly affected by operating modes (023(OBEY), 060(BUX), 062(BDX), 064(BCHX), 066(BCT), 070(CALL), 072(EXIT), 117(SMO)) see Part 1.

.10 Common Mode Programming.

Common mode programs are programs which will operate in any combination of 15AM, 22AM, DBM and EBM. When writing such a program, the following should be taken into consideration:-

.1 Length of address fields: a whole word should be devoted to the address of any location in the Upper storage area, or if the occasion arises, in the program area. (The most significant 2 bits are not in fact referred to except in character addressing operations, but bits 2 to 8 must always be regarded as part of the word address). The commonest cases are:-

a) Modifiers, both in pre-modification and indexing.

b) Parameters which convey addresses of operands or areas.

c) Addresses set up in accumulators for 126(MOVE) and similar operations.

(.10.1 cont'd.)

Note: In connection with modification, the pre-modify order 117(SMO) is not available throughout the range. A routine intended to operate on all machines must therefore avoid the use of 117, together with 116(MVCH) and 066(BCT).

.2 Count instructions: incrementing word addresses can be carried out using 101(ADN) instructions. This avoids any difficulties which might arise because of differences in the effect of the 060(BUX) and 062(BDX) instructions in 15AM and 22AM.

064(BCHX) is used to increment character addresses, and should always branch to the next instruction. On exit from a loop the contents of bits 2 to 8 will depend on the actual mode in use. In a program which switches between 15AM and 22AM, a word which has been used by a 064 instruction should not be carried from a 15AM part of the program to a 22AM part.

066(BCT) is not available throughout the range, so in most cases of common mode programming a counting process should be performed by 103(SBN) and 052(BNZ).

.3 Subroutine links: the setting of the zero suppression mode is stored in Bit 8 of the link in 15AM + DBM and bit 1 of the link in 22AM or 15AM + EBM. It cannot therefore be used as a parameter to the subroutine (unless the subroutine first discovers the current mode, which is unlikely to be economical).

CONTENTS

Section 0	Instructions	000 - 007
1		010 - 017
2		020 - 027
3		030 - 037
4		040 - 047
5		050 - 056
6		060 - 066
7		070 - 074
10		100 - 107
11		110 - 113, 116 - 117
12		120 - 127
13		076, 114, 115, 130 - 137
14		140 - 147 are illegal
15		150 - 152, 154 - 157
16		160 - 165
17		170 - 177 are illegal

The instructions in this group operate on the contents of X and N, leaving the result in X. The previous contents of X are destroyed.

.1 Instructions 000 to 003:

V, but not C, is set if the result exceeds capacity.

Note: These instructions must be used with single length operand or with the most significant part of a multi-length operand.

(a) 000 (LDX)

$$x' = n + c$$

Write n + c into X.

(b) 001 (ADX)

$$x' = x + n + c$$

Add n + c to x.

(c) 002 (NGX)

$$x' = - n - c$$

Write n + c negatively into X.

(d) 003 (SBX)

$$x' = x - n - c$$

Subtract n + c from x.

.2 Instructions 004 to 007

C, but not V, is set if the result exceeds capacity: thus the instructions should be used with the less significant words of multi-length operands. The 004 and 005 instructions first perform the specified operations on the 24-bit signed operands and set carry if, in the result, B0=1. The 006 and 007 instructions first perform the specified operations on the operands treated as 24-bit positive numbers and set carry if in the result B(-1) = 1.

B0 is always cleared before storing.

(a) 004 (LDXC)

$$x' = n+c$$

Write $n+c$ into X.

(b) 005 (ADXC)

$$x' = x+n+c$$

Add $n+c$ to x.

(c) 006 (NGXC)

$$x' = -n-c$$

Write $n+c$ negatively into X.

(d) 007 (SBXC)

$$x' = x-n-c$$

Subtract $n+c$ from x.

The instructions in this group operate on the contents of X and N, leaving the result in N. The previous contents of N are destroyed.

.1 Instructions 010 to 013:

V, but not C, is set if the result exceeds capacity.

Note: These instructions must be used with single length operand or with the most significant part of a multi-length operand.

(a) 010 (STO)

$n' = x + c$

Write x + c into N.

(b) 011 (ADS)

$n' = n + x + c$

Add x + c to n.

(c) 012 (NGS)

$n' = - x - c$

Write x + c negatively into N.

(d) 013 (SBS)

$n' = n - x - c$

Subtract x + c from n.

Instructions 014 to 017

C, but not V, is set if the result exceeds capacity; thus the instructions should be used with the less significant words of multi-length operands. The 014 and 015 instructions first perform the specified operations on the 24-bit signed operands and set carry if in the result B0 = 1. The 016 and 017 instructions first perform the specified operations on the operands treated as 24-bit positive numbers and set carry if, in the result B(-1) = 1.

B0 is always cleared before storing.

(a) 014 (STOC)

$n' = x+c$

Write $x+c$ into N.

(b) 015 (ADSC)

$n' = n+x+c$

Add $x+c$ to n.

(c) 016 (NGSC)

$n' = -x-c$

Write $x+c$ negatively into N.

(d) 017 (SBSC)

$n' = n-x-c$

Subtract $x+c$ from n.

Except orders 023, 026 and 027, the instructions in this group operate on the contents of X and N, leaving the result in X. The previous contents of X are destroyed. None of the instructions sets V. Instructions 020 to 022, 024 and 025 cannot set C.

.1 Logical Operations

(a) 020 (ANDX)

$$x' = x \,\&\, n$$

"And" of x and n

This instruction produces 1's in bit-positions where both x and n have a 1. All other bits in x' will be zeros.

(b) 021 (ORX)

$$x' = x \vee n$$

"Inclusive or" of x and n

This instruction produces 1's in bit-positions where either x or n has a 1, or both have a 1. All other bits in x' will be zeros.

(c) 022 (ERX)

$$x' = x \neq n$$

"Exclusive or" of x and n.

This instruction produces 1's in bit-positions where x and n are different. All other bits in x' will be zeros.

.2 Miscellaneous Instructions.

(a)023 (OBEY)

1. This instruction causes the machine to perform the instruction at the location specified in N. This may itself be a 023 instruction. The ultimate instruction, not an 023, which is obeyed as a result of one or more 023 instructions is known as the operand instruction. If the operand instruction is an effective branch, it will be executed in the normal way and the original sequence will be forgotten; otherwise control is returned after execution of the operand instruction to the instruction following the first 023 obeyed.

2. If the operand instruction is a 070(CALL) the link address stored is the one following the 023 (the first if there is a chain).

3. Premodification of a 023 is used in deriving the address held in N, and not in modifying the operand instruction.

4. The sequences SMO, OBEY ⟶ SMO; and OBEY ⟶ SMO, SMO are illegal.

5. If the operand instruction is a relative branch, the instruction address used in determining the branch address is the location of the 023, and not the location of the branch.

6. The 023 itself does not set either C or V, but the operand instruction may do so. If C is set immediately before obeying the 023, then it will remain set for the instruction in N.

(b)024 (LDCH)

$x' = n_j$

Write into x_3 the character n_j ($j = 0, 1, 2, 3$) and clear the rest of X. C cannot be set. In normal use, N will be character modified. If there is no modification, $j = 3$.

See Section 12.2(b) and Section 6.3

(c)025 (LDEX)

$x' = n_e$

Write into x_e the least significant 9 bits of n and clear the remainder of X. C cannot be set.

(d) 026 (TXU)

Set C if n ≠ x or C = 1

This instruction cannot clear C.

Using 026 instructions consecutively, p times, two items each contained in p words, can be compared. If any pair of corresponding words are different C will be set. The rest of the 026 instructions will leave C set. x and n are unaltered. After the p th 026 instruction the state of C may be tested by an 074/5 or 074/6 instruction. See 7.2(f) and 7.2(g).

(e) 027 (TXL)

Set C if n + c > x otherwise clear C. AL/15

This instruction compares the contents of N and X and sets C if n + c > x. If n + c < x, C is cleared. If n = x, C is unaltered. AL/15

While comparing two words each is considered as an unsigned 24-bit number. This instruction is primarily meant for words of alpha-numeric characters. If it is used with binary numbers, care should be exercised as B0 is not interpreted as sign bit.

It may be used to compare multi-length character strings in a similar way to 026.

Note: In both 026 and 027, when comparing multi-length items, comparison must commence at the least significant end and progress continuously towards the most significant end.

The instructions in this group operate on the contents of X and N, leaving the result in N. The previous contents of N are destroyed. Neither V nor C can be set.

.1 Logical Operations on Store

(a) 030 (ANDS)

$n' = n \,\&\, x$

"And" of n and x. See 2.1(a) for effect of "and".

(b) 031 (ORS)

$n' = n \vee x$

"Inclusive or" of n and x. See 2.1(b) for effect of "inclusive or".

(c) 032 (ERS)

$n' = n \not\equiv x$

"Exclusive or" of n and x. See 2.1(c) for effect of "exclusive or".

.2 Miscellaneous Instructions [9]

(a) 033 (STOZ)

$n' = 0$

Clear N. The value of X should be zero.

(b) 034 (DCH)

$n_j = x_3$

Write x_3 into jth character position of N(j = 0,1,2,3) leaving the rest of N unchanged. Normally N will be character modified. However, if there is no modification, j = 3.

See 6.3 for stepping of the character modifier.

(c) 035 (DEX)

$n'_e = x_e$

Write x_e into the least significant 9-bits of N, leaving the rest of N unchanged.

(d) 036 (DSA)

$n'_a = x_a$

Write x_a into the least significant 12-bits of N, leaving the rest of N unchanged.

(e) 037 (DLA)

$n'_m = x_m$

Write x_m into the least significant 15-bits of N, leaving the rest of N unchanged.

The instructions in this group operate the contents of X, X + 1 and N, usually leaving the result in X and X + 1. (The 047 instruction also writes into N). These instructions are performed by extracode on 1901, 1902 and 1903.

.1 Instructions 040 to 042:

Multiplication. The operands may be regarded as either integers or fractions. If they are regarded as mixed numbers, the number of bits after the binary point in the product is equal to the sum of the number of bits after the binary points in the factors. The sign bit of the less significant half of the product is always zero in the result. Overflow will occur only if the multiplier and the multiplicand both consist of 1 in the sign bit with all other bits 0. The double length product in this case will consist of 1 in the sign bit with all other bits 0, and the result will overflow except possibly with 042 (q.v.).

(a) 040 (MPY)

$$x:' = n.x$$

Unrounded multiplication. The single length quantities in N and X are multiplied to give the double length unrounded product in X and X + 1.

(b) 041 (MPR)

$$x:' = n.x + 2^{-24}$$

Rounded multiplication. The single length quantities in N and X are multiplied to give a double length unrounded product n.x. 1 is now added into bit 1 of the less significant half; if this bit is 1 the resulting carry is added to the more significant half. Thus if n and x are looked upon as proper fractions, the single length rounded product would be obtained by adding 2^{-24}, giving a rounded single length product in X; x* being "spoiled". This in effect amounts to adding $\frac{1}{2}$ to the contents of X + 1.

(c) 042 (MPA)

$$x:' = n.x + x^*$$

Semi-cumulative multiplication. The single length quantities in N and X are multiplied and the signed quantity in X + 1 is added to the less significant word of the product giving a double length product. If there is a carry while adding x*, it is communicated to the more significant half to be added. The result is available in X and X + 1.
Overflow will not be set if x = -1.0, x* is negative and n = -1.0.

.2 043 (CDB)

$$x:' = 10.x: + n_j$$

Decimal to binary conversion. The character n_j is checked to determine whether or not it is numeric. If it is, then the double length quantity in X and X + 1 is multiplied by 10 and the character added to the less significant half of it. Overflow will be set if x:' exceeds capacity.

If BO of either x or x* is 1, the result is undefined, except that x*' is unaffected by x. This means that if the result in X + 1 only is of interest, it is not necessary to ensure that x is initially positive. Where x is initially negative, the result in X and whether overflow is set remain undefined. BO in both X and X + 1 is always left clear. If n_j is non-numeric then x:' = x: and C is set. N will normally be character-modified to provide n_j, but if there is no modification then j = 3. This instruction is undefined if X=N(M) or X+1 = N(M). AL/15

.3 Instructions 044 to 046

Division. Except in order 046, the dividend has to be double length and the divisor single length. A single length quotient is formed in the less significant accumulator and the remainder in the more significant one. Except when overflow occurs, the remainder is always numerically smaller than the divisor. An extension to the length of

.3 (continued)

the quotient might be obtained by treating the remainder as quotient and continuing division by the original divisor. The remainder r is computed by the formula r = x - nx*'. The operands might be regarded as integers, fractions or mixed numbers. If they are looked upon as mixed numbers, the number of bits in the quotient after the binary point is equal to the difference between the number of bits after the binary point in the dividend and in the divisor. Overflow is set if division by zero is attempted. Overflow will also be set if the result exceeds the capacity of a word. If B0 of X+1 is not zero, the result is undefined for function 044 and 045.

(a) 044 (DVD)

$x^{*\prime}=x:/n$, x' = remainder

Unrounded double length division. The process of division is terminated when the quotient is a single length word. The quotient is then written into X + 1 and the remainder into X. The quotient is always less than the true quotient unless the remainder is zero. The remainder always has the same sign as the divisor: hence it is possible to extend the length of the quotient by continuing division by the original divisor with the remainder as dividend.

(b) 045 (DVR)

$x^{*\prime}=x:/n + 2^{-24}$, x' = remainder

Rounded double length division. The division takes place in the same fashion as in order 044 except for the fact that it is terminated when the quotient is one bit longer than a single word, and a 1 is added to this position producing a rounded quotient, which is written into X + 1. The remainder, which need not have the same sign as the divisor, but which will satisfy the equation, is written into X.

.3 (continued)

(c) 046 (DVS)

$$x^{*\prime} = x^*/n,\ x' = \text{remainder}$$

Single length unrounded integer division. This order is similar to 044, except that the dividend, in X + 1, is single length; its arithmetic significance corresponds to that of x* in 044. The dividend may be of either sign. The same rules as for 044 apply to the quotient and remainder. The function can also be used if the operands are fractions.

.4

(a) 047 (CBD)

$$x:' = 10.x:\ ,\ n_j' = \text{character}$$

Binary to decimal conversion. The double length number in X and X + 1 is multiplied by 10. The bits which would overflow at the more significant end as a result of multiplication form a 6-bit character in n_j, leaving the rest of N unchanged. If x: is regarded as a double length fraction, n_j becomes the integral part of the product. The value 'zero' will be converted into 'space' if the Zero Suppression Mode is set. (See Part 1, Section 12.2 (c)). The Zero Suppression Mode will be unset by this instruction when any character other than zero is produced.
Overflow cannot be set. This instruction is undefined if x=N(M) or x+1=N(M). AL15
If BO of either x or x* is 1, the result is undefined.
N will normally be character-modified to determine the value of j, but if there is no modifier then j = 3.

These instructions branch on the state of an accumulator.

They cannot set V and do not alter the contents of X.

N is 15-bit and cannot be modified.

.1 050 (BZE)

Branch to N if x = 0.

.2 052 (BNZ)

Branch to N if $x \neq 0$.

.3 054 (BPZ)

Branch to N if $x \geq 0$.

.4 056 (BNG)

Branch to N if $x < 0$.

.1 060 (BUX)

15AM $x'_m = x_m + 1$; $x'_c = x_c - 1$; branch if $x'_c \neq 0$ to BN.

In 15AM this instruction adds 1 to the modifier, subtracts 1 from the count, and branches to BN if the new count is non-zero. There is no carry from x_m to x_c. If x_c is initially zero, it is treated as being equal to 512.

22AM $x'_{em} = x_{em} + 1$; $x'_k = x_k$; branch to BN.

In 22AM this instruction adds 1 to the address in X, and branches unconditionally to BN.

Note: for the derivation of BN (branch address) see Part 0, Section 9.8.3.

.2 062 (BDX)

This instruction is the same as the 060, the only difference being that in 15AM the modifier is stepped up by 2 instead of 1, and in 22AM 2 is added to the address in X.

15AM $x'_m = x_m + 2$; $x'_c = x_c - 1$; branch if $x'_c \neq 0$ to BN.

22AM $x'_{em} = x_{em} + 2$; $x'_k = x_k$; branch to BN.

Note: for the derivation of BN (branch address) see Part 0, Section 9.8.3.

.3 064 (BCHX)

15AM $x_k = 0$, 1 or 2 $x'_k = x_k + 1$; $x'_m = x_m$; $x'_d = x_d - 1$.

$x_k = 3$ $x'_k = 0$; $x'_m = x_m + 1$; $x'_d = x_d - 1$.

Then if $x'_d \neq 0$ branch to the instruction in BN; if $x'_d = 0$ proceed to the next instruction in sequence.

If the index register is thought of as circular, x_k forms the least significant two bits of the 17-bit character modifier $x_m x_k$. The

(064 cont'd.)

maximum value of the count is 128 when $x_d = 0$. The instruction causes 1 to be added to the character modifier $x_m x_k$, 1 to be subtracted from the count x_d, and the instruction branches to BN if x'_d is non-zero.

22AM $x_k = 0$, 1 or 2 $x'_k = x_k + 1$; $x'_{em} = x_{em}$; branch to BN.

$x_k = 3$ $x'_k = 0$; $x'_{em} = x_{em} + 1$; branch to BN.

As there is no count in 22AM, x_k can be thought of as forming the least significant two bits of the 24-bit character modifier $x_{em} x_k$.

Note: for the derivation of BN (branch address) see Part 0, Section 9.8.3.

.4 066 (BCT) (For availability see Part 0, Section 7).

15AM $x'_m = x_m - 1$; $x'_c = x_c$. Then if $x'_m \neq 0$, branch to BN; if $x'_m = 0$ proceed to the next instruction in sequence.

This instruction subtracts one from the least significant 15 bits of X and branches to BN is x'_m is non-zero. The contents of x_c remain unchanged.

22AM $x'_{em} = x_{em} - 1$; $x'_k = x_k$. Then if $x'_{em} \neq 0$ branch to the instruction in BN, if $x'_{em} = 0$ proceed to the next instruction in sequence.

Note: for the derivation of BN (branch address) see Part 0, Section 9.8.3.

N is 15-bit (22-bit in Extended Mode) and cannot be modified.

.1(a)070 (CALL)

1. This instruction stores a link in the accumulator specified, clears V, and branches to the location N.

   C will be cleared.

2. The format of the link is as follows:-

   15AM and DBM — Bit 0 contains the setting of V.
   Bit 8 contains the setting of the zero suppression mode.
   Bits 9 to 23 contain the instruction address following the 070 (i.e. 070 instruction address + 1).

   15AM and EBM; — Bit 0 contains the setting of V.

   22AM — Bit 1 contains the setting of the zero suppression mode.
   Bits 2 to 23 contain the instruction address following the 070 (i.e. 070 instruction address + 1).

3. After obeying the instruction, the contents of V = 0; and the contents of the zero suppression mode is undefined.

4. If the 070 is the operand instruction of an 023(OBEY), the address used in forming the link is the address of the 023 (the first obeyed if there is a chain) and not the 070.

(b)072 (EXIT)

1. This instruction specifies an accumulator and a 15 bit value in N. It causes re-entry to the calling routine after a sub-routine has been performed.

2. The contents of X are assumed to be a link, as described in .1 (a) 2 above. N specifies the value to be added to the link address in X to

(072 cont'd.)

give the re-entry address.

3. The instruction sets V if it is already set, of if the value stored in the link is 1; zero-suppression is set to the value stored in the link (but see 5 below). That is:-

If V = 1 or if
Bit 0 of X = 1, V after the instruction has been obeyed = 1; otherwise V after the instruction has been obeyed = 0.

If 15AM + DBM, the contents of the zero-suppression mode after the instruction has been obeyed = Bit 8 of X; otherwise the zero-suppression mode after the instruction has been obeyed = Bit 1 of X.

4. A branch is then performed to the return instruction defined as follows:-

15AM + DBM $RI = (N_m + (p)q + x_m)q$ $(q = 15)$

15AM + EBM, 22AM $RI = (N_m + p_{\rightarrow 22} + x_{em})q$ $(q = 22)$ (but see Note iv below)

Note: i) p = premodifying value specified by preceding 117(SM0) instruction; 0 if none.

ii) q = the number of bits to which an address (inside parenthesis) is truncated.

iii) $\rightarrow$ 22 implies extension to 22 bits by propagating the most significant bit of the value concerned.

iv) If the combination 15AM + EBM is used, the formula holds true only if the processor is capable of operating in 22AM.

5. The components n_m, x_m, x_{em} and p may be negative; the sum will be

(072 cont'd.)

cyclic at 15 or 22 bits as indicated in 4 above. In most cases there is no carry effect beyond the cyclic limit; however, on some processors using 15AM + DBM a carry will be propagated and will reverse the setting of the zero-suppression.

6. 072 always clears C.

(c) The Link Word.

The form of the link stored in X by an 070 order is such that it may be used to discover the state of V before an 070 order and to control its state after an 072 order, and as a word address modifier by a sub-routine for purposes such as obtaining data and other parameters. No assumptions may be made about the state of any bits, other than the sign and the 15 address bits (22 bits in extended mode), nor may they be altered in any way.

Alteration of a link address: if it is desired to alter a link address, it should normally be done by using the N field of the 072 order, but, where this is impractical, care must be taken that only the least significant 15 bits of the link (22 bits in extended mode) are altered. Replacement by the use of logical, rather than by arithmetical functions is the advisable method.

Generation of pseudo-links: there are various possible methods open to the programmer, the choice being determined by consideration of the possible effects of V and/or the Zero-Suppression mode. If the state of th[illegible] no importance then the desired address can be placed in the appropriate bits of any accumulator, and then used as the link. The state of V can be controlled by the state of Bit 0. If, however, it is essential to preserve the state of the Zero-Suppression mode, an 072 order, with N being the address of the following instruction should be used, followed by replacement of the address and/or V part of the link.

.2 Instructions 074/0 to 074/7.

These instructions, except 074/0, branch on the state of the overflow (V) or carry (C) register. The three-bit accumulator field of the branch instruction, 0 to 7, is used to determine the desired variation of the 074 instruction group.

(a)074/0 (BRN)

Unconditional branch.

(b)074/1 (BVS)

Branch if V is set.

(c)074/2 (BVSR)

Branch if V is set and clear V.

(d)074/3 (BVC)

Branch if V is clear.

(e)074/4 (BVCR)

Branch if V is clear, otherwise clear V.

(f)074/5 (BCS)

Branch if C is set, clear C.

(g)074/6 (BCC)

Branch if C is clear.

(h)074/7 (BVCI)

Branch if V is clear and set V, otherwise clear V.

.3 076 (BFP)

Test floating point accumulator.
See Floating Point (Section 13)

The instructions in this group operate on the contents of X, leaving the result in X. The previous contents of X are destroyed. These instructions closely resemble those of Group 00, except that the quantity specified in the N address, instead of being taken as the address of an operand, is used as the operand itself.

Since the N address occupies 12 bits, the value of N must be within the range $0 \leq N \leq 4095$. This range may be extended by modification to 15 bits (or 22 bits in extended mode on 1906 or 1907).

.1 Instructions 100 to 103:

V, but not C is set if the result exceed capacity.

Note: The instructions should be used with single-length operands, or with the most significant words of multi-length operands.

(a) 100 (LDN)

$x' = N + c$

Write N + c into X.

(b) 101 (ADN)

$x' = x + N + c$

Add N + c to x.

(c) 102 (NGN)

$x' = - N - c$

Write N + c negatively into X.

(d) 103 (SBN)

$x' = x - N - c$

Subtract N + c from x.

.2 Instructions 104 to 107

C, but not V, is set if the result exceeds capacity. The instruction should be used with the less significant words of multi-length operands, though this process will have very restricted use compared with ghe equivalent process in groups 00 and 01. The 104 and 105 instructions first perform the operations on the 24-bit signed operands and set carry if, in the result B0 = 1. The ~~006~~ 106 and ~~007~~ 107 instructions first perform the specified operations on the operands treated as 24-bit positive numbers and set carry, if, in the result, B(-1) = 1. [9]

B0 is always cleared before storing.

(a) 104 (LDNC)

$x' = N+c$

Write N+c into X.

(b) 105 (ADNC)

$x' = x+N+c$

Add N+c to x.

(c) 106 (NGNC)

$x' = -N-c$

Write N+c negatively into X.

(d) 107 (SBNC)

$x' = x-N-c$

Subtract N+c from x.

.1 Shift Functions.

The functions in this group operate on the contents of X (single-length) or X and X + 1 (double-length) moving them N_s places to the left or right. For the purposes of this group, the N address of the instructions may be regarded as being sub-divided into the two most significant bits (N_t), and the 10 least significant bits (N_s). The instructions may be modified, provided that the state of the bits N_t is not affected. There are four types of shift, each of which can operate either to the left or to the right. The type of shift is determined by the value of N_t (0, 1, 2 or 3):

0 Cyclic shift.

1 Logical shift.

2 Arithmetic shift.

3 Special shift.

The maximum permissible number of places of cyclic or logical shift is 24 places for 110 and 112 (Single-length) orders, and 48 places for 111and 113 (Double-length) orders. For arithmetic and special shift the maximum is 24 and 47 places for single- and double-length respectively. Orders specifying more places of shift are not necessarily notified as illegal, but may give rise to unpredictable results. A shift of zero places will always leave the operand unaltered.

Double-length shifts are performed by extracode on the 1901, 1902 and 1903, and by hardware on all other machines of the series.

.2 Types of Shift:

(a) Cyclic Shift ($N_t = 0$)

The single or double-length word is considered as a 24- or 48-bit pattern, rather than a numerical quantity. With each place of shift, a bit is lost at one end, and is replaced in the space created at the other end. Neither V nor C can be set.

(b) Logical Shift ($N_t = 1$)

This is similar to the cyclic shift, the difference being that spaces created by the shift are left as zeros. Neither V nor C can be set.

(c) Arithmetic Shift ($N_t = 2$)

The operand is treated as a signed number, of either 24 or 47 bits, including sign bits. In the latter case, the sign bit of the less significant word should be zero, and is stepped over: C cannot be set, but the state of V may be changed by the shift. The action varies with the different functions, and will be described for each function.

(d) Special Shift ($N_t = 3$)

The action of this shift is dependent on the function, and will be described for each function.

.3 Shift Instructions:

(a) 110 ($N_t = 0$: SLC)
($N_t = 1$: SLL)
($N_t = 2,3$:SLA)

x' = x shifted left N_s places.

The cyclic and logical shifts are as described above. The arithmetic shift is similar to the logical shift, the only difference

.3(a) (Contd.)

being that V is set as soon as capacity is exceeded, i.e. as soon as the sign bit is changed. The special shift is identical to the arithmetic shift.

(b) 111 (N_t = 0: SLC)
(N_t = 1: SLL)
(N_t = 2,3:SLA)

x:' = x: shifted left N_s places.

The cyclic and logical shifts are as described above.

The arithmetic shift takes the 47-bit signed number and performs a shift setting V as soon as capacity is exceeded, i.e. as soon as the sign bit is changed. The sign bit of X + 1 is stepped over, (and, unless N_s = 0, zeroised), i.e. B1 of X + 1 when shifted one place to the left, becomes B23 of X.

The special shift is the same as the arithmetic shift.

(c) 112 (N_t = 0: SRC)
(N_t = 1: SRL)
(N_t = 2: SRA)
(N_t = 3: SRAV)

x' = x shifted right N_s places.

The cyclic and logical shifts are as described above.

The arithmetic shift takes the 24-bit signed number, and performs a shift, propagating a copy of the sign bit each time the word is shifted one place. The result is rounded by adding 1 to the least significant bit if the last bit lost was a 1.

The effect of the special shift is the same as that of the arithmetic shift if V is clear. However, if V is set, the action is to clear V, and perform a shift similar to the arithmetic shift, the only difference being that the inverse of the original sign bit is propagated.

If N_t=3, but N_s=0, V will be unchanged.

.3(c) (Contd.)

If X contains all ones, and is shifted right one place, with V set, rounding will cause overflow, but V will not be set. The result will be a clear word, except for BO = 1.

(d) 113 (N_t = 0: SRC)
(N_t = 1: SRL)
(N_t = 2: SRA)
(N_t = 3: SRAV)

x:' = x: shifted right Ns places.

The cyclic and logical shifts are as described above.

The arithmetic shift takes the 47-bit signed number, and performs a shift, propagating a copy of the sign bit each time the number is shifted one place. The result is not rounded. BO of X +1 will be zeroised, except when Ns = 0.

The special shift is the same as the arithmetic shift if V is clear. However, if V is set, the action is to clear V, and as shifting takes place, to propagate the inverse of the original sign bit. There is no rounding of the result. If $N_t = 3$, but $N_s = 0$, V will be unchanged [11]

.4 Instructions 114 and 115 (NORM):

See section 13 (Floating Point)

.5 Miscellaneous Instructions:

(a) 116 (MVCH) ~~(Available on 1906 and 1907 only).~~ For availability see Part 0 section 7

Block transfer (characters)

This instruction copies the character whose address is specified in X, into the character position whose address is specified in X + 1. 1 is then added to the character-modifier field, BO and B1, of both X and X + 1, with carry into the word-modifier field occurring automatically, and the procedure repeated until N ($\leq$511) characters have been transferred.

Group 11 (Contd.)

.5(a) (Contd.)

If N = 0, 512 characters will be transferred. This is the maximum number, and cannot be exceeded by modification.

The transfer is progressive, being carried out one character at a time. Consequently, if the beginning of the destination area overlaps the end of the source area, then the original information in the overlapping area will be overwritten before it can be transferred.

Overwriting does not occur if

$X^* > X + N$, or if $X^* \leq X$.

Word 8 must not be part of the destination field. Otherwise the program may be corrupted.

(b) 117 (SMO) ~~(Available on 1906 and 1907 only)~~

Supplementary modifier to next instruction.

This instruction causes the least significant 15 bits (compact mode) or 22 bits (extended mode) of the word whose address is specified by N(M) to be used as a modifier in the next instruction in addition to any modification already specified by that instruction.

Instruction 117 is illegal if it immediately follows another 117 instruction. This applies if either or both are operand instructions of an 023(OBEY), q.v.

DELETED AL/16

(.5 cont'd.)

(b)117 (SMO) (For availability see Part 0, Section 7).

Supplementary modifier to next instruction.

1. This instruction causes the least significant 15 bits (15AM) or 22 bits (22AM) of the word whose address is specified by N(M) to be used as a modifier in the next instruction. A flag is also set to indicate the presence of a premodifier.

2. Either 15 or 22 bits of the premodifying value are used in deriving the operand of the next instruction. Bits 0 and 1 have no effect in 15AM or 22AM. The flag is cleared.

3. The contents of C are unchanged.

4. The 117 can be used as a modifier in the next instruction in addition to any modification already specified in that instruction.

5. In EBM, bit 9 (R) distinguishes relative and replaced branches. This bit is regarded as part of the function, and it is not possible to affect its value by premodification.

6. The effect of two consecutive 117 instructions are undefined, either if they appear explicitly or if one or both are the operand instructions of 023(OBEY) instructions. Therefore, the following sequences are invalid:-

   117, 117, XXX
   117, 023 ⟶ 117, XXX
   023 ⟶ 117, 117, XXX
   023 ⟶ 117, 023 ⟶ 117, XXX

   The 023 in these sequences may be a single instruction or a chain.

7. A 117 preceding a 123 (NULL) operates on the 123 (with null effect), and not on the instruction following the 123.

Group 12

This is a mixed group of instructions in which neither V nor C can be set.

.1 Logical functions with Literal Operand

These instructions operate on the contents of X, leaving the result in X. They closely resemble instructions 020 to 022, the difference being that N is taken, not as the address of an operand, but as the operand itself. The value of N must be within the range $0 \leq N \leq 4095$. This range can be extended by modification. As in other logical functions, the word is treated as a bit-pattern rather than a numerical quantity.

(a) 120 (ANDN)

$x' = x \,\&\, N$

"And" of x and N.

This instruction produces 1's in bit-positions where both x and N have a 1. All other bits in x' will be zeros.

(b) 121 (ORN)

$x' = x \vee N$

"Inclusive Or" of x and N.

This instruction produces 1's in bit positions where either x or N has a 1, or both have a 1. All other bits in x' will be zeros.

(c) 122 (ERN)

$x' = x \not\equiv N$

"Exclusive Or" of x and N.

This instruction produces 1's in bit-positions where x and N are different. All other bits in x' will be zeros.

2 Miscellaneous Instructions

(a) 123 (NULL)

No operation

This may be used as a dummy instruction, and is useful during program development for leaving space to be filled later.

This order does not clear 'C'.

(b) 124 (LDCT)

$x'_c = N$, $x'_m = 0$.

This sets up a counter-modifier for use with loops of program containing 06 group branches (see Section 6). N may be modified, and if the resulting total exceeds 511, bits representing 512 and upwards in N (M) are ignored. If N = 0, a count of 0 is entered, but 060 (BUX) and 062 (BDX) will treat this as a count of 512.

(c) 125 (MODE)

Set, or unset 'zero-suppression' Mode, in accordance with the state of N (M).

The value of N (M) controls the state of the 'zero-suppression' mode. If N (M) = 1, the mode will be turned 'On', if N (M) = 0, the mode will be turned 'Off'. The effect of other values of N (M) is not defined. Care must be taken in the use of this function, as the use of function 072 can affect the mode-setting (see definition of Instruction 072).

The mode can also be unset by Instruction 047

(see Instruction 047)

.2 (Contd.)

(d) 126 (MOVE)

Block transfer

Transfer N ($\leq$511) words in the location starting at the address specified in X, to the N ($\leq$511) locations starting at the address specified in X + 1. If N = 0 then the number of words transferred will be 512. This is the maximum number possible, and cannot be increased by modification.

The transfer is progressive, being carried out one word at a time. Consequently if the beginning of the destination area overlaps the end of the source area, then the original information in the overlapping area will be overwritten before it can be transferred.

Overwriting does not occur if

$X^* > X + N$

or if $X^* \leq X$.

X and X + 1 are unchanged, unless they are included in the area being overwritten.

Word 8 must not be part of the destination field. Otherwise, the program may be corrupted.

This instruction is performed by extracode on the 1901, 1902 and 1903, and by hardware on all other machines of the series.

.2 (Contd.)

(e) 127 (SUM)

Form check-sum

Write into X the sum of N ($\leq$ 511) consecutive words starting with the word whose location is specified in X + 1. Bits are lost at the most significant end of the sum if it exceeds capacity, but V cannot be set. The maximum number of words which can be summed is 512. This will occur when N = 0.

This instruction is performed by extracode on the 1901, 1902 and 1903, and by hardware on all other machines of the series.

Floating-Point Instructions

.1 Representation of Floating-point numbers

In floating-point operations the operands are held in the normalised form $y = a.\ 2^e$, where a is a signed fractional argument limited within the ranges:

$1 > a \geqslant \frac{1}{2}$ or $-\frac{1}{2} > a \geqslant -1$, and e is a signed integral exponent having the range $-256 \leqslant e \leqslant 255$, giving the machine an approximate decimal range of $10^{-76} < y < 10^{76}$

All floating-point numbers are stored in double-length form. The argument is held in 38 bits (B0 to B23 of the first word and B1 to B14 of the second word), giving accuracy to within $\pm 2^{-37}$. B0 of the first word is the sign bit of the argument, and carry is propagated between the less significant and more significant words of the argument, ignoring B0 of the second word, which is used to indicate exponent overflow (see 2a.). A single-length argument (held in B0 to B23 of the first word only) can also be used, but the 38-bit argument is standard, and is assumed by the floating-point instructions.

The exponent is held in B15 to B23 of the second word in the form e+256, i.e. as a positive number in the range 0 to 511.

.2 The floating-point accumulator

Floating-point operations 130 - 137 are carried out in a special floating-point accumulator, A. Upon first activation of a program, the contents of A are indeterminate.

(a) Exponent Overflow

If, during any operation in A, the exponent attempts to exceed 255, exponent overflow is said to occur, and B0 of the second word is set to 1 (the state of the other bits being indeterminate). B0 is also set to 1 if it was originally 1 or if B0 of N+1 was 1. This can only be tested directly on the 1906 and 1907 (see instruction 076) and V does not get set at this stage.

Exponent overflow is cumulative in A, i.e. once set it remains set until A is reloaded or cleared (see functions 130,136,137); in functions 132-135 it is always left set if set in A or if set in the number in store (by that number having B0 of its least significant half set to 1).

However, if this bit is set, and the contents of A are transferred to store (instruction 137) or converted to mid-point form and stored (instruction 131) V will be set. In the case of instruction 137, B0 of the second word of the result will also be set to 1.

If a floating-point number with B0 of the second word set to 1 is loaded into A (instruction 136 and X = 0) this bit will also be set in A.

(b) Underflow

When e attempts to become less than -256 the value of the floating-point number is regarded as zero, and the argument and exponent are set to zero. This may also occur when e = -256. Floating-point zero thus has the same representation as fixed-point zero and is detectable by the same "branch-on-zero" instructions, but note: if underflow occurs during a sequence of operations in which exponent overflow has previously occurred, B0 of the second word remains set, and exponent overflow therefore continues to be indicated.

.3 Floating-point instructions

Note: in all floating-point instructions floating-point operands must be in normalised form, with one exception. A 2-word field passed into and out of the floating-point accumulator, by a 136 followed by a 137 instruction without intervening floating-point instructions, may be in any form; the full 48 bits will be copied unaltered. Note however that overflow is set by the 137 order if B0 of the second word is 1.

The result obtained from any floating-point instruction may differ from one processor to another, but will never deviate arithmetically by more than the value equivalent to 1 in the least significant bit of the argument from the result of infinite precision obtained from the same input argument or arguments.

(a) 114 (NORM)

x' = x normalised with respect to N(M)

$x^{*'}_{e} = N_{e}'$

Normalise the number whose single-length argument is held in X with respect to N(M) (= exponent +256).

For details see below.

13.3(b) (cont.)

(b) 115 (NORM)

x:' = x: normalised with respect to N(M)

$x^{*}_{e}' = N_{e}'$

Normalise the number whose double-length argument is held in X: with respect to N(M) (=exponent +256).

For details see below.

Normalisation

The instructions 114, 115 provide the ability to convert a number to floating-point form with respect to an exponent provided by the object program.

The instruction 114 operates on a single-length input argument and confines the result argument to single-length (confining the result will lose accuracy only if V is set initially). 115 operates on a double-length (47-bit) input argument and produces a full 38-bit argument (rounded). Otherwise the instructions operate in the same way, and in both cases if the inputs are taken as fractional arguments with an exponent of N(M), a floating-point number of the standard form with correct exponent is left in X:. However, if exponent overflow occurs, V is set and x': is left undefined.

The effect of the instruction is equivalent to the following: If N(M) <0 it is assumed that floating-point underflow has occurred. Hence, X: and V are cleared and the instruction terminates.

The condition for N(M) to be taken as zero is that B14 to B23 of N(M) are zero, and the condition for N(M) to be taken as negative is that B12 of N(M) is 1.

If N(M) >0 'exponent count' is set equal to B14 to B23 of N(M) and, for 114 only, X* is cleared. Succeeding actions depend upon the state of V and x; as follows:-

(A) If V is clear and x: is zero, X: is cleared, V is left clear and the instruction terminates. Hence, the only effect in this case is to clear B0 of x*if set.

13.3(b) Cont.)

(B) ~~(b)~~ If V is clear and x: is non-zero, x: is shifted left arithmetically 0 or more places. The exponent count is reduced by 1 for each place shifted. The shift terminates when either of the following conditions occurs:-

I The sign of x is different from the adjacent bit.

II The exponent count is reduced to zero.

If condition II is reached, exponent underflow is deemed to have occurred, even if condition I is reached simultaneously.
X: and V are cleared and the instruction is terminated.

If condition I, but not condition II, is reached and the top bit of the 10-bit exponent count is 1, exponent overflow has occurred, The instruction terminates with V set and x: undefined.

If condition I, but not condition II, is reached and the top bit of the 10-bit exponent count is 0, the actions are as follows:

i. (115 only) x: is rounded to produce a 38-bit argument, i.e. B0 to B23 of x and B1 to B14 of x*. B0 of x* is left clear.

ii. The 9 least significant bits of the exponent count replace the 9 least significant bits of x*.

iii. The instruction terminates with V clear.

(C) If V is set and the top bit of the 10-bit exponent count is 1, exponent overflow has occurred. The instruction terminates with V set and x: underfined.

If V is set and the top bit of the 10-bit exponent count is 0, it is assumed that X has overflowed by one bit. A special right shift of one place is performed on X:. The shift is single-length for 114 and double-length for 115: no rounding takes place at this stage. The exponent count is increased by 1. If the top bit of the 10-bit exponent count now becomes 1, exponent overflow has occurred and the instruction terminates with V set and x: underfined. Otherwise, actions (i) to (iii) of Section B take place.

13.3(c) (Cont).

c) 130 (FLOAT)

n: → a

Convert a mid-point number held in N(M) and N* (M), to normalised floating-point form and load it into the floating-point accumulator A. The instruction takes from N and N+1 a double-length number having an assumed binary point immediately to the right of B23 of N, normalises it with respect to an exponent of 279, (i.e. 256 + 23) and inserts the resulting exponent. The result is left in A. The contents of N and N+1 are left unchanged.

The results obtained may differ from one processor to another but will never deviate arithmetically by more than the value equivalent to 1 in the least significant bit of the argument from the result of infinite precision obtained from the same input argument.

d) 131 (FIX)

a → n:

Convert the floating-point number held in the floating-point accumulator A to mid-point form, and store it in N(M) and N* (M). The instruction takes from A a number whose exponent is stored as E.

.3(d)(cont.)

The exponent bits of the number are set to zero, and a double-length arithmetic shift (unrounded) is carried out upon it until either:

I. A shift of (279-E) places has occurred, a negative number of places indicating left shift and a positive number right shift.

II Sufficient places of shift, either left or right, have occurred, to cause the number to become zero.

The result is placed in N and N + 1, the sign bit of N + 1 being set to zero. V is set if overflow has occured during a left shift, or if exponent overflow is indicated in A. The contents of A remain unchanged.

(e) Instructions 132-137 (General)

In instructions 132-137 the X field is used solely as a function extension device, and has no address significance. In instructions 132-135, X controls the orders in the following manner:

X = 0: Result rounded and normalised

X = 4: Function inverted, result rounded and normalised.

The form of rounding of the argument is not defined, but the result will have a bias of less than about 2^{-48}

Note: Other values of X are illegal. Hardware floating-point units (See.4) may not detect illegal conditions, and incorrect answers may be given. Results obtained by extracode for 132 to 135 (See .5) may differ slightly from results obtained by hardware, but the difference will not exceed ± 1 in the least significant bit of the argument. Under most conditions, there will be no differences for addition, subtraction or multiplication.

[AL12]

.3 (cont.)

(f) 132 (FAD)

a' = a + n:

Add the floating point number in N(M) & N*(M) to the contents of the floating point accumulator A. [AL/11]

X is always zero. X may be 0 or 4; the results conform to the same definition in both cases, but need not necessarily be identical, see (e) above. [AL/13]

(g) 133 (FSB)

a' = a - n:

Subtract the floating-point number in N(M) & N*(M) from the floating-point accumulator A. [AL/14]

The instruction performs the above function if X = 0, or the inverted function a' = n: - a if X = 4

(h) 134 (FMPY)

a' = a.n:

Multiply the contents of the floating-point accumulator A, by the floating-point number in N(M) & N*(M). [AL/14]

X is always zero. X may be 0 or 4; the results conform to the same definition in both cases, but need not necessarily be identical, see (e) above. [AL/13]

(i) 135 (FDVD)

a' = a/n:

Divide the contents of the floating-point accumulator, A, by the floating-point number in N(M) & N*(M). [AL/14]

The instruction performs the above function, if X = 0, or the inverted function a' = n : /a, if X = 4.

(j) 136 (LFP LFPZ)

a' = n:

Load the floating-point number in N(M) & N*(M) into the floating-point accumulator A. [AL/14]

.3 (j)(cont.)

X controls the order as follows:

X = 0 (LFP)

The floating-point number in N and N + 1 is loaded into A. The contents of N and N + 1 are left unchanged, and exponent overflow is set if BO of N + 1 is set.

X = 1 (LFPZ)

A is cleared, including exponent overflow.

(k) 137 (SFP,SFPZ)

n:' = a

Store the contents of the floating-point accumulator, A, The instruction is controlled by X as follows:

X = 0 (SFP)

Store the contents of A in N and N + 1, setting BO of N + 1 and setting V if exponent overflow is set, and leaving A unchanged.

X = 1 (SFPZ)

As defined above, but A is cleared after the operation.

.4 Instruction 076/0 to 076/5: (Available on 1906/7 only)

These orders branch on the state of floating point accumulator A (see 13.2). The 3-bit accumulator field of the jump instruction is used to determine the desired order. Whenever the contents of the accumulator field ranges from 0 to 5, one of the instructions from 076/0 to 076/5 is obeyed, subject to the state of A.

(a) 076/0( )

Branch if a = 0.

(b) 076/1( )

Branch if a $\neq$ 0.

(c) 076/2( )

Branch if a $\geq$ 0.

(d) 076/3( )

Branch if a < 0.

(e) 076/4( )

Branch if floating point overflow is clear.

(f) 076/5( )

Branch if floating point overflow is set.

(g) 076/6( )

Illegal.

(h) 076/7( )

Illegal.

Execution of Floating-point instructions.

.5 The availability and methods of execution of Floating-point instructions vary between the different machines of the 1900 range, and are set out in the table below.

Function	1901	1902 & 1903	1904	1905 & 1909	1906	1907
076	NA	NA	NA	NA	XX	H
114-115	Q	XX	H	H	H	H
130-131	Q	XX	XX	X	XX	H
132-135	Q	XX	XX	FH	XX	FH
136-137	Q	XX	XX	H	XX	H

NA = Not available

Q = Not available, but may appear at a later date under category XX.

X = Performed by extracodes.

XX = Performed by extracodes which are not part of basic Executive, but are an optional extra on larger store size configurations. On machines lacking these extracodes, the instructions are not available.

H = Performed by hardware.

FH = Performed by special autonomous floating-point hardware unit. When these instructions are encountered in a program, the operands are transferred to the F.P.U. and the program is free to continue while the F.P.U. carries out its operations. If any 130-7 or 076 order is encountered while the F.P.U. is busy, the machine cycles in a waiting loop until the unit is free.

Instructions 140 to 147 are illegal.

.1 Introduction

Group 15 instructions are Extracodes issued by the object program in order to initiate peripheral action. The general form of the 15 group instructions is:

F X N(M)

In 150-156, the specified peripheral is defined as:

a) if B 15 (value 256) of N is 0

the program's unit or file X of type N.

b) if B 15 (value 256) of N is 1

the program's unit or file Y, of type N - 256, where Y = \ s. 6 bits of acc.X;B 18, B 19 must both be 0.

(But see also Part 2 Section 0.3)

(This is sometimes known as the "replaced X" mode)

Notes

1. In b) above, the remaining bits of acc X must be zero, except as described under instruction 156.
2. In all group 15 instructions (except 156 and opening modes of 157) the program member which initiates the order will be suspended awaiting Operator Intervention if the peripheral unit specified has not been pre-assigned to it. On this occurrence a message will be printed on the console typewriter.
3. Peripheral type numbers are defined in Part 2 of this manual. The way in which a relationship is established between an absolute geographical device number, and the program's unit number (of appropriate type), is also defined in Part 2, Section 0.
4. Any order except 156 which addresses a unit or file for which a data transfer is still active, i.e. not yet completed (whether faulty or not) will cause suspension of the program. The transfer will be re-instated when the (earlier) transfer is successfully completed, except in the case of 150.

5. If a data transfer cannot be started immediately because the unit is attached to a multi-unit channel e.g. magnetic tape cluster, Executive may in some cases store the instruction, but not suspend the program. In such cases the stored instructions for the various units will be initiated by Executive as soon as possible.

.2 150-156 instructions

a) 150 (SUSBY)

Suspend program/sub-program if the specified peripheral unit is busy, removing the suspension when the unit is not busy.

b) 151 (REL)

Release the specified peripheral unit from assignation to this program making it available for assignation to another program.

Notes

1. Before releasing the unit, Executive checks that it is not busy. If the unit is busy, Executive suspends the program, removing the suspension and proceeding with the instruction when the peripheral is not busy.

2. This instruction is not applicable to 'fast' peripherals, and is treated as illegal if file storage or magnetic tape devices are specified.

3. The specified peripheral is disengaged.

4. With the 1901 handswitch executive this instruction acts in the same way as the 152 instruction.

A4/15

c) 152 (DIS)

Disengage the specified peripheral unit. That is, render the unit unable to execute a transfer until it is re-engaged by operator.

Notes

1. The unit is not released from assignation to the program.
2. This instruction is illegal if file storage or magnetic tape devices are specified, or flag setting devices. AL16
3. Before disengaging the unit, Executive checks that it is not busy. If the unit is busy, Executive suspends the program, removing the suspension and proceeding with the instruction when the peripheral is not busy.

d) 153

Illegal instruction.

e) 154 (CONT)

Suspend the program, and read in more program from the specified peripheral unit.

Notes

1. Illegal if: a) issued by other than member 0 of the program
   b) specified device is not of a suitable type (suitable types include Card Reader, Paper Tape Reader, and certain File Storage devices).
   c) specified unit not assigned to the program.
2. The additional program must be standard binary form (described in part 5), but it must not include a request slip block.
3. The input itself contains the addresses into which it is to be read, and is terminated when one of the authorized entry blocks is read. The actual areas overwritten may cover any, or all, of the program's area, except those locations (at the low address end of the program) which have restricted use (for full details of these restrictions see Part 0, Section 6 ~~1900 series Programming Notice No. 34, item 53~~). AL 7 Program loaded by a 154 instruction cannot extend beyond the program's limit.
4. This instruction does not check the state of any other peripheral units assigned to the program, and, if any of these are active at the time that a 154 instruction is issued, the result of that 154 instruction is undefined.
5. After completion of the 154 instruction the program will be resumed in accordance with the 'entry block' read.
6. Errors detected during the progress of the 154 instruction (except the illegal conditions mentioned above) will produce the same effect as if the loading has been initiated by the console directive "LOad", details of messages, recovering procedures, etc., will be found in part 6. It must, however, be noted that, should a catastrophic error arise, typing "GO" will not cause the 154 instruction to be repeated, hence a major restart must be carried out in such cases.
7. For further information consult part 5.

f) 155 SUSDP

Suspend the program, and dump the whole of its area, in binary program form, on to the specified peripheral unit, preceding the program proper with a request slip and terminating with a type 3 (suspend) entry block. Continue with the program on completion of the dump process.

Notes

1. Illegal
   a) if issued by other than member 0 of the program.
   b) if the specified device is not of a suitable type (suitable types include Card Punch, Paper Tape Punch, and Magnetic Tape but not Cassette Tape).
   c) if the specified unit is not assigned to the program.
   d) on a 'non-typewriter' 1901.

2. The dump produced will be in standard binary program form, and is initiated without any initial repositioning of the medium.

3. The request slip produced will be identical to that currently held, by Executive, in respect of the issuing program, except that bit 0 of word 2 will always be dumped as zero (and the check sum adjusted accordingly, if appropriate).

4. The dump will be terminated by a type 3 entry block, and the entry address will point to the location following that from which the 155 instruction was obeyed.

5. This instruction does not check the state of any other peripheral units assigned to the program, and, if any of these are active at the time the instruction is given, the result (notably the dumped data) will be undefined. In addition, an incorrect check sum may be generated rendering reload of the dump impossible. The effects will also be undefined if any suspension device is in "direct response" mode, or if any flag setting peripheral has been assigned to the program.

155 (cont'd)

6. Errors detected during the progress of the 155 instruction (excluding the illegal considerations mentioned above) will produce the same effect as if the dumping had been initiated by the console directive "DUmp"; details of messages, recovering procedures, etc., will be found in Part 6. It must, however, be noted that the use of the "GO" directive after an error does not cause the 155 instruction to be repeated completely, hence a major restart must be carried out in such cases.

7. When a program is dumped as a result of a 155 instruction, the dumped program will not contain details of the current setting of:- "V" (overflow), "C" (carry), Zero Suppression Mode, the current "Addressing Range" Mode, the current "Branch" Mode, and the current suspension state of members other than member C, and the associated memory states. Therefore, if the dump is subsequently reloaded, the state of "V" and Zero Suppression Mode will be indeterminate, the "Addressing Range" and "Branch" Modes will be in accordance with the request slip details, and members other than member 0 will be suspended in state SL.

Group 15 (Cont'd)

g) 156 (ALLOT)

Definition of purpose of N(M)

1. Bits 0-13 have undefined effect and must always be zero.
2. Bit 14 will, in the 'one' state, prevent the assigning of a peripheral device.
3. Bit 15 defines the 'replaced-X' condition, as given above in Section 15.1.
4. Bits 16, 17 have undefined effect and must always be zero.
5. Bits 18 to 23 give the internal type number. ~~Bits 18, 19 must always be zero.~~ [AL13]

Effect of this instruction

Should I not already have a unit X of type N(M), assign me one provided that bit 14 is zero, that devices of type N(M) are defined as being assignable by this instruction, and that a suitable device is available (suitability may be further qualified by Property Code bits in 1904 processors and upwards).

In addition, or if I already have a unit X of type N(M), reply in word 9 with the unit's absolute number if applicable and (on 1904 processors and upwards only) the properties it possesses. If on completion of this instruction I do not have a unit X of type N(M), indicate this in the reply.

(Note: bits 14 and 15 are ignored by Executive when determining the internal type no. from N(M))

Form of Reply in Word 9

1. If B0 (sign) = 1

The program did not have a unit X of type N, and it was not possible to assign one as a result of this 156 instruction. All other bits of the reply will be indeterminate.

Group 15 (cont.)

2. If B0 (sign) = 0

The program already has a unit X of type N, or a suitable unit has just been assigned by this instruction. Further information is available as follows:

Bits 1 - 8 : indeterminate at all times. AL/15

Bit 9 = 0 : the properties system is not implemented on this processor, hence bits 10-17 are indeterminate.

Bit 9 = 1 : the properties system is implemented on this processor, and bits 10-17 contain the property code of the device concerned.

Bits 18 - 23: contain the absolute number of the device assigned as unit X of type N. This does not apply to Direct Access Devices (EDS, MCF, FDS, etc.)
If the absolute number contained in B_{18-23} is zero, no meaning is to be attached to this part of the reply. This could occur, for example, when an input or output data stream was buffered in a backing store by the operating system without the object program being aware of the fact.

Notes

1. Where 'replaced-X' (see page 1) is not used, it is not possible to specify any properties; Executive will assign the first available device of type N that it locates.

When 'replaced-X' is used, a property code representing those properties which the program either requires the unit to have, or requires the unit not to have, may be placed in Bits 10 to 17 of X. Bit 9 must be set as follows:-

B9 = 0 if the unit is required to have at least all the specified properties.

B9 = 1 If the unit is required not to have any of the specified properties. (see note 4).

If either the properties system has not been implemented on the processor concerned, or a device has already been assigned as unit X of type N, Executive disregards Bits 9 to 17 of X.

2. When specifying 'properties required' for the assignation of a device, if bits 9-17 of X are all zero, this is equivalent to the 'non replaced-X' condition i.e. properties will be disregarded.

3. Executive only checks for the presence, or absence, as appropriate, of the properties specifically called for (by bits set to 'one' in the property code) when assigning a device.

4. The provision of facilities to ensure that properties are not present is done for the following reasons:

   a) To permit a program to specify a choice of properties (e.g. by saying, on Mag. Tape, 'not 800 b.p.i. Executive will choose the first available deck it locates which is working at 556, or 200, b.p.i.

   b) To minimise the number of bits required to specify various properties (e.g. to get 200 b.p.i. on Mag. Tape, the program has to specify NOT (B9=1) 800 b.p.i. and 556 b.p.i.

.3 157 (PERI)

157 X N(M)

This order initiates a transfer to or from a peripheral device or in the case of a file storage device, to open or close a file. The appropriate action and the peripheral unit concerned are defined in a control area whose starting address is specified by N(M).

The general layout of the control area, as applicable to non-file storage devices, is given below, specific values to be used are to be found under the appropriate peripheral action in Part 2 of the manual.

Word N — specifies the peripheral type, and mode of operation:

bit 0 — determines whether the unit number is defined by the integer X, as when B0 = 0, or by the 6 l.s. bits of accumulator X, as when B0 = 1. Bits 18, 19 of X must be 0 (see also Part 2 Section 0.3)

bits 1-8 — specify the peripheral type.

bits 9-23 — contain the mode of operation, i.e. the manner in which Executive is to deal with the data.

Word N + 1 — is the Peripheral Reply Word. (Except with flag setting devices when it may be the address of the interrupt flag word.)

bit 0 — is set to 1 (except in opening and closing files) from the time that the 157 order is given, until the action specified has been completed, as far as the object program is concerned. B0 is set to 0 when the transfer is completed and checked.

bits 1-23 — at completion of the transfer are set to provide certain information about the transfer as completed. For details refer to relevant peripheral section in Part 2 of the manual.

WORD N + 2 — contains the count of units to be transferred i.e. characters, words, sectors etc. depending on the type of peripheral.

In some cases this count is a maximum limit but in others it is the actual quantity to be transferred.

WORD N + 3 — contains the starting address of the transfer.

In word transfer peripherals, only the l.s. 15 bits are used, or with a 1906/7 operating in Extended Store Mode, the l.s. 22 bits are used.

In character transfer peripherals the l.s. 15 bits, or with a 1906/7 operating in Extended Store Mode, the l.s. 22 bits, are used to indicate the starting address, and the 2 m.s. bits are used to indicate the character position within that word.

Notes

1. Executive checks that N to N + 3 are in the program area.
2. For input and output transfers, Executive adds the starting address (N + 3) and the maximum size of the transfer (N + 2) to check for limit violation.
3. In the case of an attempted violation, Executive suspends the program and sub-programs, indicating this by a console message.
4. Although Executive has finished with words 0 and 2 onwards of the control area as soon as the object program gets to the instruction after the PERI, it is not advised that the control area be altered until the order has been completed (i.e. until the time that a SUSBY to the device concerned would be released). In particular, disregard of this will cause difficulty in program testing, and in diagnosing possible hardware faults.
5. Whilst a transfer is in progress its reply word is liable to be changed, and must not be used in any way by the object program (except, of course, for checking if the transfer is still in progress).
6. If a reply word is common to more than one transfer, and both transfers are active at the same time, that reply word is meaningless in respect of all those transfers (it cannot even be used to test if any, or all, of the transfers have finished).
7. If a second PERI is issued to a device whilst that device is active, and both PERI's use the same reply word, the reply to the first transfer is lost. Although this may seem to be a reasonable situation, it is not an advisable technique, as it prevents the object program checking that the first transfer was completed correctly, e.g. if the first transfer FAILED and the operator typed GO, the object program would not know, and could receive erroneous data as a result.
8. In general, the same core data area may be used for either input or output (but not both) to two or more peripherals simultaneously. Exceptions are data areas for paper tape input and the block sequence word for cassette tape read/write. Object program orders may refer to an active output data area for 'read-out' only; under no circumstances may an object program attempt to change the contents of such an area.

The Group 16 instructions, all Extracodes, are a miscellaneous group covering communications between program and Executive, or the operating system (which includes human operators), and the interworking of program members. There are provisions for suspension and deletion.

.1 160 instruction.

(a) General.

1). The several variants of this instruction provide the means by which an object program may pass a text, consisting of up to 40 graphic characters, to the operating system, and, if it so desires, indicate to the operating system that it has reached a point where certain operating system action is required.

2). The contents of the X field of this instruction determines which variant of the instruction is to be performed.

3). N(M) is interpreted as the first (or only) word of a "control area", which contains information defining the store area occupied by the graphic text which is to be communicated to the operating system.

4). The control area addressed by N(M) has one of the following forms:

i) 15AM or 22AM

Word N(M)	Bits 0 to 8	number of characters in text
	Bits 9 to 23	address of first word of text

ii) 22AM

Word N(M)	Bits 0 to 17	zero
	Bits 18 to 23	number of characters in text
Word N(M) + 1	Bits 0 and 1	zero
	Bits 2 to 23	address of first word of text

Group 16 (cont'd.)

(160 cont'd).

Either of the above formats is allowed in 22AM. They can be distinguished by whether or not bits 0 to 8 of N(M) are zero.

5). The text must commence at character 0 of a word, occupy one continuous area, and must be not less than 1 or more than 40 characters in length. The characters output are the 1900 series six-bit internal characters (excluding the characters represented by octal 74, 75, 76 and 77). (See page 8 below).

6). On configurations without console typewriters, the text is not output; instead the least significant 12 bits of the effective value of N(M) are displayed, together with the value of X.

(b)X = 0 (SUSTY)

This variant causes the issuing program member to be suspended as soon as the message has been typed until reactivated by the operating system.

N.B. any, or all, of the other members of the program may also be suspended until the issuing member is reactivated by the operating system.

The message is prefixed by the word HALTED. In a multi-program environment, the program name and number also appear, typically in a form such as S#JACK.

If the operating system restarts the program (usually by the GO message), then the instruction following the SUSTY is obeyed (unless the operating system selects a special restart action i.e. the GO AT message).

(c)X = 1 (DISTY)

This variant causes the text to be passed to the operating system, the issuing program continuing after the passing over of the text without the need for intervention from the operating system (except on configurations without console typewriters).

(160 cont'd.)

The message is prefixed by the word DISPLAY. In a multi-program environment the program name and number also appear.

On the 1901 with console typewriter, the 1902 and 1903, the issuing program is suspended while the text is typed. On the 1904 and upwards, this variant copies the text into Executive's buffer area for typing, and then the program proceeds to the next instruction. The issuing program is not suspended. In either case, the activity initiated by the DISTY instruction may be regarded as being completed before the following instruction of the same member is obtained, interpreted and acted upon.

(d)X = 2 (DELTY)

This variant causes all members of the issuing program to be suspended, and the program to be deleted. However, the object program's accumulators will be left uncleared, allowing information to be passed from the deleted program to its successor. In general, the text will be interpreted as a request to the operating system that some other program be brought into use. The output message must form a valid 'LOad' or 'FInd' message. (See Part 5, Section 4, Page 3 for the definitions of 'LOad' and 'FInd).

The message typed out ('LOad' or 'FInd') is prefixed by the word DELETED. In multi-programming environments the program name and number also appear. If the message is acceptable to Executive an 'OK' suffix will be typed, but if unacceptable, an appropriate error message will be typed (see error tables at the end of Part 6). "Accumulator Carry Forward" is not operative if the 'LOad' or 'FInd' message is not successfully implemented and has to be overridden by an operator 'LOad' or 'FInd' message.

The following general considerations must be taken into account when using this variant in an object program:-

1). With the exception of the device specified in the 'LOad' message (when used), the effect upon the position of the data bearing medium

(160 cont'd.)

of any devices still assigned to the issuing program is undefined.

2). In the case where the 'LOad' message is used, the data bearing medium, on the device specified in that message, will, provided that the message is acceptable to the operating system, not be repositioned before the loading process proper commences.

Provided that the message output satisfies Executive's validity checks, the peripheral named as the loading device for the new program, will, although released from the program being deleted, not be made available for assignment to another program (in effect, it becomes assigned to Executive).

3). If, at the time that a program issues this variant, any 157(PERI) instructions, issued by any member of the program, which can have determinable replies, indicate in these replies that the action or actions requested by the 157 instructions are still in progress, then it is possible that the action or actions called for will never be successfully completed. Furthermore, if this condition occurs with respect to the device specified in the 'LOad' message (i.e. the device from which another program is to be loaded) then it is possible that the loading process will not be successfully completed.

4). If the text does not comprise a valid message to the operating system, or, if it cannot be dealt with satisfactorily for some other reason, the operating system will take appropriate action, but in no circumstances will the issuing program be reactivated as a result.

.2 161 instruction.

(a)General.

1). This instruction, in its various forms, provides a means by which an object program may pass a text consisting of two characters to the operating system, and, if it so desires, indicate to the operating system that it has reached a point where certain operating system action is required.

2). The contents of the X field of this instruction determine which variant of the instruction is to be performed.

3). N(M) is regarded as a literal operand, and the least significant 12 bits of its effective value constitute the text.

4). The text is fixed at two 1900 series 6-bit internal characters, with the exception that the characters represented by octal 74, 75, 76 and 77 cannot be used. (See page 8 below). The least significant 6 bits are taken as the second character in the text.

5). The text does not appear in store as a distinct entity, except in the particular case of an unmodified instruction (i.e. one in which neither normal, nor pre-modification is used).

6). On configurations without console typewriters, the text is displayed, together with the value of X.

7). In some environments, the distinction between 160 and 161 instructions may not be explicitly shown externally; therefore, the distinction between a text output by a 161, and an equivalent text output by a 160, should not be regarded as adequate to indicate different conditions.

(b)X = 0 (SUSWT).

This variant causes the issuing program member to be suspended as soon

(161 cont'd.)

as the two characters have been typed, until reactivated by the operating system.

N.B. any, or all, of the other members of the program may also be suspended until the issuing member is reactivated by the operating system.

The message is prefixed by the word HALTED. In a multi-program environment, the program name and number also appear (typically in a form such as S#JACK).

If the operating system restarts the program (usually by the GO message), then the instruction following the SUSWT is obeyed (unless the operating system selects a special restart action i.e. the GO AT message).

(c)X = 1 (DISP).

This variant causes the two characters to be passed to the operating system, the issuing program continuing after the passing over of the text without the need for intervention from the operating system (except on configurations without console typewriters).

The message is prefixed by the word DISPLAY. In a multi-program environment the program name and number also appear.

On the 1901 with console typewriter, the 1902 and 1903, the issuing program is suspended while the two characters are typed. With the 1904 and upwards, this variant copies the characters into Executive's buffer area for typing, and then the program proceeds to the next instruction. The issuing program is not suspended. In either case, the activity initiated by the DISTY instruction may be regarded as being completed before the following instruction of the same member is obtained, interpreted and acted upon.

(d)X = 2 (DEL).

This variant causes all members of the issuing program to be suspended,

(161 cont'd.)

and the program to be deleted.

The two characters output are prefixed by the word DELETED. In multi-program environments the program name and number also appear.

No special significance is attached to the contents of the text output, hence this instruction cannot directly request the bringing into use of another program (as is the case with the 160/2(DELTY)).

The following general considerations must be taken into account when using this variant in an object program:-

1). The effect upon the position of the data bearing medium of any devices still assigned to the issuing program is undefined.

2). If, at the time that a program issues this variant, any 157(PERI) instructions, issued by any member of the program, which can have determinable replies, indicate in these replies that the action or actions requested by the 157 instructions are still in progress, then it is possible that the action or actions called for will never be successfully completed.

Character code for messages output by 160 or 161

Least significant 3 bits	Most significant 3 bits of six-bit character							
	0	1	2	3	4	5	6	7
0	0	8	SPACE	(	@	H	P	X
1	1	9	!	)	A	I	Q	Y
2	2	:	"	*	B	J	R	Z
3	3	;	#	+	C	K	S	[
4	4	<	£	,	D	L	T	x
5	5	=	%	-	E	M	U	x
6	6	>	&	.	F	N	V	x
7	7	?	'	/	G	O	W	x

Note: The four codes marked with 'x' must not be used in any message.

.3 162 (SUSMA)

.4 163 (AUTO)

.5 164 (SUSAR/SUSIN)

With one exception, these three instructions are used only in sub-programming. Their actions and use are described under that heading in Part 4, Section 0.

The exception is the 164 (SUSIN) with X=2 and N(M)=0, which is provided on a processor that does not have sub-programming facilities but to which one or more flag-setting devices are attached. Its definition is as follows (this is a compatible subset of the range definition referred to above):-

If one or more flag-setting interrupts, from any peripheral assigned to this program, have occurred since it last issued a 164 (SUSIN) instruction, clear the indicator used to record such interrupts, and proceed to the next instruction.

If this indicator is already clear, suspend this program until a flag-setting interrupt occurs, at which time proceed to the next instruction and leave the indicator clear.

N.B. When the program is first loaded it is regarded as having just been released from suspension due to a 164 (SUSIN) instruction, and the indicator is left clear.

.6 165 (GIVE).

The various forms of this instruction provide the means by which an object program may obtain information from Executive, or request it to perform certain actions in respect of an object program. The facilities provided are, in general, confined to actions which can be performed by the operating system without the need for specific information having to be provided, to the operating system, from an external source.

The nature of the communication is defined by the value of N(M), any data required by Executive from the object program being in accumulator X (and possibly X + 1). Executive's reply to the object program is placed in accumulator X (and possibly X + 1).

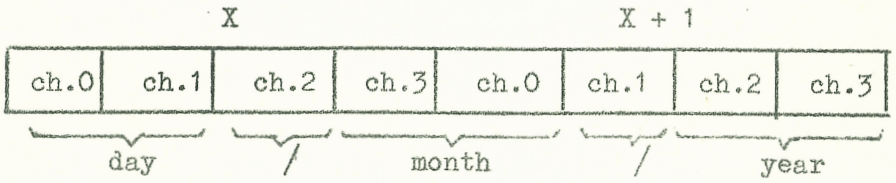
(a) N(M) = 0 Give date, in binary form.

Executive places the date, as a binary number, in accumulator X. The date is represented by a count of the number of days elapsed since the 31st. December, 1899 (1st. January, 1900 is day 1 and leap years are correctly accounted for).

The contents of X + 1 are unaffected by this form of the 165. The initial contents of X are not significant.

(b) N(M) = 1 Give date, in calendar form.

Executive places the date, in character form (64 character 1900 internal code), with day, month and year separated by solidi, into accumulators X and X + 1.



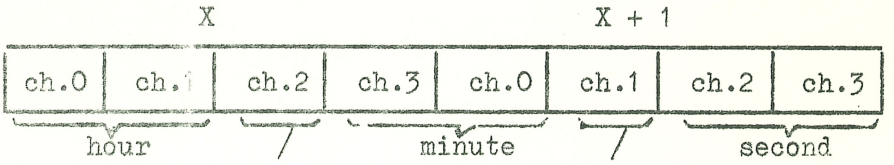
(165 cont'd.)

Zero suppression is not applied to any field.

The original contents of accumulators X and X + 1 have no significance.

(c)N(M) = 2 Give time, in character form.

Provided that the central processor is fitted with a real time clock, Executive places the time (24 hour system) in character form (64 character 1900 internal code), with hour, minute and second separated by solidi, into accumulators X and X + 1. By the time the object program has accessed the response, it may be "slow" by an amount which can range from one second upwards.



Zero suppression is not applied to any field.

The original contents of accumulators X and X + 1 have no significance.

If the central processor is not fitted with a real time clock, the effect of this form of the 165 instruction is to zeroize both accumulators.

(d)N(M) = 3 Give size of storage.

Executive places, as a 24 bit binary number, the number of words of core store currently assigned to the issuing program.

The contents of X + 1 are unaffected by this form of the instruction.

The initial contents of X are not significant.

Note: the number of words given may not be related to the amount requested by the object program, either in its request block or by means of a preceding 165/4. This is due to the fact that, in some environments,

(165 cont'd.)

the amount of store assigned to the program is determined by other factors (e.g. in a single program environment, all available core store may be assigned to the object program).

(e)N(M) = 4 Change the amount of storage assigned.

1). Initial contents of accumulator X.

The 24-bit binary number in X is interpreted by the operating system as the amount of storage which the issuing program would like to occupy (where the amount requested is not a multiple of the "unit of increment of datum" for the environment concerned, the operating system will round it up to the next higher such multiple before attempting to take any action).

The minimum value that may validly appear is the lowest multiple of 64, which, when thought of as an address, will not be the address of any location whose usage is restricted for that particular program (for restricted addresses see Part 0, Sect. 6). The number of restricted addresses will, therefore, be dependent upon the number of members in the issuing program; e.g. if the issuing program has only one member, the minimum storage allocation will be 64; if the issuing program has 2 members, the minimum will be 128, etc.

Otherwise, the amount of storage requested may be more than, less than, or the same as, that currently assigned to the issuing program.

2). Final contents of accumulator X.

The number of words of core store that are currently assigned to the issuing program on completion of the instruction, in the same form as the 165/3 variant.

3). Effect on 165/4 (in general).

The actual effect of this variant is operating environment dependent,

(165 cont'd.)

and, within one operating environment may differ from one program to another. The details below list the possibilities which may occur, and which must be allowed for by an object program:-

i) No action may be taken, the response being the same as would have been given to the 165/3 variant.

ii) The amount of store may be adjusted to the amount requested (as rounded to the next higher unit of increment of datum, if appropriate).

iii) In the case of a request for an increase in storage: the amount of storage may be increased to some amount between that already assigned and that requested (typically, this may occur when some spare storage is available, although not enough to meet the request in full).

If the use of 165/4 actually causes further storage to be assigned, such further storage will contain zero in all words.

The use of this variant can cause other programs in the system to be suspended for a significant period of time.

The stored request block of the issuing program is not updated when an alteration in the amount of store assigned occurs.

4). Action of individual Executives on receipt of a 165/4 instruction.

1901/2/3 (excluding DUAL programming Executives).

On these machines the whole of the area available for object program is assigned to a program at load time, regardless of the stated store requirement. As a result, a 165/4 instruction requires no action, other than the reply from Executive, which treats a 165/4 in exactly the same way as a 165/3.

1901 unit of increment of datum register: 128 words.
1902/3 unit of increment of datum register: 256 words.

(165 cont'd.)

1902/3 (DUAL programming Executives (EX2M)).

With the DUAL programming 1902/3 Executives, programs are not moved around to create free store for the program that requests more store.

If insufficient store is currently available between the program requesting store and the one above it in the store, the lower should be explicitly suspended until such time as the higher is deleted, at which time the free storage becomes available to the lower program. If there is not a higher program there is no possibility on this run of getting extra store and special action must be taken, i.e. it may be possible to restart the program after deletion and reloading lower in the store.

1902/3 DUAL programming unit of increment of datum register: 256 words.

1904/5/6/7

i) If the restriction on the minimum value of the request are violated the 165/4 instruction will be flagged as "ILLEGAL K".

ii) If sufficient store is available, Executive will carry out the requested alteration in full, replying accordingly.

iii) If insufficient store is available, Executive will assign all the spare store (if any) to the program, setting the reply to the appropriate value.

iv) Other programs in the processor at the time may require relocation to accomodate the change in size of the program issuing the 165/4 order. This means that the total time to adjust the size of a program cannot readily be estimated, but, in general will be similar to the time taken when a program is deleted and a new one loaded (excluding the actual reading in of the new program). Furthermore,

(165 cont'd.)

the 165/4 order may produce a "BUSY" message, in respect of a device assigned to one of the programs which has to be relocated, if such a device is active at the time, or in respect of a device assigned to this program, which requires operator action.

(v) A message in the form S #JACK; CORE xxxx will be typed on the console, where 'S' is the number of the program member which issued the 165/4 order, and 'xxxx' is the decimal value of the new size of the whole program concerned.

(vi) 1904/5 unit of increment of datum register: 64 words.
1906/7 unit of increment of datum register: 128 words.

5). Effect of dumping (operator directive or 155 order).

When a dump is carried out the request block will contain the store requirement figure as it appeared in the original request block, regardless of any subsequent alteration of store allocation (be such action due to a 165/4 order or operator action when loading a program). However, in multi-program environments, the data actually output will consist of the whole of the program's storage as at the time of dumping, i.e. the amount of store dumped is the same as the amount which would be given as the reply to a 165/3 instruction; on single program processors, Executive outputs the whole of the area available for the object program but eliminates any words at the high address end of this area which contain zero, thereby, effectively, only dumping the area that the program was using.

(f)N(M) = 5 Give details of central processor and operating environment.

Executive replies, in accumulators X and X + 1, with coded information concerning details of instruction code and other environmental features which are not necessarily present in all 1900 series processors but which are present in this processor, excluding information about peripheral devices. The initial contents of X and X + 1 are of no significance.

(165 cont'd.)

The basic reply is a 48-bit binary pattern, generally bit significant with a "one" indicating that the specified feature is present. However, one bit in the first accumulator (X) is an escape bit; if it is "zero", the contents of the second accumulator (X + 1) will be indeterminate, if it is a "one", the contents of X + 1 will be meaningful. This has been done in order to minimize the amount of Executive coding on small processors and means that if the escape bit is "zero" the features which otherwise would be indicated in X + 1 are all absent.

Bits 1 to 11 of the accumulator X define the logical properties of the order code (excluding peripheral device availability). Bits 12 to 17 and also bit 5 indicate whether certain orders are implemented by extracode or hardware. No provision has been made to distinguish between hardware and extracode for instructions 130, 131, 076.

Details of the significance of each bit of the reply appear below. Reserved bits will be in an indetminate state: these bits may be defined at a later date.

Note: If this instruction has been used and the object program is later dumped, the instruction should be repeated following the dump if there if any possibility that a restart will be on a different processor/Executive configuration.

(165 cont'd.)

Reply format when N(M) = 5

ACCUMULATOR X

Bit	Title	Value 0	Value 1
0	Escape	(X+1) indeterminate	(X+1) defined
1	Operating Environment	Handkeys (restricted 160, no 155)	Typewriter
2	(Reserved)	(not used)	(not used)
3	Level C	No	Yes (bit 4 must be 1)
4	Level A/B	A	B
5	Sub-level	1 (also 111E, 113E in level A1)	2 (also 114H, 115H always; 116H in level C2)
6	Floating Point	No	Yes
7	Floating Point Definitions	Standard	Generalised
8	Sub-programs	No	Yes
9	Sub-programs Definitions	Standard	Combined
10,11	(Reserved)	(not used)	(not used)
Speed Factors:			
12	040-042, 044-046	E	H
13	043, 047	E	H
14	126,127	E	H
15	132-137	E or not available	H
16,17	(Reserved)	(not used)	(not used)
System Features:			
18-23	(Reserved)	(not used)	(not used)

ACCUMULATOR X + 1 (R e s e r v e d)

(165 cont'd.)

Notes: H = by hardware; E = by extracode

For Instruction Classification, see Part 0, Sect. 7, (i.e. the significance of the different levels of classification in the 1900 instruction code).

(g) N(M) = 6; N(M) = 7 Undefined.

(h) N(M) = 8 Give current Addressing Range and Branch Modes.

Executive places in accumulator X coded information which indicates the Addressing-Range and Branch Modes in which the issuing program is currently operating, as follows:-

Bits 0 to 20	:	reserved
Bits 21	:	zero if in DBM one if in EBM
Bit 22	:	reserved
Bit 23	:	zero if in 15AM one if in 22AM

The contents of X + 1 are unaffected by this form of the instruction.

The initial contents of X are not significant.

(i) N(M) = 9 Request for change of Addressing-Range, and/or Branch Mode.

The contents of X + 1 are unaffected by this variant.

1). Initial contents of accumulator X.

Accumulator X contains coded information, indicating the Addressing-Range and Branch Modes in which the issuing program member wishes to operate subsequent to this instruction, as follows:-

Bits 0 to 20	:	must be zero.
Bit 21	:	zero if DBM is desired. one if EBM is desired.

(165 cont'd.)

Bit 22 : must be zero.

Bit 23 : zero if 15AM is desired.
one if 22AM is desired.

2). Final contents of accumulator X.

Accumulator X will contain coded information, indicating the Addressing-Range and Branch Modes in which the issuing program member is operating after completion of the instruction, in the same form as the 165/8 variant.

N.B. the response may not correspond with the request; see below.

3). Implementation.

In environments which do not provide all Addressing-Range and Branch Modes, an attempt to switch to a non-available mode will be ignored, the issuing member being left in the mode in which it is already operating, with the response set accordingly.

N.B. in this respect, the two switching options (i.e. Addressing-Range, and of Branch Modes) operate independently. It is the responsibility of the issuing program member to take appropriate action when an attempt to establish a particular mode fails in this way.

It is valid for a program member to request a "change" which would not actually alter either the Addressing-Range, or Branch Modes from those already established.

.8 166 (RRQ)

This instruction has two variants, differentiated by the X field, and causes certain alterations to take place starting at the address N(M).

X = 0 Read the request block stored by Executive into the 14 or 16 word area starting at N(M). Bit 0 of word 2 of the request block will be cleared and the checksum in Word 13 adjusted.

X = 1 Replace the request block stored by Executive by the contents of the 14 or 16 word area starting at N(M).

Notes

1. The request block read into the object program area by a 166 (X=0) instruction is that which was stored by the last 166 (X=1) instruction, or, if no such instruction was obeyed, that which was read when the program was loaded. The core usage, peripheral usage and/or priorities indicated by the request block read in by the 166 (X=0) instruction may differ from those currently pertaining, due to the use of GIVE with N(M)=4, ALLOT or REL instructions within the program, or to operator action; or to an earlier 166 (X=1).

2. The request block is read by the 166 (X=0) order into the 16 word area starting at N(M); if the request block is only 14 words long, then the contents of the last two words of the area are indeterminate.

3. The reservation violation check on the 166 instruction, with X=0 or 1, is on the basis of a 16 word transfer, whether the request block itself is 14 or 16 words long.

4. The 166 (X=1) instruction may replace a 14 word request block by a 16 word request block, or vice versa, or another block of the same length.

5. Member 0 is the only member which may issue a 166/1, and may do so only when other members are in state SL.

(.8 cont'd)

The following effects can be achieved:-

(a) Change of program name.
(b) Reduction of trusted status.
(c) Introduction of new program members.
(d) Change of priority of any program members.

The instruction must NOT attempt to:-

(a) Change to an illegal name.
(b) Increase trusted status.
(c) Delete a program member.
(d) Set bit 0 of word 2 of the request slip to 1.

If the instruction attempts to duplicate a program name, the program will be suspended, unless an operating system intervenes.

Changes to the program name extension, the peripheral requests and the core store request will be noted, but not implemented, so that they will only affect subsequent 166/0 instructions and request blocks dumped by Executive.

When used in machines with 4K store or with no console typewriter the 166 instruction has null effect, except that C is cleared.

.9 167 Instruction is illegal.

Instructions 170 to 177 are illegal.

CONTENTS

Section:

0 Assignation, Release, Referencing.

1 List of Peripheral Types.

2 Execution of Transfers.

3 Peripheral Incidents.

4 Peripheral Device Details.

- 4.0 Paper Tape.
- 4.1 Paper Tape Readers.
- 4.2 Paper Tape Punches.
- 4.3 Line Printers.
- 4.4 Punched Cards.
- 4.5 Card Readers.
- 4.6 Card Punches.
- 4.7 Magnetic Tape Formats.
- 4.8 Magnetic Tape Devices.
- 4.9 Direct Access Devices.
- 4.10 Direct Access Control Standards.

5 Peripheral Device Properties System.

6 Own Monitoring Facilities.

1. Assignation

The assignation of peripheral units for use by a program is carried out by Executive and may take the form of 'dynamic assignation' initiated by program, or 'console assignation' initiated by the operator. (Cynamic assignation is not available with EX1H or EX2L).

a. Slow Peripherals

i. Dynamic

A slow peripheral is assigned by program either when indicated in the request slip (generated by PLAN Compiler from the # PERIPHERAL directive, or when a 156 (ALLOT) instruction is encountered) but the operator or operating system must ensure that the right files or output media are on the relevant peripherals.

ii. Console

Assignation via the console typewriter is initiated by using the GIve input directive (see Part 6 of this manual). This method is always used when an operator wishes to make a particular assignment.

b. Magnetic Tape Decks

i. Dynamic

Assignation of magnetic tape decks depends of course upon which deck the required reel has been loaded on by the operator. The assignation is initiated by a 157 instruction with a special mode.

The program provides the File Name, Generation Number and Reel Sequence Number of the tape, and Executive searches the unassigned tape decks for this tape; if it cannot find the required tape, the program is suspended and the Operator notified.

ii. Console.

Assignation via the console is initiated by using the GIve input directive (see Part 6). This method is used when a tape with a non-standard label is to be used.

.2 Release

a) Slow Peripherals

A peripheral unit is released from its assignation to a program when the 15¢ instruction is issued, when the program is deleted, or when the operator uses the TAKE console input directive (see part 6). The peripheral is not necessarily disengaged (except by the 15¢ instruction), unless its disengagement is required by the operating system for correct operation when the peripheral is reassigned.

b) Magnetic Tape Decks

A deck is released from its assignment to a program when a Close File instruction is issued; disengagement is determined by the instruction. A deck is also released when the program is deleted if no Close File instruction has been issued for it; however a deck referred to in a 160/2 order message is not rewound at this time if the deletion was due to such an order. If the operator disengages a deck without notification from xecutive that an incident has occurred on it, program errors may occur in consequence.

A deck is also released when the operator uses the TAKE console input directive.

.3 Referencing

For the purpose of initiating peripheral actions, each peripheral type has been allocated a Type No. (see Section 1). In addition, each program is written with its own logical numbering of peripherals: if the peripherals are assigned by request slip, then the numbering is from 0 onwards for each type; if assignment is by a 156 instruction then any number from 0 to 15 may be used. In effect some Flag-setting devices (e.g. Multiplexor) function as if they were several individual devices, except that they can only be assigned to a program in their entirety. The effect of this is that, in most modes of PERI instruction, the "program's unit number" is interpreted in a different way from the standard form (see the specifications of these peripheral devices).

On assigning a peripheral unit to a program, Executive sets up a relationship between the 'Type No. and Program's Unit No.' by which the unit is known to the programmer, and the permanent 'absolute unit number' by which the unit is known to Executive and Operators. Thus Executive is thereafter able to identify any peripheral unit referred to by program.

Peripheral Type Nos. and Property Codes. April, 1967

Note:- Under 'Property Code' undefined bits, indicated by 'X', are indeterminate.

.1 Commonly Encountered Devices

Type No.	Peripheral	Catalogue Number	Property Code
0	TR Paper Tape Readers	1915/1 (300 c.p.s.)	X00XXXXX
		1916/2 (1,000 c.p.s.)	X01XXXXX
1	TP Paper Tape Punches	1925/2	X00XXXXX
2	LP Line Printers	1931/1 (96 print positions)	X00000X1
		1931/2 (120 print positions)	X00010X1
		1932/1 (96 print positions)	X01000X1
		1932/2 (120 print positions)	X01010X1
		1933/1 (96 print positions)	X11001X1
		1933/2 (120 print positions)	X11011X1
		1933/3 (160 print positions)	X11111X1
		2401/1 (96 print positions)	X00000X0
		2401/2 (120 print positions)	X00010X0
		2402/1 (96 print positions)	X01000X1
		2402/2 (120 print positions)	X01010X1
		Via 1004	X00110X0
3	CR Card Readers	1911/1 (without Card Image)	X1000XX0
		1911/1 (With Card Image)	X1001XX0
		1912/1 (without Card Image)	X0000XX0
		1912/1 (with Card Image)	X0001XX0
		1912/9 (with Card Image)	X1001XX0
		2101 (without Card Image)	X1100XX0
		2101 (with Card Image)	X1101XX0
		2102 (without Card Image)	XX010XX0
		PF 311 (without Card Image)	X0100XX0
		PF 311 (with Card Image)	X0101XX0
		Via 1004	X0000XX0
4.	CP Card Punches	1920	X01XXXXX
		1922	X00XXXXX
		2151	X11XXXXX
		Via 1004	X10XXXXX

Peripheral Type Nos. and Property Codes

.1 cont'd

Type No.	Peripheral	Catalogue Number	Property Code
5	MT Magnetic Tape Units (Industry Compatible)	1970 (7-track 200 b.p.i.)	01111X00
		1970 (7-track 556 b.p.i.)	01111X01
		1974 (7-track 200 b.p.i.)	01111X00
		1974 (7-track 556 b.p.i.)	01111X01
		1974 (7-track 800 b.p.i.)	01111X10
		1971 (7-track 200 b.p.i.)	00000X00
		1971 (7-track 556 b.p.i.)	00000X01
		1972 (7-track 200 b.p.i.)	00101X00
		1972 (7-track 556 b.p.i.)	00101X01
		1973 (7-track 200 b.p.i.)	00101X00
		1973 (7-track 556 b.p.i.)	00101X01
		1973 (7-track 800 b.p.i.)	00101X10
		2504 (9-track with 'read reverse')	10111X00
		2504 (9-track without 'read reverse')	10101X00
		2505 (9-track with 'read reverse')	11011X00
		2505 (9-track without 'read reverse')	11001X00
6	ED Exchangeable Disc Stores (E.D.S.)	2801 2802	
7	MC Magnetic Card File (M.C.F.)	1958	
8	Data Disc File	1956	
9	DR Magnetic Drums	1962 (32768 words) 1963 (131072 words) 1964 (524288 words)	

(.1 cont.)

Type No.	Peripheral	Catalogue Number	Property Code
10	IT Single Channel Communication Devices	7006 - Standard Interface/Data Terminal Interface (Uniplexer) 1938 - Interrogating Typewriter	
11	MX Communications Multiplexers	7007	
12	CT Cassette Loading Magnetic Tapes	2501	
13	FD Fixed Disc Stores (Bryant Discs etc.)	2803, 2804, 2805, 2806	
14	ID Interrogating Visual Display Units	7154	

Peripheral Type Nos. and Property Codes.

.2 Less Common Devices

Type No.	Peripheral	Catalogue Number	Property Code
20	GP Digital Incremental Plotters	1934/1	QXX00X00
		1934/2	QXX00X01
		1934/3	QXX01X00
		1934/4	QXX01X01
		1934/5	1XX00X01
		1934/6	1XX01X01
21	General Purpose Visual (C.R.T.) Display Units		
22			
23	Universal Document Transports		

.3 Rare Devices

Type No.	Peripheral		Catalogue number
59	ET	Magnetic Tape (Emidec 1100 Units)	1198/99
58	RR	Magnetic Tally Roll Readers	2701
57	CS	Document Sorter/Readers (Cheque Sorters)	8500
56	OO	Link to Orion 1	
55	OI	Link from Orion 1	
54	EM	E.M.I. Camera/Fingerprint Scanner	
53	CO	Copath Curve Generator	
52	GE	General Purpose Executive Package	
51	OP	Interprogram Communication: Output	
50	IP	Interprogram Communication: Input	

EXECUTION OF PERIPHERAL TRANSFER OPERATIONS

A transfer operation is initiated by a 157 instruction (q.v.) causing entry to executive. The type of operation and unit required are specified in a series of words known as a Control Area. The general form for this area, and definition of the 157 order are to be found in section 15, Part 1. Upon entry, Executive then:

1. determines the reason for entry,
2. identifies the absolute peripheral number,
3. checks whether the unit is on-line and not busy,
4. refers to the control area of the 157 instruction and records
   a) mode of operation and start command
   b) the count and starting address of the transfer,
5. tests whether the transfer will exceed the program's limit,
6. checks, only on 1902/3, that the transfer will not overload the central processor,
7. if all checks are satisfied, signals to appropriate peripheral control to start the transfer,
8. returns control to the next instruction of the user program, provided that no other interrupts are waiting to be serviced.

When any transfer has been completed, an interrupt occurs and Executive checks whether the transfer has been carried out successfully. If there has been an error, Executive takes the necessary action, dependent on the type of peripheral involved. See section 4 for further details.

If the transfer is successful, Executive removes any suspension which was due to that transfer and then returns control to the highest priority program/sub-program which is not suspended at the time.

Peripheral Incidents

1. Disengaged Unit

The following does not apply to file storage or magnetic tape devices.

A peripheral unit is disengaged by Executive and may result from the following:

a) 152 instruction in the object program,

b) operator pressing 'Hold' button,

c) device becoming inoperable due to fault or lack of appropriate medium,

d) deletion of program and release of assigned peripheral units,

e) TC4 character on 8-track paper-tape reader.

If any of the above occur during a transfer operation, Executive awaits completion of the transfer before disengaging the unit.

With the 1901/02/03 processors, when a program requests a transfer on a peripheral unit which has been assigned to the program but is in the disengaged state, then the program is suspended and a 'UNIT Q FIX' message typed out on the console typewriter.

With the 1904/5/6/7/9 operator Executives, the program will continue; the reply word for the transfer is set negative, and the peripheral is busy in the sense that a further transfer instruction, or a SUSBY instruction, to the same peripheral, will result in suspension.

When an operator re-engages the unit (having cleared any faults) the suspension is lifted and the transfer request reinstated.

Note: For simplified facility peripheral devices, see Part 2, Section 4.

.2 Failed Transfers

The action taken when a peripheral device indicates that a transfer is in some way faulty (e.g. parity failure, or read-check failure) depends upon the type of device (see also Section 6, Own Monitoring). For this purpose devices are classified in the following way:-

Automatic Repeat Devices

Transfers can be repeated automatically under Executive control. Devices in this class are the file storage devices and magnetic tape decks, and some card punches. In the latter case some operator action may ultimately be required to extract offset cards from the pack.

Assisted Repeat Devices

The transfer can be repeated automatically under Executive control after the operator has suitably repositioned the data bearing medium.

This class includes card and paper tape readers, and the 33 c.p.m. card punch. Line printers and paper tape punches are in this class only on 1904/5/6/7/9 Executives, otherwise they are Non-Repeat Devices.

Non-Repeat Devices

Because of the type of data-bearing medium involved, a simple repeat of transfer is of very little use, and special program restart action is needed. This class includes line-printers and paper tape punches.

The ways in which the various processors deal with certain failure conditions differ. The actions are detailed under the three classes mentioned above.

a) Automatic Repeat Devices

i) The transfer will be repeated under Executive control up to some fixed number (n) of times (a specific value is assigned to n for each type of device). If the transfer takes place correctly during this process then it is made to appear terminated to the ordinary program. The program is not aware that anything has happened.

While Executive is repeating it, the transfer is regarded as still in progress, and suspension of the requesting program may occur in the usual ways.

ii) If the transfer fails n times in succession the following occurs.

1902 & 3

A "UNIT Q ERR" message is typed. The program is suspended awaiting operator action whether or not it was previously suspended awaiting termination of the transfer.

The operator can type a 'GO AT' message to enter the program's recovery procedure. The order is discarded and B3 of the reply word is set to 1.

If the operator types "GO", Executive carries on with the program from the point where it was interrupted. B3 of the reply word is set to 1.

For magnetic tape, each time the total number of repeats in the current pass reaches 10, ~~when reading, or 5 when writing~~ Executive types "UNIT Q 10R" and resets the count to zero. AL [4/12]

For the card punch, Executive types "UNIT q ERR" for each repeat.

1904 and up

A message of the form s#prog; UNIT q:- FAIL ∅ is typed [AL3] (see part 6, section 2, page 6). The transfer is made to appear terminated to the program i.e. the reply word is reset positive, but a fail marker is also set in the reply word (B3 = 1), together with the store area next address or the unexpired count for those devices where one of these is the standard reply. The program is suspended awaiting operator action, regardless of whether it was previously suspended awaiting termination of the transfer. A 'GO' message causes the program (or sub-program) to continue from the point of suspension.

Alternatively a 'GO AT N' message can be typed by the operator to enter the program's recovery procedure, which must be within the master program.

To permit monitoring of failures on 1904 and up, the following actions occur:-

Magnetic tapes - a count of the total repeats per deck is maintained and typed out when the reel is rewound by modes 512 and 519. A message of the following form is printed out on the console typewriter:-

a) mode 512
CLOSED; UNIT q:- file name, n REPEATS

b) mode 519
UNLOAD; UNIT q:- file name, n REPEATS

Card Punch - Each time a transfer is automatically repeated a message of the following form is printed out:-

S #prog; UNIT q:- ~~RPTD~~ ERR [AL3]

b) Assisted Repeat Devices

1902 & 3

A 'UNIT q ERR' is typed. The operator may adjust the device suitably, e.g. by repositioning the medium (unless the program notes indicate otherwise).

1. When the 'ALLOCATE' button on the device is pressed the transfer will be repeated under executive control. The above cycle can be repeated an unlimited number of times if failure persists. From the program's point of view the transfer is regarded as still in progress until it takes place correctly, and suspension can occur in the usual ways.

2. The operator may type a 'GO AT' message at any time to initiate the program's own recovery procedures, without having pressed the 'ALLOCATE' button to re-engage the device.

3. If the operator types "GO" the program will be re-entered at the point of interruption; Both 'GO' and 'GO AT' discard the order and give the error reply.

1904 and up

A 'UNIT Q R&FX' message is typed out. The operator may adjust the device suitably e.g. by repositioning the medium.

When the 'ALLOCATE' button is pressed the transfer will be repeated under Executive control. This process is repeated up to a fixed number of times, n (dependent on the device). The transfer is regarded as still in progress until it takes place correctly, and suspension of the program may occur in the usual ways.

If the transfer fails n times in succession, a FAIL message is typed, and the transfer made to appear terminated but with B3 of the reply word set to 1 (as for automatic repeat).

c) Non-repeat Devices

1902 & 3

A 'UNIT Q ERR' message is typed. The program is suspended awaiting termination of the transfer.

The operator can type a 'GO AT' message to enter the program's recovery procedure. The order is discarded and B3 of the reply word is set to 1.

If the operator types 'GO' Executive carries on with the program from the point where it ~~was~~ was interrupted. B3 of the reply word is set to 1. [AL/14]

1904 and up

A 'UNIT Q FAIL' message is typed. The transfer is made to appear terminated to the program i.e. the reply word is reset positive, but a fail marker is also set in the reply word (B3=0), together with the store area next address or the unexpired count for those devices where one of these is the standard reply. The program is suspended awaiting operator action, regardless of whether it was previously suspended awaiting termination of the transfer. A 'GO' message causes the program (or sub-program) to continue from the point of suspension.

Alternatively a 'GO AT N' message can by typed by the operator to enter the program's recovery procedure, which must be within the master program.

.3 Reply Word

The second word of the control Area specified in a 157 instruction is used to provide information to the program during execution of the transfer and at completion of the transfer.

For all peripheral transfers, Executive sets B0 to 1 when the transfer is initiated, and then to 0 at completion of the transfer.

When Bit 0 = 0, Bits 1 to 23 have significance as defined in section 4 for the peripheral concerned; when Bit 0 = 1, Bits 1 to 23 are indeterminate.
If the transfer was a failure because of a parity error or a similar condition, then Bit 3 is set to 1.

Contents

.1 Configuration
.2 Parity
.3 Tape Punchings
.4 Internal Representation
.5 I.C.T. 1900 8-track Code
.6 Other Tape Codes

Table A I.C.T. 1900 Standard 8-track code
Table B 7-track Paper-Tape to Internal Code Relationship
Table C 6-track Paper-Tape to Internal Code Relationship
Table D 5-track Paper-Tape to Internal Code Relationship

Paper Tape

A character on the tape is recorded as a row of holes across the tape, each character being represented by a unique combination of holes.

.1 Configuration

Four widths of paper-tape are in use, carrying 8,7,6 or 5 data tracks, along with a feed-hole track. The least significant 3 tracks lie on one side of the feed-hole.

.2 Parity

8-track tape has even parity, i.e. each character has an even number of holes, track 8 being used where necessary to achieve this.

7-track tape has odd parity, i.e.each character has an odd number of holes.

6- and 5-track have no parity track.

.3 Tape Punchings

An 8-track tape, 7 data tracks are available for punching giving 2^7 possible punching arrangements and therefore 128 possible characters.

On 7-track tape, 6 data bits are available for punching, thus providing 64 possible characters.

6-track tape provides 64 possible characters.

5-track tape provides 32 possible characters.

.4 Internal Representation

Internally, a character is represented by 6 bits, allowing 64 possible characters. This range is expanded, by the use of special characters called shift characters, to take advantage of the extra characters available on 8-track tape. The form of conversion between punched representation and internal representation of characters varies according to the width of tape used and the Mode adopted.

.5 I.C.T. 1900 8-track code

The I.C.T. 1900 Standard 8-track paper-tape code is shown in Table A.

Each character on the tape is considered as three 'zone' bits (most significant) and four 'numeric' bits (least significant) whilst the 8th or parity track, is ignored.

e.g. an 8-track paper-tape frame punched -

01000001

Zone Numeric

has a zone value 4 and numeric value 1, and can be represented by 4/1.

For input and output purposes there are two modes of conversion between the 6-bit internal character and the 7-bit paper-tape character.

a) Shift Set

Shift characters are used in the following manner to qualify the characters which succeed them in the core store.

i) An alpha or beta character qualifies all succeeding characters in the store, up to and excluding the next alpha or beta character, and excluding any delta character and its immediate successor.

ii) A delta shift character qualifies for only the character which immediately succeeds it.

code	octal value	name
3/12	74	alpha shift (α)
3/13	75	beta shift (β)
3/14	76	delta shift (δ)

input

As each paper-tape character is read, the parity bit is removed and the character establishes in the reader the 'shift status' corresponding to the shift group to which the character belong. The delta shift status applies only to the character which establishes it, the previously established alpha - or beta - shift status still applicable for subsequent characters. A character shown as belonging to both alpha - and beta - shift groups cause no change in the shift status. Any shift character needed is generated by hardware and transferred to the store immediately prior to the converted character.

TABLE A

I.C.T. 1900 PAPER TAPE CODE (8-TRACK)

TAPE CHARACTER		INTERNAL VALUE			SYMBOL
PUNCHING	VALUE	SHIFT SET		GRAPHIC SET	
		SHIFT	OCTAL	OCTAL	
•	0/0		80		TC0 NULL
• •	0/1		81		TC1 SOH
• •	0/2		82		TC2 STX
• ••	0/3		83		TC3 ETX
• •	0/4		84		TC4 EOT
• • •	0/5		85		TC5 ENQ
• •••	0/6		86		TC6 ACK
• ••••	0/7		87		BELL
• • •	0/8		90		FE0 BS
• • •	0/9		91		FE1 HT
• • •	0/10		92		FE2 NL
• • • ••	0/11		93		FE3 VT
• • •••	0/12		94		FE4 FF
• • • •	0/13		95		FE5
• • • •••	0/14		96		SHIFT OUT
• • ••••	0/15	δ	97	UNASSIGNED	SHIFT IN
• •	1/0		00		TC7 DLE
• • •	1/1		01		DC1
• • •	1/2		02		DC2
• • • ••	1/3		03		DC3
• • • •	1/4		04		DC4 STOP
• • • • •	1/5		05		TC8 NACK
• • • ••	1/6		06		TC9 SYNC
• • • •••	1/7		07		TC10 ETB
• •• •	1/8		10		CANCEL
• •• • •	1/9		11		EM
• •• • •	1/10		12		SS
• •• • ••	1/11		13		ESC
• •• • •	1/12		14		IS 4 FS
• •• • ••	1/13		15		IS 3 GS
• •• • •••	1/14		16		IS 2 RS
• •• • ••••	1/15		17		IS 1 US
• • •	2/0		20	20	space
• • • •	2/1		21	21	!
• • • •	2/2		22	22	"
• • • ••	2/3		23	23	#
• • • •	2/4		24	24	£
• • • • •	2/5		25	25	%
• • • ••	2/6		26	26	&
• • • •••	2/7		27	27	'
• • • •	2/8		30	30	(
• • • • •	2/9		31	31	)
• • • • •	2/10		32	32	*
• • • • ••	2/11		33	33	+
• • • • •	2/12		34	34	,
• • • • ••	2/13		35	35	−
• • • •••	2/14		36	36	.
• • • ••••	2/15	α/β	37	37	/
• •• •	3/0		00	00	0
• •• • •	3/1		01	01	1
• •• • •	3/2		02	02	2
• •• • ••	3/3		03	03	3
• •• • •	3/4		04	04	4
• •• • ••	3/5		05	05	5
• •• • •••	3/6		06	06	6
• •• • ••••	3/7		07	07	7
• ••• •	3/8		10	10	8
• ••• • •	3/9		11	11	9
• ••• • •	3/10		12	12	:
• ••• • ••	3/11		13	13	;
• ••• • •••	3/12		14	14	<
• ••• • ••••	3/13		15	15	=
• •••• •	3/14		16	16	>
• •••• • •	3/15		17	17	?

TAPE CHARACTER		INTERNAL VALUE			SYMBOL
PUNCHING	VALUE	SHIFT SET		GRAPHIC SET	
		SHIFT	OCTAL	OCTAL	
•• •	4/0		40	40	@
• • •	4/1		41	41	A
• • •	4/2		42	42	B
•• • ••	4/3		43	43	C
• • •	4/4		44	44	D
•• • • •	4/5		45	45	E
•• • ••	4/6		46	46	F
• • ••••	4/7		47	47	G
• •• •	4/8		50	50	H
•• •• •	4/9		51	51	I
•• •• •	4/10		52	52	J
• •• • ••	4/11		53	53	K
•• •• •	4/12		54	54	L
• •• • • •	4/13	α	55	55	M
• •• • ••	4/14		56	56	N
•• •• • ••••	4/15		57	57	O
• • •	5/0		60	60	P
•• • • •	5/1		61	61	Q
•• • • •	5/2		62	62	R
• • • ••	5/3		63	63	S
•• • • •	5/4		64	64	T
• • • • •	5/5		65	65	U
• • • ••	5/6		66	66	V
•• • • ••••	5/7		67	67	W
•• •• •	5/8		70	70	X
• • •• • •	5/9		71	71	Y
• • •• •	5/10		72	72	Z
•• •• • ••	5/11		73	73	[
• • •• •	5/12		64	74	∅
•• •• • • •	5/13	δ	65	75	]
•• •• • ••	5/14		66	76	↑
• • •• • ••••	5/15		67	77	←
•• •	6/0		40	40	_ (ul)
••• • •	6/1		41	41	a
••• • •	6/2		42	42	b
•• • ••	6/3		43	43	c
••• • •	6/4		44	44	d
•• • • •	6/5		45	45	e
•• • ••	6/6		46	46	f
••• • ••••	6/7		47	47	g
••• • •	6/8		50	50	h
•• •• •	6/9		51	51	i
•• •• •	6/10		52	52	j
••• • • ••	6/11		53	53	k
•• • • •	6/12		54	54	l
••• • • • •	6/13	β	55	55	m
••• • • ••	6/14		56	56	n
•• • • ••••	6/15		57	57	o
•••• •	7/0		60	60	p
••• • •	7/1		61	61	q
••• • •	7/2		62	62	r
•••• • ••	7/3		63	63	s
••• • •	7/4		64	64	t
•••• • • •	7/5		65	65	u
•••• • ••	7/6		66	66	v
••• • ••••	7/7		67	67	w
•••• •	7/8		70	70	x
••••• • •	7/9		71	71	y
••••• • •	7/10		72	72	z
••••• • ••	7/11		73	73	③₇
•••• • •	7/12		70	74	③₈
•••• • • •	7/13	δ	71	75	③₉
•••• • ••	7/14		72	76	③₆
••••• • •••	7/15		73	unassigned	Del.

IN SHIFT SET INTERNAL CODE THE FOLLOWING DEFINITIONS EXIST:

74 ALPHA SHIFT
75 BETA SHIFT
76 DELTA SHIFT
77 UNASSIGNED IN ALL SHIFTS
40–63 UNASSIGNED IN δ SHIFT

UNASSIGNED INTERNAL CODES TRANSMITTED TO A PUNCH IN SHIFT SET MODE MAY GIVE RISE TO NO PUNCHINGS OR TO ERRONEOUS PUNCHINGS.

INTERNAL CODES 40–77 TRANSMITTED TO A PUNCH IN GRAPHIC SET MODE GIVE RISE TO PUNCHINGS 4/0 TO 5/15 RESPECTIVELY. IN THIS MODE, PUNCHINGS 0/0–1/15 AND 6/0–7/15 CANNOT BE PRODUCED.

③ = NATIONAL CHARACTERS

TABLE B

7-track Paper-tape to Internal Code Relationship

Track No. 4 3 2 1	7- 6-	0 0	0 1	1 0	1 1	'Numeric' Value
0 0 0 0		00	20	40	60	0
0 0 0 1		21	01	61	41	1
0 0 1 0		22	02	62	42	2
0 0 1 1		03	23	43	63	3
0 1 0 0		24	04	64	44	4
0 1 0 1		05	25	45	65	5
0 1 1 0		06	26	46	66	6
0 1 1 1		27	07	67	47	7
1 0 0 0		30	10	70	50	8
1 0 0 1		11	31	51	71	9
1 0 1 0		12	32	52	72	10
1 0 1 1		33	13	73	53	11
1 1 0 0		14	34	54	74	12
1 1 0 1		35	15	75	55	13
1 1 1 0		36	16	76	56	14
1 1 1 1		17	37	57	77	15
	'Zone' Value --	0	1	2	3	

Notes:
a. The Internal Codes are shown in OCTAL
b. In the Paper Tape Codes, a '1' represents a hole in the tape.
c. The 'Stop Character' is that code which is represented Internally by *22.
d. Track 5 is punched so as to produce Odd-Parity.
e. A frame (character) on Paper Tape with all seven tracks unpunched will cause a Parity Error on reading.

TABLE C

6-track Paper-tape to Internal Code Relationship

Track No. 4 3 2 1	6- 5-	0 0	0 1	1 0	1 1	'Numeric' Value
0 0 0 0		00	20	40	60	0
0 0 0 1		01	21	41	61	1
0 0 1 0		02	22	42	62	2
0 0 1 1		03	23	43	63	3
0 1 0 0		04	24	44	64	4
0 1 0 1		05	25	45	65	5
0 1 1 0		06	26	46	66	6
0 1 1 1		07	27	47	67	7
1 0 0 0		10	30	50	70	8
1 0 0 1		11	31	51	71	9
1 0 1 0		12	32	52	72	10
1 0 1 1		13	33	53	73	11
1 1 0 0		14	34	54	74	12
1 1 0 1		15	35	55	75	13
1 1 1 0		16	36	56	76	14
1 1 1 1		17	37	57	77	15
'Zone' Value -		0	1	2	3	

Notes: a. The Internal Codes are shown in OCTAL

b. In the Paper Tape Codes, a '1' represents a hole in the tape.

c. The 'Stop Character' is that code which is represented Internally by *02.

Part 2

TABLE D

5-track Paper-tape to Internal Code Relationship

Track No. 5 4 3 2 1	Internal Code	Track No. 5 4 3 2 1	Internal Code
0 0 0 0 0	20	1 0 0 0 0	00
0 0 0 0 1	21	1 0 0 0 1	01
0 0 0 1 0	22	1 0 0 1 0	02
0 0 0 1 1	23	1 0 0 1 1	03
0 0 1 0 0	24	1 0 1 0 0	04
0 0 1 0 1	25	1 0 1 0 1	05
0 0 1 1 0	26	1 0 1 1 0	06
0 0 1 1 1	27	1 0 1 1 1	07
0 1 0 0 0	30	1 1 0 0 0	10
0 1 0 0 1	31	1 1 0 0 1	11
0 1 0 1 0	32	1 1 0 1 0	12
0 1 0 1 1	33	1 1 0 1 1	13
0 1 1 0 0	34	1 1 1 0 0	14
0 1 1 0 1	35	1 1 1 0 1	15
0 1 1 1 0	36	1 1 1 1 0	16
0 1 1 1 1	37	1 1 1 1 1	17

Notes: a. The Internal Codes are shown in OCTAL

b. In the Paper Tape Codes, a '1' represents a hole in the tape.

Contents

.1 General

.2 Summary of Applicable Instructions

.3 Control Area Details

.4 Mode Definitions
- a) 8-track
  - i) Character Set Modes
  - ii) Starting Shift Modes
  - iii) Stopping Modes
  - iv) End of Tape Character
- b) other tracks
  - i) Stopping Modes

.5 Peripheral Incidents

.6 **Property Code** } **AL 6**

.7 **Reading Olivetti tape** }

1 General

The two types of paper-tape reader used on the 1900 series are the 1915 operating at a maximum input rate of 300 characters per second, and the 1916 operating at a maximum of 1000 ch./sec. As far as the programmer is concerned, the operation of the two models is identical.

Of each of these types of reader there are two versions: the 1915/1 and 1916/1 are supplied with 1909 and early 1904/5 processors; and 1915/2 and 1916/2 are supplied with all other processors. Almost identical facilities are available on the /1 and /2 versions, and differences are mentioned where necessary.

Three widths of tape may be used:

1" carrying 8 or 7 data tracks
7/8" carrying 7 or 6 data tracks
11/16" carrying 5 data tracks

8-track is normally used, but the readers can be set up, by means of a switch on /1 versions, or exchangeable wiring Boards on /2 versions, to deal with other track widths.

6-track tape cannot be read on /1 versions.

On /2 versions some other alternative facilities can be made available by using special Boards - this manual only describes the Range Compatible facilities.

Facilities for reading Olivetti paper tape are to be found on page 14.

.2 Summary of Applicable Instructions

150 Suspend program if reader is busy.
151 Release reader from assignation.
152 Disengage reader.
154 Load more program from reader (in 8-track standard code only).
156 Assign a reader to program.
157 Activate data transfer from reader.

The general definitions of these instructions are to be found in Part 1 of the manual.

.3 Control Area Details

1st Word (type and mode)

bits 0-8 Type = 0

bits 9-23 Mode (see also 4.1 page 5)

8-track

B18 0 (Only permissible state)
B19 0 Read Normal (ignoring NULL, DELETE)
1 Read all characters
B20 0 Read Shift Set
1 Read Graphic Set
B21 0 Start in α Shift
1 Start in previous Shift
B22 0 Stop by count (in 3rd word)
1 Stop by Stop-character (subject to count)
B23 0 (Only permissible state)

5-, 6-, and 7-track

B18 0 (Only permissible state)
B19 1 (Only permissible state)
B20 1 (Only permissible state)
B21 0 (Only permissible state)
B22 0 Stop by Count (In 3rd Word) (only permissible state for 5-trade)
1 Stop by Stop-character (subject to count)
B23 0 (Only permissible state)

2nd Word (reply word)

bit 0	1	Transfer in progress
bit 3	1	Error in transfer
bits 17-23		Remainder Count

3rd Word (no. of characters to be transferred)

Maximum number of internal code 6-bit characters to be transferred to store; cannot exceed 128.

4th Word (Start address)

Character address of position in store to receive first character of transfer.

.4 Mode Definitions

For information on conversion from internal to paper-tape codes, refer to Section 4.0.

a) 8-track modes

Table of Mode Values corresponding to Bit States

Mode Value		Internal Code	Start in	Method of
Octal	Bits	Character Set	Shift	Stopping Transfer
00	000000	SHIFT, NORMAL	alpha	COUNT
02	000010	" "	"	STOP-CHARACTER
04	000100	" "	previous	COUNT
06	000110	" "	"	STOP-CHARACTER
20	010000	SHIFT, ALL CHARS	alpha	COUNT
22	010010	" " "	"	STOP CHARACTER
24	010100	" " "	previous	COUNT
26	010110	" " "	"	STOP CHARACTER
10	001000	GRAPHIC	N/A	COUNT
12	001010	"	"	STOP CHARACTER

Other mode values are not defined and may give rise to undesirable results (which may vary between /1 and /2).

i) Character Set Modes

SHIFT SET, NORMAL

Characters are converted and shift characters inserted where appropriate except that the characters Null (0/0) and Delete (7/15) are ignored, i.e. no corresponding internal character is transferred.
In this mode the TC4 character (0/4) may cause the Reader to Become disengaged.

SHIFT SET, ALL CHARACTERS

1900 PDRM/IAB
AL1

All 128 paper tape characters are converted, shift characters inserted where necessary.

GRAPHIC SET

Characters are either converted or not transferred, Referring to Table A, characters 0/0 to 1/15 (and 7/15) are ignored.

ii) Starting Shift Modes

START IN ALPHA SHIFT

Alpha-shift status is established in the Reader just prior to the start of the data transfer.

If the last character transferred to the store during the previous data transfer was a shift character, then the converted paper tape character which would otherwise have followed that shift character in the store will be lost.in the case of /2 Readers, but in the case of /1 Readers will be transferred in the current transfer, preceded if necessary by a beta or delta shift character.

START IN PREVIOUS SHIFT

The shift status established in the Reader during the previous transfer is retained, and is applicable at the start of the current transfer.

If the last character transferred to the store during the previous data transfer was a shift character, then the converted paper-tape character which would otherwise have followed that shift character in the store will appear as the first internal character of the current data transfer. This rule still holds if the paper tape character is one which is ignored in 'Shift Normal' but which was not ignored (because the previous transfer specified 'Shift, All Characters') and the current transfer specifies 'Shift, Normal.'

Note

The first transfer instruction issued to a Reader in a program must specify α shift.
When using Graphic Set, the starting mode must be α.
If a program changes from Graphic Set to Shift Set, then the starting shift must be α.
If a program calls for repositioning of the tape, including loading a new tape, then α shift must be specified.

iii) Stopping Modes

STOP BY COUNT

The transfer terminates when the number of internal code characters transferred to the store, including shift characters, is equal to the number specified in the 3rd word of the control area of the 157 instruction. The number of paper-tape characters actually passed under the reading head may be greater than the specified count if some characters are not transferred, or it may be less than the specified count if some shift characters are transferred.

If the Shift Set internal code was specified for the transfer, the last character transferred to the store may be a shift character; the converted paper-tape character itself is retained in the Reader, and is available at the start of the next transfer.

STOP BY STOP-CHARACTER

The transfer terminates when any of the stop-characters defined below are detected on the paper tape, or if the conditions for Stop-by-count are satisfied, whichever occurs first. The stop characters recognised are dependent upon the internal code specified for the transfer.

Shift Set

The stop characters recognised are FE2 (Newline). FE3, FE4 and FE5. The converted stop character and its preceding delta shift character are transferred to the store, except that if the transfer of the delta shift character causes the conditions for Stop-by-count to be satisfied then the converted stop character itself is not transferred, but is available at the start of the next transfer. If, in the latter case 'Start in Previous Shift, Stop by Stop-character' is specified for the next transfer, then one character only (the converted stop-character) is transferred to the store, and the transfer then terminates, regardless of the mode of stopping specified for the first transfer.

Graphic Set

The stop characters recognised are values 0/1 to 0/15. These characters are not transferred to the store.

Note

If the first character of the transfer is a stop character and the Count was 128, the Remainder Count will be zero.

iv) End of Tape Character

Regardless of the mode of stopping specified for the transfer, the transfer is terminated and the Reader disengaged if:

1. the internal code specified is Shift Set Normal, and
2. a TC4 character is detected on the paper tape, and
3. after transfer of the delta-shift character and the converted TC4 character to the store, the number of internal code character transferred in the current data transfer is less than the number specified in the 3rd Word of the Control Area.

Note:- that if the number of internal code characters transferred becomes equal to the number specified in the 3rd Word of the control area, when either

1. the converted TC4 character is transferred, or
2. the associated delta-shift character is transferred (causing the converted TC4 character to be retained for the next transfer),

then that specific TC4 character never causes the Reader to become disengaged.

b) Modes for 5; 6 and 7-track codes

Table of Mode Values corresponding to Bit States.

Octal	Binary	Meaning
30	011000	Stop on count
32	011010	Read to stop character (subject to Count) (not 5-track)

Other mode values are not defined and may give rise to undesirable results (which may vary between /1 and /2 versions).

i) Stopping Modes

STOP BY COUNT

The transfer terminates when the number of internal code characters transferred into the store is equal to the number specified in the 3rd word of the control area of that 157 instruction.

STOP BY STOP-CHARACTER (not 5-track)

The transfer terminates when the stop character is detected on the tape, or if the Stop-by-Count condition is satisfied, whichever occurs first. The stop character is that code which consists of one hole (in track 2) only, and is always read into the store.

.5 Peripheral Incidents

The action in the event of a machine fault is described in section 3, page 1. (See also below)

In the case of a parity failure, the Tape Reader is treated as for Assisted Repeat Devices (Section 3 page 4).

/1 Readers

When the Reader becomes disengaged as a result of an operator pressing the 'Off' button whilst a transfer is in progress, then the current transfer remains active and will run to completion when the operator presses the 'On' button.

This difference may affect the organization of the data available to the program if the operator repositions the tape under the reading head when the Reader is disengaged.

.6 Property Code

(see also Section 5 ~~page 2~~ page 1)

[AL/14]

Structure

B10	:	free
B11-12	:	speed
B13-17	:	free

Code

a)	1915	:	00000000
b)	1916	:	00100000

7 . Reading OLIVETTI Paper Tape

It is possible, on certain readers, to have special facilities for the reading of Olivetti Paper Tape. Where such facilities are available the following details are applicable.

Olivetti paper tape is 20.5 mm in width and has up to 6 data tracks punched. The perforations are square in shape and there is no sprocket track, clocking of the data frames being by a "logical or" of the data holes.

Due to the nature of the clocking system there is no character corresponding to 'null' (no data holes), and the reading modes available to the programmer are as follows:

*30 - Read Graphic Set, All Characters, Stopping on Count.
*32 - Read Graphic Set, All Characters, Stopping on Stop Character.

The paper tape to 1900 internal code relationship is straight "hole = 1" and the normal 6-track code table (in the PDRM) may be used, remembering that internal code *00 can never appear.

Contents

.1 General
.2 Summary of Applicable Instructions
.3 Control Area Details
.4 Mode Definitions
.5 Peripherals Incidents
.6 Property Code AL 6

.1 General

The 1925 Paper-tape Punch operates at a maximum output rate of 110 character per second. There are two versions of the Punch: the 1925/1 is supplied with the 1909 and early 1904/5 Processors; the 1925/2 is supplied with all other Processors.

Almost identical facilities are available on both versions, and any differences in operation will be mentioned where necessary. The Punch can be set up by means of a switch in the case of /1 version, and Exchangeable Wiring Boards in /2 versions, to translate the internal code to 8-,7-,6- or 5-track tape punchings. (/1 versions cannot punch 6-track)

As standard, the unit will translate information from the 1900 series internal machine code into ICT 8-track (7 data-bit) code. For 5-,6-,7- or other 8-track codes, special routines must be used to convert the data and arrange it in appropriate form in the output area. Conversion tables for the codes are to be found at the end of Section 4.0.

.2 Summary of Applicable Instructions

150 Suspend program if Punch is busy.
150 Release Punch from Assignation.
152 Disengage Punch.
155 Suspend program and dump onto paper tape (8-track only)
156 Assign a Punch to program.
157 Activate data transfer to Punch

The general definitions of these instructions are to be found in Part 1 of the manual.

.3 Control Area Details

1st Word (type and mode)

bits 0 - 8 Type = 1

bits 9 - 23 Mode

8-track

B18	0	(Only permissible state)
B19	0	Punch Data
	1	Punch blanks
B20	0	Shift set Mode
	1	Graphic Set Mode
B21	0	Start assuming α shift
	1	Start assuming previous shift
B22	0	(Only permissible state)
B23	0	(Only permissible state)

5-,6-,7-track

B18	0	(Only permissible state)
B19	0	Punch Data
	1	Punch blanks
B20	1	(Only permissible state)
B21	0	(Only permissible state)
B22	0	(Only permissible state)
B23	0	(Only permissible state)

2nd Word (reply word)

bit	0 1	Transfer in progress
bit	3 1	Error in transfer
bits	17-23	Remainder count if B3 = 1
		Zero if B3 = 0

3rd Word (no. of characters to be transferred)

Number of internal 6-bit characters to be output to Punch. Maximum = 128.

4th Word (Start address)

Character address of position in store of first character to be output.

.4 Mode Definitions

For information on conversion from paper-tape codes to internal codes, refer to Section 4.0.

a) 8-track modes

Table of Mode Values corresponding to Bit States

Mode Value		Meaning
octal	bits	
00	000000	Punch Data in Shift Set, start assuming ✱ Shift.
04	000100	Punch Data in Shift Set, start assuming previous shift.
10	001000	Punch Data in Graphic Set.
20	010000	Punch blanks.

Other mode values are not defined and may give rise to undesirable results (which may vary between /1 and /2 versions).

i) Character Set Modes

PUNCH SHIFT SET

All tape characters may be punched in this mode by the transfer of the corresponding shift and other internal code characters.

Unassigned internal code characters must not appear in the date to be transferred.

PUNCH GRAPHIC SET

Tape characters 2/0 - 5/15 only may be punched in this mode by transfer of the corresponding internal code characters.

START ASSUMINGα SHIFT

This mode is to be used where the total numbers of characters to be transferred is less than 128, or for the first transfer instructions in the case of the total no. characters to be transferred exceeding 128.

START ASSUMING PREVIOUS SHIFT

This mode is to be used for transfers which continue on from a previous transfer instruction in cases where the total no. characters to be transferred exceeds 128.

PUNCH BLANKS

The 3rd Word of the control area specifies the number of blank characters to be punched. The 4th Word of the control area may contain any address such that the total area to be transferred is within the programm's limit. The actual contents of this are transferred but ignored by the Punch.

b) 5-. 6- and 7-track modes

Table of Mode Values corresponding to Bit States

Octal	Binary	Meaning
10	001000	Punch Data
30	011000	Punch Blanks

Other modes are not defined and may give rise to undesirable results (which may vary between /1 and /2 versions).

PUNCH DATA

The 6-bit internal characters are converted as determined by the wiring of the Exchangeable Wiring Board. The conversion tables for 5-, 6- and 7-track tape punching are to be found at the end of Section 4.0 (Tables B, C and D respectively).

PUNCH BLANKS

This mode is as defined under 'a) 8-track' above.
On the 1925/1 Punch blank characters cannot be produced. Instead, Punch Blanks mode produces the character represented internally by octal 27.

NOTE

1. If Blanks are punched on 7-track tape and subsequently read, errors will occur.
2. When punching 5-track tape, the 2nd bit of the internal character must always be zero, otherwise mispunching may result.

.5 Peripheral Incidents

The action in the event of a machine fault is described in Section 3, page 1. (see also below)

In the case of an error in the transfer the Tape Punch is treated as for Non-repeated device (Section 3, pages 5,6)

1925/1 version

When the Punch becomes disengaged as a result of an operator pressing the 'off' button whilst a transfer is in progress, the the Punch stops operating at once, but the transfer remains active and will run to completion when the Operator presses the 'on' button.

.6 Property Code

[AL1 14]

(see also Section 5 page 1/2)

Structure

B10	:	free
B11-12	:	speed
B13-17	:	free

Code

1925	:	00000000

contents

.1 General

.2 Summary of Applicable Instructions

.3 Control Area Details

.4 Control Characters

.5 Peripheral Incidents

.6 Property Code AL 6

.1 General

Five types of printer are available with the 1900 series. A brief description of each is given below in tabular form:

Feature	Printer Type					
	~~1903~~ 1930	1931	1932	1933	2401	[AL/12]
max.output rate	1000 l.p.m.	300 l.p.m.	600 l.p.m.	1350 l.p.m.	300 l.p.m.	
no. of print positions	120	/1 = 96 /2 =120	/1 = 96 /2 =120	/1 = 96 /2 =120 /3 =160	/1 = 96 /2 =120	
line spacing	6 l.p.i.	6/~~8~~ l.p.i.	6/~~8~~ l.p.i.	6/8 l.p.i.	6 l.p.i.	AL 6
format loop	non-standard	standard	standard	standard	only 3 channels	
'automatic write'	no	no	no	yes	no	
others	does not indicate 'overflow' in reply word.				does not indicate 'overflow' in reply word.	

Except where indicated, the following information applies to all types of printer listed above.

The printer has a repetoire of 64 characters (see table on next page).

Multipart stationery may be used as follows:

up to 6 parts with interleaved carbon
up to 10 parts with NCR paper
width 4" to 18"

The printer has a reading head which senses holes punched in a tape loop and transmits signals to the control electronics of the printer. These signals are used to control 'paper throw' operations so that a program may advance the printing stationery to pre-determined positions. There are 8 channels (1-8) available, channel 1 being used for the top-of-form and channel 8 being used for end-of-form. Instructions on the preparation of this loop are to be found in 'Systems Manual' Vol.2 (Technical Publications 3338.

The 1930 and 2401 Printers have non-standard format control mechanisms.

On 1933 Printers only, an 'Automatic Write' mode is available whereby printing of more than one line may be initiated by a single 157 instruction.

I.C.T. 64 character code

least significant 3 bits	Most significant 3 bits of internal 6-bit character							
	0	1	2	3	4	5	6	7
0	0	8	space	(	@	H	P	X
1	1	9	!	)	A	I	Q	X
2	2	:	"	*	B	J	R	Z
3	3	;	#	+	C	K	S	[
4	4	<	£	,	D	L	T	∅
5	5	=	%	-	E	M	U	]
6	6	>	&	.	F	N	V	↑
7	7	?	'	/	G	O	W	←

.2 Summary of Applicable Instructions

150 Suspend program if Printer is busy

151 Release Printer from assignation and disengage it

152 Disengage Printer

156 Assign a Printer to program

157 Activate data transfer to Printer

The general definitions of these instructions are to be found in Part 1 of the manual.

.3 Control Area Details

1st Word (type and mode)

bits 0 - 8 Type = 2

bits 9 - 23 Mode

0 Print Normal (advance paper and/or print one item)

2 Automatic Write (1933 only)

2nd Word (reply word)

bit 0 1 Transfer in progress

bit 1 1 Overflow (end-of-form position on format tape)

bit 2 1 Paper low

bit 3 1 Error in transfer

bits 4 - 23 Indeterminate

3rd Word (no. of characters)

Mode 0

For 1933 printers only, the count is the number of characters to be printed, plus one for the paper feed control character. The maximum number of characters is limited by the number of print positions available on the printer. If the count is greater than the number of print positions, then the excess characters are ignored.

For all other printers, the count must be one more than the number of print positions available on the printer. If the count is greater than the number of print positions, then the excess characters are ignored.

A zero count in the case of 1931 or 1932 printers will produce an 'illegal instruction' suspension, whilst a count of 1 will produce a 'no print' even if the paper feed control character indicates that printing is required.

If the program must use a printer, the details of which are not known, then 120 print positions should be assumed (providing spaces for unused positions 96 to 119), in which case the maximum count is 121.

Mode 2

This mode is only available on the 1933 Printer. The value of the count is the number of printable and control characters to be sent to the printer. There is no maximum value except that implied by the program's limit address in the store.

4th Word (start address)

A character address specifying the start address of the output area in store. In the case of Mode 0 it must be the address of character 3 of a word.

For both the 96 and the 120 character position unbuffered printer, the character address must be the first word of an area of 31 words wholly in store. Although in the case of the 96 position printer, the contents of the last six words will not be printed, the complete area must exist although these last six words may be used by the program for other purposes.

.4 Control Characters

a) Print Normal Mode

The information to be printed as a single line is arranged as a consecutive number of six-bit characters in core store. This area must be immediately preceded by a word whose least significant six bits (character position 3) contain a control character which specifies the paper movement required. The control character regarded as an octal pair, has values:-

No Print	Print	Type of Movement
00	40	No movement.
01	41	Single line advance
02	42	Double line advance
11	51	Form feed (to channel 1)
12-17	52-57	Feed to next hole in channel 2-7

Note

Paper advance occurs before printing.

b) Automatic Write Mode (1933 only)

The data to be printed is arranged in an output area in a series of six-bit character, and control characters both precede and are interspersed among the data to specify the format requirements.

character	octal	meaning
Escape ←	77	Next character is a control character, except that if the following character is another # 77, treat the second as a data character and print it
Printout ↑	76	Print out whatever is in the buffer and load the next data character into position 1 of the buffer unless otherwise instructed. *
Vertical Paper Movement	00 01 02 11-17	No movement Feed one line Feed two lines Throw to Channel 1 - 7
Horizontal Tabulation	54/55 40-51 60-71	Hundreds (0 or 1) Buffer position Tens (0 - 9) of next character Units (0 - 9) to be printed **

* Note 1. The Printout character is not required when the extreme right-hand location of the buffer is filled; however with /1 and /2 printers it is advantageous to employ the function of this character for compatibility with printers having more print positions.

** Note 2 Addresses of buffer positions are 0 - 95 (96 print positions) A4/11
0 - 119 (120 print positions)
0 - 159 (160 print positions)

.5 Peripheral Incidents

a) Machine Fault

The action in the event of a machine fault is described in Section 3, page 1.

possible causes

i. No paper in stacker feed rollers

ii. Stacker nearly full

iii. No paper in mechanism

iv. Incorrect paper tension between feed tractors

v. Paper torn between upper and lower feed tractors

vi. No ink ribbon

vii. Lack of ribbon tension

viii. Ribbon torn

b) Error

In the case of an error occurring during the transfer, the Printer is treated as for Non-repeat devices, Section 3, pages 5, 6.

c) 'Paper Low'

The paper low condition arises when the last form leaves the hopper. Warning to the program is given by means of setting B2 = 1 in the Reply Word. If printing continues, eventually fault condition iii (see above) occurs.

d) 'Overflow'

The following does not apply to the 1930 ~~and 2401~~ Printers. ~~AL 7~~

i) Mode 0

When the hole in Channel 8 (overflow hole, or end-of-form) is sensed during a transfer utilizing paper control characters #01, #02, #41 or #42, then the Reply Word is set B1 = 1 for the next transfer that utilises any of these control characters, unless a transfer utilising paper control characters #11-17 or #51-57 occurs first. Once the overflow reply has appeared, it will continue to appear for all transfers utilising paper control characters #00, #01, #02, #40 or #42, until a transfer utilising paper control characters #11-17 or #51-57 takes place, when the overflow condition will be cleared.

The overflow Reply will never be set in the Reply to a transfer utilizing control characters #11-17 or #51-57 AL 6

If the first control character after the hole is sensed is a 'no movement' character (#00, #04) the overflow reply will not be set then. If a control character #01, #02, #41 or #42 then operates before another paper throw character (#11-17 or #51-57 then the overflow reply will be set.

ii) Mode 2

The overflow reply is the same as that for the equivalent Mode o transfers; see above.

.6 Property Code

(see also Section 5 page 1 2)

Structure

B10 : ~~free~~ undefined

B11-12 : speed

B13-14 : Line width

B15 : automatic write availability

B16 : ~~free~~ undefined

B17 : 'overflow' ~~indication availability~~
Full format loop (7 tracks and overflow) AL 6

Code

cat. number \ no. print positions	96	120	160
1930	-	01101000	-
1931	00000001	00001001	-
1932	00100001	00101001	-
1933	01100101	01101101	01111101
2401	00000000	00000000	-

CONTENTS

.1. General

.2. I.C.T. 64-Character Card Code

.3. Other Codes
- .1 1500 Card Code
- .2 Card Image
- .3 Decoder B
- .4 Decoder E
- .5 Decoder G

.4. Code Relationships
- .1 1911, 1912/1, 1912/9, 2707 Card Readers
- .2 2102 and P.F. 311 Card Readers
- .3 2103 Card Reader
- .4 1004 Card Reader
- .5 Card Punches.

.1 General

Standard I.C.T. cards are $7\frac{3}{4}$" long and $3\frac{1}{4}$" wide. Along the card are 80 columns, each comprising 12 punching positions, or rows, across the card. The punching positions are identified as 10, 11, 0-9, the 10 position being nearest the top of the card with column 1 to the left. They are punched with rectangular holes.

.2 I.C.T. 64-Character Card Code

In this mode of representation, each of the 80 columns on a card represents a 6-bit internal computer character which is specified in the I.C.T. 64, 6-bit Character code. On a card, a character is represented by a combination of 1, 2 or 3 punchings in the 12 row positions in a column.

The "upper-curtate" position of the card contains the row positions numbered 10, 11, 0 and the "lower-curtate" position contains row positions 1-9.

A relationship between the punching and 6-bit internal value exists and this conversion is carried out in the process of reading or punching by the electronics of the peripheral unit concerned. A further relationship exists between the 6-bit internal value and the I.C.T. 64-character code, and this conversion is carried out by the control unit hardware.

The I.C.T. 64, 6-bit character code is shown in table in sub-Section .4.

.3 Other Codes

Facilities are provided for reading codes different from the 1900 card code.

.3.1 1500 Card Code

I.C.T. 1500 card code can be punched and read, as it is almost a subset of the 1900 card code. However, punching combinations with both the 7 and 8 rows punched are not used, and, more important, the punching combinations "10" and "10-0" are indistinguishable.

.3.2 Card Image

Some card readers have a "card image" mode of reading in which each column is considered as representing two 6-bit characters, the first formed by rows 10, 11, 0, 1, 2, 3, the second by rows 4, 5, 6, 7, 8, 9. Thus the punching of one 80-column card is equivalent to 160 internal 6-bit characters.

.3.3 Decoder B

Some card readers have a decoder type B, capable of decoding the I.C.T. 1900 card code and the I.C.T. 1500 card code, (except that the punching combinations "10" and "10-0" will be indistinguishable). The codes transmitted to the processor are the correct 1900 internal codes for the characters represented by the punchings, and therefore no further code translation is required.

.3.4 Decoder E

Some card readers have a Decoder type E with a switch. According to the setting (code 1/code 2) of the switch, certain punchings are paired together differently in assignment to the internal codes, thus permitting unambiguous recognition of various card codes when reading.

Within Executive are further code conversion features so that when cards with standard 1900 punchings are to be read with the decoder switch in the "code 1" position, conversion between zone/numeric internal code and 1900 standard internal code may also take place.

The card codes which can be decoded uniquely are as follows:-

Code 1: I.C.T. 1900; and, except as stated above (.3.1), 1500; the "dense subset" of the I.S.O. card code.

Code 2: I.C.T. 1004 and 1300; I.B.M. 1401. The I.C.T. 1300 card code facility permits the reading of 3-zone, 4-zone and 5-zone codes.

The various groups of codes do not use the same punching combinations, so it is not a practical proposition to read packs of mixed cards.

.3.4 cont'd

The codes transmitted by the reader do not correspond with 1900 internal code for the characters punched, except in the case of I.S.O. "dense subset"; hence translation of codes is required in the majority of cases. Executive includes code translation facilities, which perform the translation required when using the 1900 card code, but any other translation required is the responsibility of the object program. Whilst the use of Executive's translation facility may simplify the coding required in the object program, maximum efficiency will be obtained only if all the translation is performed by the object program.

Conversion tables for these codes are to be found in sub Section 4.4 pages 5 to 19.

.3.5 Decoder G

This decoder is switchable, by the operator, to one of two states known as "Code 1" and "Code 2". The Card Image feature is available with this decoder, and its operation is not affected by the state of the code selection switch.

The "Code 1" state provides decoding of the I.C.T. 1900 and 1500 card codes and the ISO dense subset, code relationships appearing on page 8, internal codes being those under "Mode 0" (Translated).

The "Code 2" state provides decoding of the 1004; I.C.T. 1300; and IBM 1401 card codes, code relationships appearing on page 8, internal codes being those under "Mode 0" (Translated).

Decoder G differs from Decoder E in that the latter does not have a Card Image feature, and also translates data read into 1900 internal code before giving the object program a "transfer completed" indication, thus providing object program compatibility.

.4. Code Relationships

.4.1 1911, 1912/1, 1912/9, 2101 Readers

When using the following tables, note:-

a) Internal codes are expressed in "Octal".

b) Where no character is shown, that punching combination is not used.

c) See Part 2, Section 4.4 p.2, regarding punching "10-0".

d) 2101 will decode 8-2 to *22 (same as "11-0"), other readers will produce indeterminate internal code.

e) See Part 2, Section 4.4, p.3, regarding I.S.O. code.

Holes Punched	Character 1900	1500	ISO	Internal Code
None	Sp	Sp	Sp	20
0	0	0	0	00
1	1	1	1	01
2	2	2	2	02
3	3	3	3	03
4	4	4	4	04
5	5	5	5	05
6	6	6	6	06
7	7	7	7	07
8	8	8	8	10
9	9	9	9	11
8-2			:	Note d)
8-3	#	¼	;	23
8-4	@	@	<	40
8-5	(	(	=	30
8-6	)	)	>	31
8-7	]		?	75
10	&	&	\	26 Note c)
10-0		+0		
10-1	A	A	A	41
10-2	B	B	B	42
10-3	C	C	C	43
10-4	D	D	D	44
10-5	E	E	E	45
10-6	F	F	F	46
10-7	G	G	G	47
10-8	H	H	H	50
10-9	I	I	I	51
10-8-2	+	+	J	33
10-8-3	.	.	K	36
10-8-4	;	;	L	13
10-8-5	:	:	M	12
10-8-6	,	,	N	27

Holes Punched	Character 1900	1500	ISO	Internal Code
10-8-7	!		O	21
11	-	-	P	35
11-0	"	½		22
11-1	J	J	Q	52
11-2	K	K	R	53
11-3	L	L	S	54
11-4	M	M	T	55
11-5	N	N	U	56
11-6	O	O	V	57
11-7	P	P	W	60
11-8	Q	Q	X	61
11-9	R	R	Y	62
11-8-2	[	EI	Z	73
11-8-3	¢	£	[	74
11-8-4	*	*	~	32
11-8-5	>	ED	]	16
11-8-6	<	EF	^	14
11-8-7	↑		-	76
0-1	/	/	!	37
0-2	S	S	"	63
0-3	T	T	#	64
0-4	U	U	CS	65
0-5	V	V	%	66
0-6	W	W	&	67
0-7	X	X	'	70
0-8	Y	Y	(	71
0-9	Z	Z	)	72
0-8-2	£	EB	*	24
0-8-3	,	,	+	34
0-8-4	%	%	,	25
0-8-5	?	ISS	-	17
0-8-6	=	=	.	15
0-8-7	←		/	77

4.2 2102 and P.F.311 Card Readers

When using the following tables, note:-

a) Internal codes are expressed in "Octal".
b) Where no character is shown, that punching combination is not used.
c) Interface code is that produced by mode 32 (*40).
d) Translated code is that produced by use of mode 0.
e) In the "I.S.O." column, only characters within the "dense subset" are shown.

Punched Cards

Switch to Code 1

Holes Punched	Character 1900	1500	ISO	Internal Code Interface	Translated	Individual Notes
None	Sp	Sp	Sp	20	20	
0	0	0	0	00	00	
1	1	1	1	01	01	
2	2	2	2	02	02	
3	3	3	3	03	03	
4	4	4	4	04	04	
5	5	5	5	05	05	
6	6	6	6	06	06	
7	7	7	7	07	07	
8	8	8	8	10	10	
9	9	9	9	11	11	
8-2			:	12	22	Same int. codes as "11-0"
8-3	#	¼	;	13	23	
8-4	@	@	<	14	40	
8-5	(	(	=	15	30	
8-6	)	)	>	16	31	
8-7	]		?	17	75	
10	&	&	`	40	26	ISO ch. is "grave" accent.
10-0		+0				
10-1	A	A	A	41	41	
10-2	B	B	B	42	42	
10-3	C	C	C	43	43	
10-4	D	D	D	44	44	
10-5	E	E	E	45	45	
10-6	F	F	F	46	46	
10-7	G	G	G	47	47	
10-8	H	H	H	50	50	
10-9	I	I	I	51	51	
10-8-2	+	+	J	52	33	
10-8-3	.	.	K	53	36	
10-8-4	;	;	L	54	13	
10-8-5	:	:	M	55	12	
10-8-6	'	'	N	56	27	
10-8-7	!		O	57	21	

Switch to Code 1 (cont.)

Holes Punched	Character 1900	Character 1500	ISO	Internal Code Interface	Internal Code Translated	Individual Notes
11	-	-	P	60	35	
11-0	"	½		12	22	Same int. code as "8-2"
11-1	J	J	Q	61	52	
11-2	K	K	R	62	53	
11-3	L	L	S	63	54	
11-4	M	M	T	64	55	
11-5	N	N	U	65	56	
11-6	O	O	V	66	57	
11-7	P	P	W	67	60	
11-8	Q	Q	X	70	61	
1-9	R	R	Y	71	62	
11-8-2	[	EI	Z	72	73	
11-8-3	∅	£	[	73	74	
11-8-4	*	*	~	74	32	
11-8-5	>	ED	]	75	16	
11-8-6	<	EF	^	76	14	
11-8-7	↑		-	77	76	ISO ch. is "underscore"
0-1	/	/	!	21	37	
0-2	S	S	"	22	63	
0-3	T	T	#	23	64	
0-4	U	U	CS	24	65	CS = Currency Symbol
0-5	V	V	%	25	66	
0-6	W	W	&	26	67	
0-7	X	X	'	27	70	ISO ch. is "acute" accent
0-8	Y	Y	(	30	71	
0-9	Z	Z	)	31	72	
0-8-2	£	EB	*	32	24	
0-8-3	,	,	+	33	34	
0-8-4	%	%	,	34	25	
0-8-5	?	ISS	-	35	17	
0-8-6	=	=	.	36	15	
0-8-7	←		/	37	77	

Switch to Code 2

Holes Punched	Character 1004	1300	1401	Internal Code Interface	Translated	Individual Notes
None	Sp	Sp	Sp	20	20	
0	0	0	0	00	00	
1	1	1	1	01	01	
2	2	2	2	02	02	
3	3	3	3	03	03	
4	4	4	4	04	04	
5	5	5	5	05	05	
6	6	6	6	06	06	
7	7	7	7	07	07	
8	8	8	8	10	10	
9	9	9	9	11	11	
8-2	+	10		12	22	
8-3	¼	11	#	13	23	
8-4	@	12	@	14	40	
8-5	½	13	:	15	30	
8-6	>	14	>	16	31	
8-7	¾	15	√	17	75	
10	10	10	&	40	26	
10-0	?		?	52	33	
10-1	A	A	A	41	41	
10-2	B	B	B	42	42	
10-3	C	C	C	43	43	
10-4	D	D	D	44	44	
10-5	E	E	E	45	45	
10-6	F	F	F	46	46	
10-7	G	G	G	47	47	
10-8	H	H	H	50	50	
10-9	I	I	I	51	51	
10-8-3	.		.	53	36	
10-8-4	£		◇	54	13	
10-8-5	[		(	55	12	
10-8-6	<		<	56	27	
10-8-7	=		≠	57	21	
11	11	11	-	60	35	
11-0	/		!	72	73	
11-1	J	J	J	61	52	

Switch to Code 2 (cont.)

Holes Punched	Character 1004	1300	1401	Internal Code Interface	Translated	Individual Notes
11-2	K	K	K	62	53	
11-3	L	L	L	63	54	
11-4	M	M	M	64	55	
11-5	N	N	N	65	56	
11-6	O	O	O	66	57	
11-7	P	P	P	67	60	
11-8	Q	Q	Q	70	61	
11-9	R	R	R	71	62	
11-8-3	-		¢	73	74	
11-8-4	*		*	74	32	Same int. codes as "1-8"
11-8-5	]		)	75	16	" " " " "1-9"
11-8-6	;		;	76	14	
11-8-7	¢		△	77	76	
0-1	&	&	/	21	37	
0-2	S	S	S	22	63	
0-3	T	T	T	23	64	
0-4	U	U	U	24	65	
0-5	V	V	V	25	66	
0-6	W	W	W	26	67	
0-7	X	X	X	27	70	
0-8	Y	Y	Y	30	71	
0-9	Z	Z	Z	31	72	
0-8-2	≠		≠	32	24	Same int. codes as "1-2"
0-8-3	,		,	33	34	" " " " "1-3"
0-8-4	%		%	34	25	" " " " "1-4"
0-8-5	(		=	35	17	" " " " "1-5"
0-8-6	\		'	36	15	" " " " "1-6"
0-8-7	)		"	37	77	" " " " "1-7"
1-2		%		32	24	" " " " "0-8-2"
1-3		¼		33	34	" " " " "0-8-3"
1-4		-		34	25	" " " " "0-8-4"
1-5		/		35	17	" " " " "0-8-5"
1-6		½		36	15	" " " " "0-8-6"
1-7		.		37	77	" " " " "0-8-7"
1-8		@		74	32	" " " " "11-8-4"
1-9		¾		75	16	" " " " "11-8-5"

.4.3 2103 Card Reader

When using the following tables, note:-

a) Internal codes are expressed in "Octal".

b) Where no character is shown, that punching combination is not used.

c) "Interface" code is that produced by reading in "mode 32 (*40)".

d) "Translated" code is that produced by reading in "mode 0".

e) In the "I.S.O." column, only characters within the "dense subset" are shown.

f) In codes A1 and C1, "no punching" may represent "zero".

CODE RELATIONSHIPS (Normal Reading)

Holes Punched	Character Represented 1900	A1	C1	ISO	Internal Code Interface	Translated	Individual Notes
None	Sp	Sp	Sp	Sp	20	20	See note f)
0	0	0	0	0	00	00	
1	1	1	1	1	01	01	
2	2	2	2	2	02	02	
3	3	3	3	3	03	03	
4	4	4	4	4	04	04	
5	5	5	5	5	05	05	
6	6	6	6	6	06	06	
7	7	7	7	7	07	07	
8	8	8	8	8	10	10	
9	9	9	9	9	11	11	
8-2				:	12	22	Same int. code as "11-0"
8-3	#			;	13	23	
8-4	@			<	14	40	
8-5	(			=	15	30	
8-6	)			>	16	31	
8-7	]			?	17	75	
10	&	10	10	`	40	26	ISO ch. is "grave" accent.
10-0							
10-1	A	A	A	A	41	41	
10-2	B	B	B	B	42	42	
10-3	C	C	C	C	43	43	
10-4	D	D	D	D	44	44	
10-5	E	E	E	E	45	45	
10-6	F	F	F	F	46	46	
10-7	G	G	G	G	47	47	
10-8	H	H	H	H	50	50	
10-9	I	I	I	I	51	51	
10-9-2	+			J	52	33	
10-8-3	.			K	53	36	
10-8-4	;			L	54	13	
10-8-5	:			M	55	12	
10-8-6	,			N	56	27	
10-8-7	!			O	67	21	

Holes Punched	Character Represented 1900	A1	C1	ISO	Internal Code Interface	Translated	Individual Notes
11	-	11	11	P	60	35	
11-0	"				12	22	Same int codes as "8-2"
11-1	J	J	J	Q	61	52	
11-2	K	K	K	R	62	53	
11-3	L	L	L	S	63	54	
11-4	M	M	M	T	64	55	
11-5	N	N	N	U	65	56	
11-6	O	O	O	V	66	57	
11-7	P	P	P	W	67	60	
11-8	Q	Q	Q	X	70	61	
11-9	R	R	R	Y	71	62	
11-8-2	[			Z	72	73	
11-8-3	≠			[	73	74	
11-8-4	*			~	74	32	
11-8-5	>			]	75	16	
11-8-6	<			^	76	14	
11-8-7	↑			--	77	76	ISO ch. is "underscore"
0-1	/	J	S	!	21	37	
0-2	S	S	T	"	22	63	
0-3	T	T	U	#/=	23	64	
0-4	U	U	V	CS	24	65	CS= Currency Symbol
0-5	V	V	W	%	25	66	
0-6	W	W	X	&	26	67	
0-7	X	X	Y	´	27	70	ISO ch. is "acute" accent
0-8	Y	Y	Z	(	30	71	
0-9	Z	Z	&	)	31	72	
0-8-2	£			*	32	24	
0-8-3	,			+	33	34	
0-8-4	%			,	34	25	
0-8-5	?			-	35	17	
0-8-6	=			.	36	15	
0-8-7	<--			/	37	77	

.4.4 1004 Card Reader

When using the following tables, please note that 1900 internal codes are expressed in "Octal".

Notes:

a) 1900 code and character when using conversion table 1

b) 1900 code and character when using conversion table 2

c) When using conversion table 2, this punching is not distinguishable from "11".

d) When using conversion table 2, this punching is not distinguishable from "10".

e) Alternative for "10-8-2" (provided for cards punched via 1004, see Section 4.5, p.5).

f) Alternative for "11-8-2" (provided for cards punched via 1004, see Section 4.5, p.5).

Punched Cards

CODE RELATIONSHIPS, conversion tables 1 and 2 (1004 card code)

Holes Punched	1004 ch.	1900 Int. code	1900 ch.	Notes
None	Sp	20	Sp	
0	0	00	0	
1	1	01	1	
2	2	02	2	
3	3	03	3	
4	4	04	4	
5	5	05	5	
6	6	06	6	
7	7	07	7	
8	8	10	8	
9	9	11	9	
8-2	+	33	+	
8-3	¼	27	'	
8-4	@	40	@	
8-5	½	76	↑	
8-6	>	16	>	
8-7	¾	77	<—	
10	10	12	:	a)
		26	&	b)
10-0	?	17	?	
10-1	A	41	A	
10-2	B	42	B	
10-3	C	43	C	
10-4	D	44	D	
10-5	E	45	E	
10-6	F	46	F	
10-7	G	47	G	
10-8	H	50	H	
10-9	I	51	I	
10-8-3	.	36	.	
10-8-4	£	24	£	
10-8-5	[	73	[	
10-8-6	<	14	<	
10-8-7	=	15	=	
11	11	22	"	a)
		35	-	b)
11-0	/	37	/	
11-1	J	52	J	
11-2	K	53	K	
11-3	L	54	L	
11-4	M	55	M	
11-5	N	56	N	
11-6	O	57	O	
11-7	P	60	P	
11-8	Q	61	Q	
11-9	R	62	R	
11-8-3	-	35	-	c)
11-8-4	*	32	*	
11-8-5	]	75	]	
11-8-6	;	13	;	
11-8-7	∅	74	∅	
0-1	&	26	&	d)
0-2	S	63	S	
0-3	T	64	T	
0-4	U	65	U	
0-5	V	66	V	
0-6	W	67	W	
0-7	X	70	X	
0-8	Y	71	Y	
0-9	Z	72	Z	
0-8-2	≠	23	#	
0-8-3	,	34	,	
0-8-4	%	25	%	
0-8-5	(	30	(	
0-8-6	\	21	!	
0-8-7	)	31	)	

CODE RELATIONSHIPS, conversion table 3
(1900 card code)

Holes Punched	1900 ch.	1900 Int. code	Note
NONE	Sp	20	
0	0	00	
1	1	01	
2	2	02	
3	3	03	
4	4	04	
5	5	05	
6	6	06	
7	7	07	
8	8	10	
9	9	11	
8-2		73	e)
8-3	#	23	
8-4	@	40	
8-5	(	30	
8-6	)	31	
8-7	]	75	
10	&	26	
10-0		33	f)
10-1	A	41	
10-2	B	42	
10-3	C	43	
10-4	D	44	
10-5	E	45	
10-6	F	46	
10-7	G	47	
10-8	H	50	
10-9	I	51	
10-8-2	+	33	
10-8-3	.	36	
10-8-4	;	13	
10-8-5	:	12	
10-8-6	'	27	

Holes Punched	1900 ch.	1900 Int. code	Note
10-8-7	!	21	
11	-	35	
11-0	"	22	
11-1	J	52	
11-2	K	53	
11-3	L	54	
11-4	M	55	
11-5	N	56	
11-6	O	57	
11-7	P	60	
11-8	Q	61	
11-9	R	62	
11-8-2	[	73	
11-8-3	$	74	
11-8-4	"	32	
11-8-5	>	16	
11-8-6	<	14	
11-8-7	↑	76	
0-1	/	37	
0-2	S	63	
0-3	T	64	
0-4	U	65	
0-5	V	66	
0-6	W	67	
0-7	X	70	
0-8	Y	71	
0-9	Z	72	
0-8-2	£	24	
0-8-3	,	34	
0-8-4	%	25	
0-8-5	?	17	
0-8-6	=	15	
0-8-7	←	77	

4.5 90-Column Card Punches

When using the following tables, please note:-

a) Internal Codes are expressed in "Octal".

b) Where no character is shown, that punching combination is not used in the card code that it is listed under.

c) It is not possible to punch the 1500 character "+0" (10-0).

.5 90-Column Punch Code Relationships

Internal Code	Holes Punched	Character 1900	Character 1500
00	0	0	0
01	1	1	1
02	2	2	2
03	3	3	3
04	4	4	4
05	5	5	5
06	6	6	6
07	7	7	7
10	8	8	8
11	9	9	9
12	10-8-5	:	:
13	10-8-4	;	;
14	11-8-6	<	EF
15	0-8-6	=	=
16	11-8-5	>	ED
17	0-8-5	?	ISS
20	NONE	Sp	Sp
21	10-8-7	!	
22	11-0	"	½
23	8-3	#	¼
24	0-8-2	£	EB
25	0-8-4	%	%
26	10	&	&
27	10-8-6	'	'
30	8-5	(	(
31	8-6	)	)
32	11-8-4	*	*
33	10-8-2	+	+
34	0-8-3	,	,
35	11	-	-
36	10-8-3	.	.
37	0-1	/	/
40	8-4	@	@
41	10-1	A	A
42	10-2	B	B
43	10-3	C	C
44	10-4	D	D
45	10-5	E	E
46	10-6	F	F
47	10-7	G	G
50	10-8	H	H
51	10-9	I	I
52	11-1	J	J
53	11-2	K	K
54	11-3	L	L
55	11-4	M	M
56	11-5	N	N
57	11-6	O	O
60	11-7	P	P
61	11-8	Q	Q
62	11-9	R	R
63	0-2	S	S
64	0-3	T	T
65	0-4	U	U
66	0-5	V	V
67	0-6	W	W
70	0-7	X	X
71	0-8	Y	Y
72	0-9	Z	Z
73	11-8-2	[	EI
74	11-8-3	$	£
75	8-7	]	
76	11-8-7	↑	
77	0-8-7	←	

CONTENTS

.1 General

.2 Types and Facilities

.3 Instruction Codes and Other Major Facilities

.4 Error Conditions

.5 Operational Aspects

.1 General

A standard 80-column card is used for card input operations on the 1900 series. Rectangular punched holes are most commonly used. However, one type of reader, the 2103, is available for reading 40-column cards with the 1900 processor (see .2, p 5).

The feeding mechanism of the card readers is either "cyclic", in which case a card can only be fed from hopper to reading station at certain points in the basic feed cycle of the mechanism; or is an "on demand" type which can feed a card into a reading station more directly after the issue of a card read instruction.

Each card is read twice during the reading cycle, and the two readings are compared. If a reading error is discovered, the card is automatically diverted to a special reject stacker (this does not apply on the 2101, 2102 or 2103 readers) and Executive is informed.

A number of photo-electric devices are fitted to the card track of a reader to check the arrival of a card at certain points. Misfeeds and jams are detected in this way.

Cards are translated from the various codes accepted into 1900 card code before being feed to the Central Processor. For code relationships, see Section 4.7.

.1.1 Reading 90-Column Cards

1912/9 card reader is a special version of 1912/1 which can read (in addition to normal 80-col. cards) 90-column cards. Switching between 80- and 90-column working is by means of a manual switch, there being no indication to the processor of the current state of the switch.

90-column cards are basically 45-column cards, as used by some early Powers-Samas equipment and by Remington Rand, but punching is in the form of a six-bit code, allowing two characters to be punched in each column, rather than the usual 1, 2, or 3-hole punchings.

On 1900 series such cards will be read as if they were 45-column cards by use of the 'Card Image' mode of reading (the card image feature will always be present on 1912/9), giving 90 (maximum) internal characters from a card. Programs which read 90-column cards must use mode *02, with a count of not more than 90 in the third word (word 2) of the control area.

The object program has a choice of reading a complete card, or just the first "N" internal characters which appear (in normal reading, this means the "first N columns"). It is not possible to read a card in several sections, as once a card starts to move past the read head of a reader, it cannot be stopped, i.e. when a program calls for only part of a card to be read, the remaining data on that card is inaccessible in the reading pass concerned.

N.B. In cases where the object program specifies a number of internal characters which is not a multiple of four (i.e. the data area does not consist of an integral number of 1900 words), the "spare" characters in the last word of the data area will, on completion of the reading of the card, contain indeterminate data.

.2 Types and Facilities

	CARD READER - TYPE		
Catalogue No.	1911/1	1912/1	2101
Project File No.	P.F. 131	P.F. 238	P.F. 207
No. of columns	80	80	80
Type of Hole	Rectangular	Rectangular	Rectangular
Max. Speed	900 c.p.m.	300 c.p.m.	1600 c.p.m.
Type of Feed	On-demand	½ cyclic	cyclic
Non-1900 Codes	Restricted 1500	Restricted 1500	Restricted 1500, 1004, 1300, IBM 1401, I.S.O.
Card Image Read	Yes	Yes	Yes
Reject Stacker	Yes	Yes	No
Miscellaneous	Decoder B Full Facility	Decoder B Full Facility	Decoder B Full Facility

	CARD READER - TYPE	
Catalogue No.	2102	2104/1
Project File No.	P.F. 219	P.F. 311
No. of columns	80	80
Type of Hole	Rectangular	Rectangular
Max. Speed	300 c.p.m.	600 c.p.m.
Type of Feed	cyclic	cyclic
Non-1900 Codes	1500 1004 1300 I.B.M. 1401 I.S.O.	1500 1004 1300 I.B.M. 1401 I.S.O.
Card Image Read	No	No
Reject Stacker	No	No
Miscellaneous	Decoder E Simplified Facility	Decoder G Simplified Facility

.2 Cont'd.

	CARD READER - TYPE			
Catalogue No.	1912/9	2103	1004	(a) see below
Project File No.	P.R.238/2	P.F. 222	P.F. 246/1	--
No. of Columns	80/90	40	80	80/51
Type of Hole	Rectangular	Round	Rectangular	Rectangular
Max. Speed	900 c.p.m.	600 c.p.m.	Environment Dependent	1200 c.p.m. (b)
Type of feed	cyclic	cyclic	cyclic	cyclic
Non-1900 Codes	Restricted 1500	A1 C1 I.S.O.	1004	1500
Card Image Read	Always Present	No	Yes, when not on-line to 1900	Yes
Reject Stacker	Yes	No	Yes	Yes
Miscellaneous	Decoder B Full Facility	Simplified Facility	Restricted	Decoder B. Full Facility

(a) This reader is a "special" based on the CDC 405 mechanism, to satisfy a customer requiring to read 51 column cards.

(b) This speed increases to 1600 c.p.m. when reading 51-column cards.

.2 cont'd

.2.1 2103 Reader

The 2103 card reader is a simplified facility device, which will read 40-column round hole punched cards at a rate of 500 c.p.m. These cards present certain restrictions which are:-

i) They cannot be used for loading Executive.

ii) They cannot be used for loading "Binary" programs.

iii) They cannot be used for loading operational directives.

Due to these restrictions, the 2103 card reader must be accompanied by an 80-column card reader or a paper tape reader on any one processor.

This reader is fitted with a special decoder, capable of uniquely decoding the following codes:-

I.C.T. 1900 card code

I.C.T. A1 card code

I.C.T. C1 card code (Powers-Samas card code)

I.S.O. "dense subset" card code

In some 40-column conventional equipment installations, the "upper curtate" (rows 10, 11 and 0) of certain columns of numeric fields is used for designatory, or tabular control, purposes, under which circumstances any combination of holes may appear in the upper curtate. (N.B. in many cases "zero" is represented by a blank "lower curtate", rather than by a punching in the "0" row.) The result of this is that punching combinations outside the defined card code ranges may appear. In order that such cards can be read into a processor, a special mode of reading, known as "modified read mode", which functions as follows, is provided:-

Each column of the card will be represented within the object program data area by two consecutive 1900 characters. The first (more significant) character of each pair will contain either:

(a) if there was no hole, or only one hole, punched in the upper curtate of the column: the same character as would appear in respect of that column in a "normal mode read" (see Section 4.4, page 12.)

or (b) If there was more than one hole in the upper curtate of the column : the binary value of the punching in the lower curtate (rows 1-9

.2.1 cont'd

inclusive); the punching in the lower curtate must be confined to a single hole, be one of the combinations 8-2 to 8-7 inclusive, or be "blank"; any other punching combination produces an indeterminate code.

The second character of each pair will always contain an "image" of the punching in the upper curtate, which will occupy the three least significant bits of the character (the other three bits will always contain "zero"), with the 10 row appearing in the least significant bit, the 11 row in the middle bit, and the 0 row in the most significant bit of the three.

Whilst the translation facility is designed for the 1900 card code, it may prove useful when reading other card codes, (particularly A1, which only differs from a subset of the 1900 in respect of the significance of three punching combinations), but it must be noted that, if the object program performs any further translations, overall efficiency will be inferior to that which would be obtained if the object program undertook the complete translation process.

.2.2 Reading via Link to 1004 Data Processor

The 1004 card code differs from 1900 card code only in the case of the punchings "8-2" and "10-0", which are used in the 1004 code but not in 1900 code. Due to the fact that it may be required to read 1900 code as well as 1004 code on a 1004 Reader, (i.e. I.C.T. library routines and utility programs), a modification is present in which the punching "10-8-2" is read as "10-0", and "11-8-2", the same as "8-2". The result is that packs punched in either code can be read.

Since 1004 code and 1900 code are not identical, a special 1004 program performs the appropriate code translations. This translator program is loaded by cards and can be changed by reading in a new pack. Three versions are available:-

Conversion Table 1 : for 1004 standard card code

Conversion Table 2 : for 1004 card code with "single column pence"

Conversion Table 3 : for 1900 card code

.2.2 cont'd

It must also be noted that some of the "miscellaneous" characters have different interpretations between 1900 code and 1004 code. The tables on Pages 16, 17 will show this to be apparent.

Loading of Executive, Binary and reading of directives can be effected on the 1004, subject to punching changes noted above, which will be taken care of if the cards are generated via a 1004 punch. Alternatively, Table 1 can be used for loading binary programs, provided the whole pack to be read was punched out via 1004 and that Table 1 was used for punching.

.3 Instruction Codes and Other Major Facilities

.3.1 Internal Type No.: 3

.3.2 Method of Programming: Suspension, with Direct-Response Mode Facilities

.3.3 Summary of Applicable Peripheral Control Instructions

150 (SUSBY) : Suspend program member if reader is busy
151 (REL) : Release reader from program
152 (DIS) : Disengage reader
153 (-) : Illegal
154 (CONT) : Load more "Binary" program from reader (2103 - undefined)
155 (SUSDP) : Illegal
156 (ALLOT) : Assign and/or identify reader
157 (PERI) : Activate reader

The general definitions of these instructions are to be found in Part 1, Section 15. of the manual.

.3.4 Other Major Facilities

All readers, apart from the 2103, can be used for the following purposes (switched to Code 1 in the case of Decoder E or G):-

(a) Loading of binary program (Directive: LOad)
(b) Reading of Operational Directives
(c) Loading of Executive

.3.5 Property Codes

.3.5.1 Structure (applicable to all devices of internal type 3)

Bit(s) 10 : undefined
11-12 : speed
13 : availability of additive mode 32 (*40)
14 : availability of basic mode 2
15-16 : undefined
17 : cannot read 80-column cards

Note: Additive mode 32 is available on readers with Decoder E and means "inhibit translation into 1900 code". (On these readers Mode 0 includes translation from hardware code into correct 1900 code.) Basic mode 2 is "card image" on readers with Decoder B (provided F911 or F912 is fitted to reader), but is "modified mode" in 40-column (a partial card-image system).

.3.5.2 Specific

The following are the codes which will appear in response to a 156 instruction on those Executives which provide "property" facilities (x = indeterminate):-

	1911/1	1912/1	1912/9	2101	2102	~~P.P. 2404 341~~
With "Card Image"	X1001XX0	X0001XX0	X1001XX0	X1101XX0	--	X0101XX0
Without "Card Image"	X1000XX0	X0000XX0	(cannot occur)	X1100XX0	X0010XX0	X0100XX0

AL/16

	1004	2103
With "Card Image"	--	--
Without "Card Image"	X0000XX0	X0011XX1

.3.6 150 Instruction

Suspension due to this instruction, when Executive's code translation facility is in use, is not removed until the translation process is completed, which is some time after the actual reading is completed.

.3.7 152 Instruction

This instruction will turn a "full facility" device offline. In the case of

the 2101, 2102, 2103 and 1004 readers, being simplified facility devices, this instruction will not cause them to be ~~turned offline~~ disengaged. However, after this instruction, Executive will not allow the object program to activate them until the operator has intervened. AL15

.3.8 154 Instruction

On the 2101, code switch must be set to Code 1. This instruction may only be used when the cards being read are standard 80-column cards punched in 1900 code, and any attempt to use this instruction when these conditions are not satisfied has an indeterminate effect.

.3.9 156 Instruction

The full definition is applicable to all these readers.

.3.10 157 Instruction

.3.10.1 Suspension

In the case of card readers fitted with Decoder E, suspension due to any 157 instruction issued before the action initiated by the previous 157 to the reader will, if that previous 157 was "mode 0", not be released until the code translation performed by Executive has been completed.

.3.10.2 Form of Control Areas

Word 0 : Type 3/Mode (see .3.10.3)

1 : Reply Word

Bit 0 : Transfer in progress (when set, bit 3 is indeterminate).

Bit 3 : Error in transfer

All other bits : Undefined; except as provided for when working in Direct-Response mode.

Word 2 : Count in characters

Word 3 : Start address, must point to character 0 of a word.

Note: When card image mode (basic mode 2) is used, it is possible to specify an odd number of characters, therefore transferring only the top half (rows 10-3) of the last column read.

.3.10.3 Modes.

(a) Basic Mode 0

1911/1, 1912/1, 2101 - Read one card, in "normal" fashion (one internal character per card column); permitted count values 1-80 inclusive.

2102 - Read one card, translating the data read into 1900 internal code.

2103 - Read one card, in "normal" fashion (one internal character per card column), translating data read into 1900 internal code; permitted count values 1-40 inclusive.

1912/9 - This mode has an indeterminate effect on this reader.

1004 - See Section 4.5 page 7.

(b) Basic Mode 2

1911/1, 1912/1, 1912/9, 2101 - Read one card, in "card image" fashion, (two internal characters per card column); permitted count values 1-160 inclusive where reading 80-column cards (1-90 where 1912/9 reader is reading 90-column cards).

N.B. When a card reader is not fitted with "card image" facilities, this mode will have an indeterminate effect.

2103 - Read one card in "modified" manner, (two internal characters per card column), translating data read into 1900 internal code; permitted count values 1-80 inclusive.

(c) Additive Mode 32 (#40)

Inhibit code translation. With 80-column card readers, this mode is only available on readers fitted with Decoder E, namely the 2102.

2102 - Inhibit code translation, applicable to basic mode 0.

2103 - Inhibit code translation, applicable to either of the basic modes.

.3.10.4 Own Monitoring.

Mode 12288 (# 30000) is permitted with any valid 157 instruction, and will have effect as defined in Part 2, Section 6.

.3.10.5 The Count.

The following are the permitted ranges of values for the count (all being inclusive), any violation having undefined effect:-

.3.10.5 Cont'd

Card Size	80-column	51-column	"90"-column	40-column
Mode 0	1-80	1-51	x	1-40
Mode 2	1-160	1-102	1-90	1-80
Mode *40	1-80	x	x	x

Notes: (1) "x" indicates conditions which are invalid uses of the mode concerned, hence the effect will be indeterminate, regardless of the count value.

(2) When reading in mode 2 (card image), odd values of count are permitted and the transfer will be performed correctly.

(3) In cases where the count is not a multiple of 4, i.e. where the end of the data area does not coincide with a word boundary, the "spare" character positions in the last word of the data area will contain indeterminate data.

.4 Error Conditions

.4.1 Executive's Recovery Procedure

These readers are "assisted repeat" devices, i.e. when an error occurs, Executive calls for the operator to take the last card read and to place it at the front of the remaining unread cards in the hopper and then re-engage the reader.

.4.2 Miscellaneous

Should operator re-engage the reader without first re-positioning the offending card, neither Executive nor the object program will be aware of this.

N.B. In the case of the 1912/9 reader, should the card type selector switch be wrongly set, an error will be signalled as a result of the mis-registration of the punching in the card.

.4.3 Peripheral Incidents

(a) Machine Fault

The action in the event of a machine fault is described in Section 3, p.1.

.4.3 cont'd

Possible causes

i. hopper empty
ii. low level of cards in hopper with no card weight
iii. reject stacker full
iv. main stacker needs emptying

(b) Error

In the case of an error occurring during the transfer, the Card Reader is treated as for "assisted repeat" devices, Section 3, p.4.

Possible Causes

i. misfeed
ii. misreading (read check fail)
iii. error in transfer of data
iv. card jam (card wreck)

.5 Operational Aspects (Manual Operating Environments)

.5.1 Operator Directives

The following 1904/1905 Executive Operator Directives are applicable:-

GIve
RIght (not applicable to 1004)
TAke (" " " ")
WRong (" " " ")
LOad (" " " 2103)

On other Executive/Processor combinations, the equivalent directives are applicable.

In the case of readers having a code selection switch, i.e. those with either Decoder E or G, the switch must be in the "Code 1" state when the LOad directive is issued, and remain so throughout the loading process.

.5.2 Messages from Executive

Messages will appear in respect of the following incidents:-

UNIT q USED Assignment of a reader by 156 instructions.

UNIT q FREE Release of a reader by 151 instruction.

UNIT q ERR, or UNIT q R & FX Read Error.

UNIT q FIX 157 instruction issued to reader whilst disengaged,

.5.2 cont'd

or 157 instruction issued to reader after 152 instruction unless the operator has since attended to the reader (only applicable to 2102, 2103),

or 157 instruction issued when the 1004/7201 link is inoperable (applicable to 1004 only).

.5.3 Action Required from Operator on Occurrence of Read Error

1911/1 Reader - Press "run out" button, wait for reader motors to stop, remove the three cards which will be found in reject stacker, place these three cards at front of unread cards, replace same in hopper, press "allocate" button.

1912/1, 1912/9 Readers - Remove card which will be found in reject stacker, place it at front of unread cards, replace same in hopper, check setting of code selection switch, press "allocate" button.

2101, 2102, P.F. 311 Readers - Remove last card from the stacker (the card at the end nearest the card track), place it at front of unread cards, replace pack in hopper, press "start", then "allocate" buttons.

Miscellaneous - On the 1912/9 reader, there is a card type selection switch which must be set to the appropriate state when reading cards, according to whether 80 or "90" column cards are to be read.

N.B. This switch is actually labelled 45/80 - this is because "90" column cards are, physically, the same as 45 column cards.

.5.4 Action required from Operator on occurrence of "FIX" (or equivalent) Message

1911/1, 1912/1, 1912/9, 2101 Readers - After performing any special action called for in individual operating instructions, press "allocate" button.

2102, 2103 Readers

Causes: (a) If reader is found "disengaged": operator had punched "Hold".
(b) If reader is found "engaged": 152 instruction.

Remedies: After performing any special action called for in individual operating instructions, press "allocate" button. (If message was due to 152 instruction, "Hold" button must be pressed before pressing "Allocate".)

5.4 continued

1004 Readers

When reading via 1004 processor, the operator types messages to Executive instead of operating control buttons.

CONTENTS

.1 General

.2 Punch Types and Characteristics

.3 Instruction Codes

.4 Other Major Facilities

.5 Property Codes

.6 Instructions

.7 Miscellaneous

.8 Error Conditions

.9 Operational Aspects

.1 General

Normally card punching is in accordance with the I.C.T. 64-character card code, and 80-column cards are used. Conversion is from 1900 internal machine code and is effected automatically.

Card punches have either "cyclic" feed where a card can only be fed at certain points in the basic feed cycle of the mechanism, or an "on demand" feed where a card can be fed into the punching station mode directly after the issue of a punching instruction.

The punching of each card is checked, either by a brush-sensory technique where, after punching, a card is read back and the data compared with the punching data, or by monitoring the displacement of punch knives and comparing this with the punching data.

.2 Punch Types and Characteristics

FEATURE	CARD PUNCH TYPE			
Catalogue No.	1920	1922/1	2151	1004
Project File No.	241	237	302/1	246/1
Maximum Speed (Cards/Minute)	100	33	300	Environment Dependent
Action on Errors	Cards "offset"	--	Cards to "reject stacker"	Cards to "reject stacker"
Feed	1/14th cyclic	on-demand	1/5th cyclic	Type dependent
Checking	Delayed	Immediate	Delayed	Delayed
Punching	By rows	By columns	By rows	By columns
Miscellaneous	Unbuffered	Serial Punch	Buffered	Buffered

.3 Instruction Codes and Other Major Facilities

.3.1 Internal Type No. : 4

.3.2 Method of Programming : Suspension

.3.3 Summary of Applicable Peripheral Control Instructions

150 (SUSBY)	:	Suspend program member if punch busy
151 (REL)	:	Release punch from program
152 (DIS)	:	Disengage punch
153 (-)	:	Illegal
154 (CONT)	:	Illegal
155 (SUSDP)	:	Dump program in "binary" form on punch
156 (ALLOT)	:	Assign and/or identify punch
157 (PERI)	:	Activate Punch

The general definitions of these instructions are to be found in Part 1, Section 15 of the manual.

.4 Other Major Facilities

The operator can cause a program to be dumped onto cards, by use of the directive "DUMP", on any of these punches.

When dumping via a 1004 processor, code conversion table 3 should be used. Alternatively, code table 1 may be used, but the resulting pack cannot be mixed with cards punched in 1900 code, nor with any punched using table 3 and, furthermore, table 1 will have to be used when the pack is read back. Binary programs punched out via a 1004 cannot be loaded into a 1900, except via a 1004.

.5 Property Codes

.5.1. Structure (applicable to all devices type 4)

Bit(s)	10	: undefined
	11 - 12	: speed
	13 - 17	: undefined

.5.2 Specific

The following are the codes which will appear in response to a 156 instruction, on those Executives which provide "property" facilities (x = indeterminate):-

	1920	: X01XXXXX
	1922/1	: X00XXXXX
	2151	: X11XXXXX
via	1004	: X10XXXXX

.6 Instructions

.6.1 152 Instruction

Because all these punches are "Full Facility" devices, this instruction will cause the punch to be turned "offline".

On 1004, the punch cannot be disengaged individually, so Executive maintains a memory state of 152 instructions which is unset by an operator directive which prevents the object program from operating the reader, after a 152 order, until the operator has intervened.

.6.2 156 Instruction

The full definition is applicable to these punches.

.6.3 157 Instruction

(a) Form of Control Area

Word 0 : Type (4)/Mode (0)

1 : Reply word

Bit 0 : Transfer in progress (bit 3 indeterminate).

Bit 3 : Error (see d, below)

Other Bits : Indeterminate (except for successful transfers on 1920 and 2151, the least significant 7 bits will contain 20 and 0 respectively).

2 : Count (characters), must be 80.

3 : Start Address, must point to character 0 of a word.

.6.3 cont'd

(b) Modes

Basic Mode 0 : Punch one card.

(c) Own Monitoring

Mode 12288 (*30000) is permitted with any valid 157 instruction, and will have effect as defined in Part 2, Section 6.

(d) Errors

The exact significance of bit 3 of the reply word depends on the model of punch in use, viz.:-

1920 and 2151 : Error in previous card

1922 : Error in current card

Object programs which include special recovery routines which operate on the detection of this bit of the reply must make allowance for this fact. (This comment applies both where Own Monitoring is in use and in normal programming, as the operator may have typed "GO" rather than have just re-engaged the punch, which has a similar effect to Own Monitoring). It must also be noted that, on 1920 and 1951, the signalling of an error causes the offsetting/rejection of both the offending card and the one being currently punched, which may be mispunched, without further indication. A study of the error conditions (see .8 below) should be made before attempting to write any error recovery routines.

.7 Miscellaneous

To ensure that all cards punched are fully checked, the object program must, before releasing a punch (or deleting itself), punch two dummy (blank) cards. (This is applicable even when cards have been punched as a result of a 155 instruction).

.8 Error Conditions

.8.1 1920 and 2151 Card Punches

These punches fall into the category of "Automatic Repeat, with Operator

.8.1 cont'd

Notification" devices, (i.e. offending cards repunched without operator intervention, but a message is output to the operator on each repunch cycle).

Because checking takes place one card cycle after punching, not only the offending card has to be repunched, but also the card which follows it. Otherwise, the card sequence would be broken. In order to do this, Executive keeps a copy of data punched into a card until the following card has been correctly punched, so that both can be repunched if an error occurs.

In all cases, both cards are "offset" (1920 Punch) or diverted into the reject stacker (2151 Punch).

.8.2 1922 Card Punch

This punch is an "assisted repeat" device, i.e. after operator intervention, Executive will repunch the offending card. (Note that only the current card has to be repunched, because checking takes place within the punching cycle.)

.8.3 1901 without Console Typewriter

On such configurations (EX1H Executive), 1920 and 2151 punches become "assisted repeat" devices, in that the operator must acknowledge the display for each repunch cycle.

.9 Operational Aspects (Manual Operating Environments)

.9.1 Operator Directives

The following 1904/1905 Executive Operator Directives are applicable:-

GIve
RIght (not applicable to 1004)
TAke (" " " ")
WRong (" " " ")
DUmp
OUtput (not applicable to 1004)

On other Executive/Processor combinations, the equivalent directives are applicable.

.9.2 Messages from Executive

Messages will appear in respect of the following incidents:-

Assignment of a punch due to 156 instruction
Release of a punch due to 151 instruction
Repunch cycle taking place (1920 & 2151 only)
Operator assistance needed for a repunch cycle (1922 only)
Persistent Error
157 instruction issued whilst punch disengaged (or, on 1004, before operator has intervened)

.9.3. Operator Action Required for Repunch Cycle

.9.3.1 1922 Punch - Run-out punch, remove offending card (last one punched), re-prime punch, press "allocate" button.

.9.3.2 1920 & 2151 Punches -

1901 without console typewriter - Acknowledge the display
Processors with console typewriters - none

General - The purpose of these displays is to warn the operator that cards have been offset/rejected, and that such cards should be destroyed.

.9.4 Operator Action Required on Persistent Errors

See individual operating instructions.

Contents

.1 INTRODUCTION
.2 DEFINITIONS
- .2.1 File
- .2.2 Block
- .2.3 Record
- .2.4 Tape mark, Qualifier block, Sentinel
- .2.5 Header label
- .2.6 Trailer label

.3 PRINCIPAL CATEGORIES OF FILES
- .3.1 Simple file
- .3.2 Composite file
- .3.3 Subfile
- .3.4 Executive dump file

.4 OTHER FORMATS
- .4.1 Source program
- .4.2 Unconsolidated semi-compiled program
- .4.3 Consolidated " " "
- .4.4 Binary program
- .4.5 " " with overlay
- .4.6 Uncategorized data in standard records
- .4.7 " " " non-standard records
- .4.8 External image data
- .4.9 Subfile directory
- .4.10 Subroutine directory
- .4.11 Composite Subfile. AL/15

Note: In this section certain terms carry particular meanings, as follows:

Reserved	This location should be written as zero; its contents may become anything.
*	This field is optional; if not used it should be set to zero.

.1 INTRODUCTION

This section defines the standard formats for 1900 series industry-compatible (also referred to as Type 5) and cassette magnetic tapes. Formats not defined here are to be regarded as non-standard. Only limited Executive facilities will be available to tapes with non-standard formats and they will not be acceptable input to standard software packages.

.2 DEFINITIONS

.2.1 File

This form is used for an assembly of information recorded on one or more magnetic tapes that includes such items of information as are required to allow it to be used in reading and writing operations. Thus, a file must include, besides its main information (program or data), other items of information such as the identity of the tape itself, the points where the program or data begins and finishes, the type of file, and so forth.

A multi-reel file carries additional items for identifying its constituent reels.

.2.2 Block

A block is the unit of transfer of information between magnetic tape and core store as the result of a single 157 (PERI) instruction and is thus the unit of recording on the tape. Successive blocks are separated by inter-block gaps. On industry-compatible tape a block comprises (except for a tape mark) 5 to 32,767 words according to limits imposed by the hardware, but in practice a much lower maximum is imposed by certain software. Throughout a file, blocks may be fixed or variable in length.

.2.2. Cont'd.

On cassette tape an additional word, the block sequence word, is recorded before the first word of data proper in a block; this will be considered as word -1. If, in the block sequence word, Bit 0 = 1, the block is equivalent to a tape mark followed by a qualifier block; if Bit 0 = 0, the block is a normal data block. Excluding the block sequence word, the minimum standard block is 5 words long, but Executive will handle shorter blocks correctly, including blocks of length 0 (i.e. block sequence word only).

.2.3. Record

A record is the unit of data, contained within a block. There are various types of record (e.g. source language, semi-compiled). Unless otherwise stated, a standard record comprises an integral number of words. The first word contains (in the modifier part) a count of the number of words in the record, including the count word. The minimum record size is two words, i.e. the count word plus one data word. The maximum is 512 words, 511 data words plus the count word.

As with blocks, records may be of fixed or variable length throughout a file. A block may contain only one record, in which case if the number of words is five or more, the record and the block are identical. If the number of words in the record is less than five, filler words (which must be zero) must be included in the block to bring it up to five words.

When a file consists entirely of single-record blocks it is called an 'unbatched' file. When some (but not necessarily all) blocks contain more than one record, it is called a 'batched' file.

N.B. When working with External Image Data (see 4.7.4.8) note that if word 12 of the subfile sentinel (4.7.3.3.1) contains the characters B and 5 in the most significant character positions, then there is an implicit extension to the record header. The second word (word 1) contains additional information, the format for which is in section 4.7.4.8.

.2.4. Tape Mark, Qualifier Block, Sentinel

A tape mark is the only one-word block that may be recorded on industry-compatible magnetic tape. When followed immediately by a qualifier block (which usually has 20 words), the two together make a 'sentinel'. A sentinel is used to indicate the beginning, and sometimes the end, of a file, for users' purposes (e.g. to divide a file into sections, for descriptions, for summary totals). When a tape mark is read, Executive sets an indication in the reply word to the 157 (PERI) instruction (see section 4.8.3). The contents of the qualifier block include such information as the type of file, maximum block size, record size, etc. The details vary according to the type of sentinel.

A tape mark is also used to precede a trailer label. With cassette tape, single block sentinels are used, as explained in 4.7.2.2.

Type of Sentinel	Section	Page Number
Start-of-data sentinel	4.7.3.1.1.	7
Start-of-dump sentinel	4.7.3.1.2.	8
End-of-dump sentinel	4.7.3.1.3.	8
User sentinel	4.7.3.1.4	9
Subfile sentinel	4.7.3.3.1	10
End-of-subfile sentinel	4.7.3.3.2	13
Pre-request block sentinel	4.7.4.4.	18
Request block sentinel	4.7.4.4	18

.2.5 Header Label

A header label is the first block recorded on a magnetic tape, and defines that file, and the reel within that file. When a 157 (PERI) instruction is given to open a tape, Executive first examines the standard header labels to find the tape whose name, reel sequence number, and generation number are those specified in the control area.

Tapes with non-standard header labels must be opened by the input

.2.5 Cont'd.

of a GIVE directive to Executive.

Header labels may be of any length, but only the first ten words are used by Executive. The standard format for these is as follows:

Word 0	The characters HDDR
Word 1	Tape serial number
Words 2 to 4	File name
Word 5	Reel sequence number
Word 6	File generation number
Word 7	Security bit / retention period
Word 8	Date written
Word 9	Bits 0 to 5: geographical address of tape deck when header label was written; Bits 6 to 23 are undefined.

.2.5.1. Tape Serial Number

This is a unique number, usually allocated sequentially, which identifies that particular physical tape within the installation's magnetic tape library. It may be up to eight octal digits in length, to a maximum of #37777777.

.2.5.2. File Name

This, with the reel sequence number and the file generation number, identifies the information on the tape. When a new output tape is being written, the file name and file generation number are copied onto the output tape's header label. A standard file name consists of twelve characters; letters, figures, spaces, or the character ' - ' only, and must begin with a letter. Executive, but not most standard software, will accept a file name of twelve characters from the first sixty of the 1900 Series character code.

.2.5.3 Reel Sequence Number

This qualifies the file name and identifies each particular reel within a multi-reel file. Up to 512 reels, numbered from 0 to 511, are permitted within any file name. For a particular multi-reel file, reel sequence numbers may start at any number in ther permitted range, but must follow sequentially.

.2.5.4 File Generation Number

This provides further means of qualifying the file name, entirely within the user's choice. Its main use is in identifying successive generations of a master file that is updated regularly. For instance, it could be a day, week, or month number.

.2.5.5 Retention Period

The least significant 12 bits of Word 7 hold (in binary form) the number of days, from the date written, that the information on the file must be retained for. By adding the retention period to the date written, Executive obtains a 'purge date'. Bit 0 of this word forms the 'security indicator', which, together with the purge date, is used by Executive in certain modes to decide if a tape is valid to be written to. Bits 1 to 11 are reserved.

.2.5.6 Date Written

The date on which the tape was written, expressed as the number of days since 31st December, 1899 (1st January, 1900 being day 1). Executive is given this information afresh every day.

.2.6 Trailer Label

This is the last block on a tape. It is preceded by a tape mark and has twenty words. It indicates whether the last file on the tape continues on another reel (end-of-reel trailer label) or that it is the end of the file, and if so, whether this is a simple file (end-

.2.6 Cont'd

of-simple-file trailer label) or a composite file (end-of-composite-file trailer label).

The processing of trailer labels is done by program, not by Executive.

The formats for industry-compatible and cassette tape are :

	End-of-reel	End-of-simple file	End-of-composite file
Word 0 Bits 0 to 1	11	10	00
Word 0 Bits 2 to 17	Zero	Reserved	Reserved
Word 0 Bits 18 to 23	Zero	Zero	7
Word 1	Count of data blocks *		Reserved
Words 2 to 3	Zero		Reserved
Words 4 to 19	User information		User information

* The count of data blocks excludes sentinels and dump blocks. Unless the file is a simple file, or the last subfile sentinel indicated a simple subfile and there has been no subsequent end-of-subfile sentinel, the count is set to zero.

.3 PRINCIPAL CATEGORIES OF FILES

.3.1 Simple file

A simple file is one that contains no subfiles, and may extend to one or more reels. Certain types of information, i.e. source language programs, unconsolidated semi-compiled programs, and external image data, cannot be kept in a simple file. Information that can be kept in a simple file comprises:

Consolidated semi-compiled programs, which consist of blocks of program in the format described in 4.7.4.3, plus binary programs without overlays as described in 4.7.4.4 (not necessarily in that order).

Binary programs in the format described in 4.7.4.4 and 4.7.4.5.

~~Binary programs with overlays.~~ AL17

Data in Magnetic Tape Housekeeping formats.

Each reel of a simple file begins with a header label. Each reel but the last is terminated by an end-of-reel trailer label, as described in .4.7.2.6. The last reel is terminated by an end-of-simple file trailer label. The only sentinels allowed in a simple file are:-

Pre-Request block sentinels (cassette tape only),

Request blocks, & Supplementary Request Blocks. AL17

Start-of-data sentinels,

Start-of-dump sentinels,

End-of-dump sentinels,

User sentinels.

.3.1.1 Start-of-data sentinel

Where applicable, this normally follows the header label in a simple file (or the subfile sentinel in a simple subfile). The format of the qualifier block is given overleaf.

.3.1.1 Cont'd.

Word		Contents or use
0	Bit 0	One
	Bits 1 to 17	Reserved
	Bits 18 to 23	#02.
1		Maximum block size. Optional, set to zero if not used. Enables software to check adequacy of input buffer.
2 and 3		Zero.
4 to 19		User's information.

.3.1.2 Start-of-dump sentinel

This sentinel immediately precedes a standard non-Executive dump, i.e. one produced by (or for) the Dump and Restart Package.

The format of the qualifier block is:

Word 0	B0 = 1, B18-23 = #03, all other bits indeterminate.
" 1	DUMP
" 2	-nn-
Words 3-4	date: dd/mm/yy
Word 5	-UNI
" 6	Tnn-
Words 7-9	File name
Word 10	-REE
" 11	Lxxx
" 12	-fgn
" 13	xxx-
" 14	REST
" 15	ART.
" 16	Unit no. in binary

.3.1.3 End-of-dump sentinel

This sentinel must immediately follow the last dump block.

The format of the qualifier block is:

Word 0	B0 = 1, B18-23 = #05, all other bits indetminate.
" 1	Block count.
" 2	Zero if dump completed
" 3	Non-zero if dump not completed
" 3	Unit no. in binary
" 4	Core size
" 5-19	Immaterial

.3.1.4 User sentinel

This sentinel may be placed among data blocks anywhere after the start-of-data sentinel and its use is entirely at the user's choice, for example to subdivide a file into sections, for collecting summary totals, or for describing the succeeding data blocks. The format of the qualifier block is:

Word 0 B0 = 1, B18-22 = #04, all other bits indeterminate.
" 1 Data block count (i.e. of the preceding data blocks)
Words2-3 Set to zero
" 4-19 User information

.3.2 Composite file

A composite file is one that contains one or more subfiles, and may extend to one or more reels.

Each reel starts with a header label and is terminated by an end-of-reel or an end-of-composite-file trailer label, as described above.

Between these lie the subfile or subfiles, except that a single binary program, for example a search program, may appear between the header label and the first subfile.

.3.3 Subfile

A subfile begins with a subfile sentinel, which is similar to a header label and is described below. It is terminated by an end-of-subfile sentinel, (See 4.7.3.3.2). Theoretically subfiles may be ~~listed~~ nested to any depth, but in practice limits will be imposed for particular applications. The rules governing the contents of simple files apply also to simple subfiles. AL15

When a continuation reel starts within a subfile, continuation subfile sentinels for all current subfiles occur after the header label.

The type of data may be indicated by the subfile sentinel.

(The references are to where the formats are described):

(.3.3. contd)

Source Program	4.7.4.1
Unconsolidated semi-compiled program	4.7.4.2
Consolidated semi-compiled program possibly including overlays	4.7.4.3
Binary program without overlays	4.7.4.4
Binary ~~overlay~~ program with overlays	4.7.4.5
Uncategorized data in standard records e.g. MTH format	4.7.4.6
" " " non-standard records	4.7.4.7
External image data	4.7.4.8
~~Subfile directory~~	~~4.7.4.9~~
~~Subroutine directory~~ composite subfile.	~~4.7.4.10~~ 4.7.4.9

[AL/12]

AL/10

3.3.1 The sub-file sentinel has a qualifier block in the following format:

Word	Contents or use
0	# 00000006
1	Reserved
2-4	Subfile name. The name may contain letters, digits, spaces, and minus signs, and must start with a letter
5	0 if initial sentinel for this subfile, n if nth continuation sentinel for this subfile (placed on continuation reel, if Subfile incomplete on preceding reel).
6	Subfile generation No.
7	Reserved
8	*Date written
9	*Implicit Programming information:- Characters 0, 1 are defined within the processing system, which is defined by Characters 2, 3 as follows:-

AL/16

Ch 2,3	01	FB	AB	CB	A1
Use:	COSY (PLAN batch compiler)	Fortran batch compiler	Algol batch compiler	Cobol batch compiler	PATSY

If word 9 were 0, this would instruct the processing system to examine word 9 of the sentinels of the contained subfiles.

10	*Bits 0-11: Level of this subfile i.e. number of levels above this: 0 if highest level, *Bits 12-23: Relative depth of this subfile i.e. 0 if this is a simple subfile, > 0 if this subfile is itself composite.

.3.3.1 Cont'd.

11 *Security key. To be further specified to control whether, and by whom, this file may be read from or written to.

12 If zero, uncategorised blocks.

Characters 0 and 1:

A1 Binary program possibly including overlays.

A2 Consolidated semi-compiled programs possibly including overlays.

A3 Unconsolidated semi-compiled programs.

A4 Binary overlay (not available on mag. tape).

B as the first character indicates that the subfile contains standard records as defined in 4.7.2.3.

B1 Uncategorised records.

B2 Source program - normal character set.

B3 Source program - extended character set.

B4 Unconsolidated semi-compiled program in standard records (not available on mag. tape).

B5 External Image Data (see 4.7.4.8).

C1 Composit subfile.

D1
D2 Not yet defined in detail.

Characters 2 and 3:

Language or sub-type. This at present only applies to source programs.

13 Generally set to zero. When working in External Image Data this word may contain at the user's discretion the peripheral type / mode, and can indicate the external form of any data. It may also be the name for a loadable program (it is set and used by certain utility routines).

Magnetic Tape Formats: Industry-compatible and cassette

(.3.3.1 contd)

Word	Contents or use
14	* Bits 0-8: Reserved
	* Bits 9-23: Maximum block size
15	* Bits 0-11: Record size in words, if fixed.
	* Bits 12-23: Blocking factor, i.e. number of records per block except last block. Zero if variable number.
16	* This word gives information about sequencing fields. If present, a sequencing field is a fixed-length character string, and all records of a subfile must be in ascending order in this field.
	Bit 0: 0 if sequencing field is located from the beginning of the record. 1 if sequencing field is located from the end of the record.
	Bits 6-11: Zero if sequencing field is absent, otherwise character length of sequencing field.
	Bits 12-23: Character position of start of sequencing field relative to: (i) start of record if bit 0 of word 16=0 (ii) end of record if bit 0 of word 16=1
17	* This word gives information about end-of-record terminators.
	Bits 0-9: Reserved
	Bits 10-11: Number of characters in record terminator: 00 means no record terminator. 01 means single-character record terminator 10 means double-character record terminator
	Bits 12-17: Zero if 10 is zero, otherwise first record terminator (reading L to R).
	Bits 18-23: Record terminator if bits 10-11=01, or second terminator (reading L to R) if bits 10-11=10.
18,19	Reserved

3.3.2 The end-of-subfile sentinel has a qualifier block in the following format:

Word	Bit	Contents or use
0	B0-1	10
	B2-17	Reserved
	B18-23	0
1		Count of data blocks (excluding sentinels and dump blocks)
2-4		*Subfile name. The name may contain letters, digits, spaces, and minus signs, and must always start with a letter.
5		Reserved
6		*Subfile generation number
7-19		Reserved

(*Note: these are only required on Library tapes holding programs of GEORGE Operating Systems).

3.4 Executive dump file

This type of file is fully described in Part 5, Section 1, particularly Section 1.3 (g).

.4 OTHER FORMATS

.4.1 Source program

Source Program may only be held in a subfile.

There are two standard formats for source program, one allowing only a limited character set but suitable for most languages, the other allowing a larger character set for languages such as ALGOL. Particular compilers may accept both formats or only one. The format used in a subfile is specified in its subfile sentinel.

A source program record corresponds to a punched card or line of paper tape and has the following format:

Word	Contents or use
0	No. of words in record
1 onwards	Source line, including sequence numbers, comments, and any other information acceptable to the compiler.

The source line is held in character form. If characters 0 and 1 of word 12 of the sentinel are B2, the line will be held in the normal 64-character internal code.

Normally such a record occupies an integral number of words and will have been space-filled on the right if necessary. Alternatively, a terminate character may be specified in word 16 of the sentinel.

If characters 0 and 1 of word 12 of the sentinel are B3, the line will be held in the three-shift internal code, and may therefore contain the full papertape character set. Each line is assumed to start in alpha shift. Standard record terminators are;

delta-shift newline
delta-shift line feed
delta-shift form feed
delta-shift carriage return

Magnetic Tape Formats: Industry-compatible and cassette

(4.1 cont'd)

These terminators need not be specified in word 16 of the sentinel. Any other terminator may be so specified. A terminator is not essential.

Source program has a subfile sentinel whose qualifier block is in the following format:

Word		Contents or use
0-11		As for a subfile; see 4.7.3.3.
12		Characters 0 and 1: the characters B2 or B3, according to whether normal or extended character set. Characters 2 and 3: Language. The present allocation of characters is:- A0, A1..A9 ALGOL B0, B1..B9 Operating Systems C0, C1..C9 COBOL E0, E1..E9 EMA F0, F1..F9 FORTRAN G0, G1..G9 GIN M0, M1..M9 Matrix scheme and matrix generators N0, N1..N9 NICOL P0, P1..P9 PLAN S0, S1..S9 Simulation languages (e.g. CSL) Others will be allocated by Programming Support Department.
13		Zero
14	Bits 0-8	Reserved
	Bits 9-23	Maximum block size normally 512, but may be less for a particular compiler.
15		Standard
16, 17		Normally fixed for a particular compiler
18, 19		Standard

.4.2 Unccnsolidated semi-compiled program

Unconsolidated semi-compiled program may only be held in a subfile.

A block may contain 1 or more records. Semi-compiled records are not in the standard format of 4.7.2.3. The first character of the first word of the record denotes a type of record. The second caaracter is a count of the number of words in the record including checksum (for magnetic tape this character must be > 2, whereas for other media it may be 0). The last word of the record is a checksum complement of the whole record.

The maximum length of a semi-compiled record is 18 words, including checksum.

The first record of a block is normally an organisation record of the form:

Word	Bit	Contents or use
0	B0-11	#0702
	B12-23	n-1, where n is the total number of records in the block.
1		Checksum complement of word 0.

Absence of this record implies that the block contains one record only.

A semi-compiled block may contain only semi-compiled records as above, optionally followed by zero words.

Each object program title record and leader title record starts in a new physical block.

(4.2 Continued)
The sentinel has a qualifier block in the following format:

Word		Contents or use
0-11		Standard ubfile contents, see 4.7.3.3
12		Chars 0,1 A3
		" 2,3 Reserved
13		Zero
14	B0-8	Reserved
	B9-23:	Maximum block size, ≤110, normally either 18 or 20
15-17		Zero
18-19		Standard

.4.3 Consolidated semi-compiled program

Consolidated semi-compiled program may be held in a simple file or in a subfile. It consists of some blocks of semi-compiled program and a General Purpose Loader, not necessarily in that order.

For block structure see 4.7.4.2.

The subfile sentinel has a qualifier block in the following format: AL16

Word		Contents or use
0 - 11		Standard subfile contents: see 4.7.3.3
12	Chars. 0,1	A2
	" 2,3	Reserved
13		~~Zero~~ Program name (optional) ~~on cassette tape~~ AL17 AL16
14	B0-8:	Reserved
	B9-23:	Maximum block size, normally 512 on industry-compatible tape because of Type 5 binary blocks in the General Purpose Loader.
15 - 17		Zero
18 - 19		Standard.

.4.4 Binary program without overlays

Binary program without overlays may be held in a simple file, in a subfile, or in an Executive dump file. The sentinel for a subfile has a qualifier block in the following format:

Word	Contents or use
0-11	As in 4.7.3.3
12 Chars 0,1:	A1
" 2,3:	Reserved
13	~~Zero~~ Program name (optional) ~~on cassette tape~~
14	Maximum block size
15-17	Zero
18-19	Standard.

AL17 AL16

The binary program following this, or the Header Label of a simple file, is permanent program with the following format:

Pre-Request Block Sentinel (cassette tape only, see below)
Request Block Sentinel, i.e. Tape Mark and Request Block
Binary program data blocks, Types 5 (not cassette) and 0
Entry block, Type 2 or 3.

For details see Part 5, Section 1, which also describes Executive dump files (note that some of these have no Tape Mark before the Request Block).

A Pre-Request Block Sentinel has a qualifier block in the following format (it is similar to an Overlay identifier block but the latter is not part of a sentinel):

Word	Contents or use
0	#73000076
1	Zero
2	Reserved
3	Program name

The number and contents of other words are undefined.

.4.5 Binary program with overlays

Binary program with overlays may be held in a simple file or in a subfile. The sentinel for subfiles has a qualifier block in the same format as for binary program without overlays.

The format of the binary program following this or the Header Label is:

Overlay units in sequence
Dummy overlay identifier block (Word 1 zero)
Permanent program (as in 4.7.4.4)
Dummy overlay identifier block (Word 1 negative)

An alternative format (available on cassette tape except for standard run-time routines) is:

Dummy overlay identifier block (Word 1 zero)
Permanent program (as in 4.7.4.4)
Overlay units in sequence
Dummy overlay identifier block (Word 1 negative)

An overlay unit consists of an overlay identifier block followed by a number of Type 0 or 5 binary blocks and a Type 2, 3, or 4 entry block (see Part 5, Section 1.3).

An overlay identifier block has the following format:

Word	Contents or use
0	#73000076
1	B0-8 Overlay area no. B9-23 Overlay unit no.
2	Unit chain i.e. the number of blocks from the previous identifier block to the current identifier (inclusive)
3	Reserved
4	Reserved

.4.6 Uncategorised data in standard records

Uncategorised data comprising standard records (as defined in 4.7.2.3.), may be held in a simple file or simple subfile. In the latter ~~either~~ case its subfile sentinel has a qualifier block in the following format:

Word	Contents or use
0 to 11	As in 4.7.2.3.
12 Bits 0 to 11	B1
Bits 12 to 23	Reserved.
13	Zero.
14	Maximum block size.
15 to 17	Zero.
18 and 19	Standard.

.4.7 Uncategorised data not in standard records

Data not split into standard records, and not otherwise categorised, may be held in a simple file or simple subfile. In the latter case its subfile sentinel has a qualifier block in the following format:

Word	Contents or use
0 to 11	As in 4.7.2.3.
12	Zero.
13	Peripheral type / mode, where applicable.
14	Maximum block size.
15 to 17	Zero.
18 and 19	Standard.

.4.8 External Image Data in Standard Records

Data in a form that is simply an image of data held on some other type of peripheral (e.g. paper tape) may be held in a subfile. Its subfile sentinel has a qualifier block in the following format:

.4.8 Cont'd.

Word	Contents or use
0 and 1	As in 4.7.2.3.
2 to 4	Document name.
5 to 11	As in 4.7.2.3.
12	B5 in most significant character positions.
13 Bits 0 to 8	Peripheral type number } This is optional.
Bits 9 to 23	Mode }
14	Maximum block size.
15 to 17	Zero.
18 and 19	Standard.

Note that External Image Data may be uncategorised data not in standard records, in which case the format is as specified in 4.7.4.7. Also, when using External Image Data, there is an extension to record headers (see 4.7.2.3) in the following format:

Word	Contents or use
0	Number of words in record (standard).
1 Bits 0 and 1	The position of the last character in the last word of the record. Thus by using the formula 4n + c - 11, the number of characters of object information can be determined, where n is the number of words and c is the last character position in the last word.
Bits 2 to 11	The Document identifier. Any number between 0 and 1023, used to distinguish between mixed streams in a single subfile.
Bits 12 to 17	The mode (optional if in word 13 of start-of-data sentinel).
Bits 18 to 23	Line printer control character (where applicable).

(4.8 cont.)

There are also Document User-Sentinels to hold additional information about the document(s) when working with External Image Data:

Word	Contents or use
0 to 4	As in 4.7.3.1.4.
4 to 6	Document name.
7 and 8	Date in characters (optional)
9 and 10	Zero
11 Bits 0 to 5	Peripheral type (optional if in subfile sentinel).
Bits 6 to 11	Unit number.
Bits 12 to 17	Zero.
Bits 18 to 23	If zero, then this user sentinel precedes all of the records of the document. If one, then this user sentinel succeeds all the records of the document. All other values are non-standard.
12	Document identifier (optional).
13 to 19	Additional user information.

4.9 Composite Subfile

A composite subfile is a subfile which contains further subfiles, which may themselves be simple or composite. It may extend to one or more reels. It must contain nothing but subfiles, so that all data in a composite subfile must be contained in simple subfiles.

It is terminated by an end-of-subfile sentinel in which word 1, the block count, is zero.

Magnetic Tape Devices

(Industry Compatible and Cassette)

.1 INTRODUCTION

.1.1 Beginning and End of Tape indications.

.1.2 Write Permit Ring / Button.

.2 APPLICABLE PERIPHERAL CONTROL INSTRUCTIONS

.2.1 156 (ALLOT) Instruction

.2.2 157 (PERI) Instruction : Summary of Modes.

.2.2.1 Control Area (Modes 0, 1, 2, 3, 4, 5, 6 and 7)

.2.2.2 " " (Open File Modes 64,128,192,256,320 and 384)

.2.2.3 " " (Close File Modes 512 and 519)

.2.2.4 Summary of File Security checks for Open File Modes.

.2.2.5 Action of Open File Modes (256, 320, 384, 14336)

.2.2.6 " " Close File Modes (512, 519)

.2.2.7 File Extension Procedure.

.2.3 Industry Compatible Tape - Property Codes.

.3 TAPE UNIT CHARACTERISTICS

Magnetic Tape Devices
(Industry Compatible and Cassette)

.1 INTRODUCTION

This section describes the programming aspects of Industry Compatible (both 7- and 9-track) and Cassette loading magnetic tape.

Provided the programmer uses the 'range compatible' order code, the object program will not distinguish between 7- and 9-track tape. Programming differences between Industry Compatible and Cassette Tape are not numerous and are clearly shown in this section.

.1.1 Beginning and End of Tape indications

(a) Industry Compatible Tape

At least 10 feet from the beginning of a reel of tape is a permanent 'load point marker' which is sensed by the hardware when loading and rewinding tapes. The first block recorded on a tape is the header label (format described in section 4.7) which serves to identify the information on that tape.

25 feet from the end of the tape is a permanent 'end of tape marker' which is sensed by the hardware and signalled via Executive to the object program in the reply word of a 'write' or 'write tape mark' instruction. It is then the programmer's responsibility to read or write the trailer label (format described in section 4.7). Note: on some early I.C.T. tapes and on some I.B.M. tapes the E.O.T. marker is fitted only 15 feet from the end of the tape. On such tapes an additional E.O.T. strip should be attached 25 feet from the end of the tape.

(b) Cassette

Each cassette holds a continuous loop of magnetic tape which is divided into 8 parallel recording tracks along the tape numbered from 0 to 7. Tracks begin and end at the joint called the 'Align Point', which lies between two Align Point Markers.

Recording takes place along each track in turn, the track number being incremented at the Align Point by Executive after each circuit of the tape. All blocks written on Cassette Tape are assigned a 'Block Sequence Number' which is held in the 'Block Sequence Word' (a special word written by Executive as word - 1 of the block); On reading and writing, this word is copied by Executive to word -1 of the data area. The format is as follows:- (see next page)

(.1.1 (b) continued)

Bit 0 : 0 or 1 (See note below)

Bit 1 : 1 (Except in any block written in the overlap area on track 7, when it is zero)

Bit 2 : 0 (Except in the Guard Block, when it is 1)

Bits 3 to 23 Block Sequence Number. (Indeterminate in Guard Blocks)

Tape marks are not written on Cassette Tape. Instead, the block sequence word is considered as a tape mark if it is set negative; the following words in that block then function like a qualifier block. If the block sequence word is set positive, the rest of the block is treated as a normal data block.

Following the Align Point marker on track 0, the header label (with block sequence number zero) is written in the same format as for Industry Compatible Tape. When the 'write' operation has ended, after the last block has been written, the object program should issue an 'Align Tape' or 'Close File' instruction, which will cause Executive to write a special 'Guard Block'. The purpose of this block is to enable subsequent file extension to be performed without using lengthy routines to position the tape before the last valid block written. The format of the Guard Block is undefined (except for its block sequence word).

Writing may take place up to the end of track 6; Executive then erases an 'overlap area' at the beginning of track 7, sufficient to hold one 20 word block and two 512 word blocks. Writing may then continue up to the end of track 7, when 'End of Tape' is signalled to the object program. The programmer may then use the erased area at the beginning of track 7 as necessary.

.1.2 Write Permit Ring / Button

Each reel of Industry Compatible tape may be fitted with a 'write permit ring' which cancels a write inhibit switch on the deck and allows writing to take place to that reel. The same function is performed on Cassette tape with a 'write permit button'.

There are other conditions which have to be satisfied before writing can take place; these are detailed under .2.2.4 'Summary of File Security Checks for Open File Modes'.

.2 APPLICABLE PERIPHERAL CONTROL INSTRUCTIONS

150 (SUSBY) : Suspend program member if specified tape unit busy.
151 (REL) :)
152 (DIS) :) Illegal
153 (-) :)
154 (CONT) : Load more 'binary' program from specified tape.
155 (SUSDP) : Illegal on Cassette Tape; dump whole program in binary form on specified Industry Compatible Tape.
156 (ALLOT) : Identify tape deck / station.
157 (PERI) : Activate specified tape.

.2.1 156 (ALLOT) Instruction

The restricted (identify only) function is available with this instruction.

Property codes may be identified on Industry Compatible Tape, but not on Cassette Tape.

The reply in word 9 of the program has the following form:-

Bit 0 = 0 : The required tape has (previously) been assigned and the remaining bits are meaningful.
= 1 : The required tape has not (previously) been assigned and the remaining bits of this word are indeterminate.
Bits 1 to 8 : Indeterminate.
Bit 9 = 0 : The properties system is not implemented on this processor (or device) and thus bits 10 to 17 are indeterminate. (Bit 9 is always zero for Cassette Tape)
= 1 : The property system is implemented and bits 10 to 17 are meaningful.
Bits 10 to 17 : Contain the property code of the tape deck concerned. (See .3 'Industry Compatible Tape - Property Codes'). Bits 10 to 17 are always indeterminate for Cassette Tape.
Bits 18 to 23 : Contain the absolute unit number of the device.

.2.2 157 (PERI) Instruction : Summary of Modes

Basic Modes			Function	
Decimal	Octal		Industry Compatible	Cassette
0	0	:	Read one block in a forward direction.	
1	1	:	Write one block	
2	2	:	Read one block in reverse direction. (Not on 1971, 1972, 1973; 2504/5 when attached to 1901/2/3/4/5 or 9.	Erase a section of tape
3	3	:	Backspace one block.	Illegal
4	4	:	Skip forward to gap immediately beyond next tape mark.	Illegal
5	5	:	Write a Tape Mark.	Illegal
6	6	:	Skip backward to gap immediately beyond next tape mark. (Not on 1971)	Illegal
7	7	:	Rewind to Load Point Marker, (keeping tape assigned to program) ready to read or write header label.	Align tape,
64	100	:	Open named tape for reading. (No check for Write Permit Ring / Button).	
128	200	:	Open named tape for reading. (Write Permit Ring / Button must be absent).	
192	300	:	Open named tape for reading and writing. (Write Permit Ring / Button must be present).	
256	400	:	Create named output tape.	
320	500	:	Relabel tape.	
384	600	:	Create 'Scratch' tape.	
512	1000	:	Close tape file, leaving on line to processor	
519	1007	:	Close tape file, leaving off line.	

(.2.2 continued)

Additive Modes			Function	
Decimal	Octal		Industry Compatible	Cassette
8192	20000	:	As defined in Section 6. Permitted only with basic modes 0,1,2,3,4, 5 and 6. Has 'Null' effect in all manual operating environments except E4BM and E6BM and perhaps EX2M.	Has 'Null' effect.
12288	30000	:	'Own Monitoring', as defined in Section 6. Permitted only with basic modes 0,1,2,4,5 and 6.	Has 'Null' effect.
14336	34000	:	'Own Monitor Open File' operation. Permitted only with basic modes 64, 128, 192, 256 and 384. See '.2.2.5 Action of Open File Modes'.	

The format of the control area varies with the mode used. In the following subsections, therefore, the format is considered separately for modes 0 to 7 (4 word control area), modes 64 to 384 (9 word control area) and for modes 512 and 519 (4 word control area).

.2.2.1 Control Area (Modes 0,1,2,3,4,5,6 and 7)

Word 0 (Type and Mode)

Bits 0 to 8 : Type = 5 for Industry Compatible Tape.
= 12 for Cassette Tape.

Bits 9 to 23 : Mode.

(.2.2.1 continued)

Word 1 (Reply)

Bit 0 : Operation in progress. (All other bits indeterminate.).

Bit 1 : Long block - modes 0 and 2 only; indeterminate in other modes and also in mode 2 for Cassette Tape.

Bit 2 : In-flight corrected error - modes 0 and 2, 9-track Industry Compatible Tape only; otherwise indeterminate.

Bit 3 : Error; indeterminate in modes 3 and 7.

Bit 4 :

Mode	Industry Compatible	Cassette
0	Tape Mark	End of Tape
1	End of Tape	"
2	Tape Mark	"
3	Indeterminate.	Not applicable
4	"	"
5	End of Tape	"
6	Indeterminate	"
7	"	Indeterminate

Bit 5 : Indeterminate.

Bits 6 to 23 : Address of next word of data area which would have been addressed if transfer had not terminated. Indeterminate in modes 3 to 7 (and, for Industry Compatible Tape, in modes 0 and 2 if Bit 4 is set).

Word 2

Size in words of data to be read/written (modes 0,1 and 2 only). The maximum is 32767* words for Industry Compatible Tape and 512 words for Cassette Tape. (The programmer should bear in mind that when 'end of tape' is detected on Industry Compatible Tape, there is only sufficient space to hold 4096 words).

*Except for 1971 tape decks attatched to 1904E/05 and 1906/07 processors on which the maximum is 8191 words.

(.2.2.1 (Word 2) continued)

In mode 1 a minimum of 5 words must be transferred for Industry Compatible Tape. With Cassette Tape, a count value of zero is valid, and results in the block sequence word only being read/written.

In modes 3 to 7 the contents of this word are not significant, but it must lie in the control area.

Word 3

Address of first word of data area (Industry Compatible) }
Address of second word of data area (Cassette) } Modes 0,1,2

For Cassette Tape, the value of the count refers to the area starting at the address quoted in Word 3, which is the address:

(into which the word following the block sequence word is read)
(or from " " " " " " " " written)

The block sequence word is read into (or written from) the word whose address is one less than the word quoted in word 3 by the object program.

In modes 3 to 7 the contents of word 3 are not significant, but it must lie in the control area.

.2.2.2 Control Area (Open File Modes 64,128,192,256,320 and 384)

Word	Significance Mode :	64	128	192	256	320	384
0	Type / Mode.	P	P	P	P	P	P
1	Primary Reply Word (and Tape Serial No)	R	R	R	R	R	R
2,3,4	File Name	P	P	P	P	P	E
5	Reel within file (from 0 to 511)	P	P	P	P	P	E
6	File Generation	P/R	P/R	P/R	P	P	E
7	Security Bit / Retention Period	R	R	R	P	P	E
8	Date written	R	R	R	E	E	E

(.2.2.2 continued)

P : Item to be preset by object program (final contents = initial contents).

R : Reply provided by Executive (initial contents not significant)

P/R : Preset and reply, see 'Word 6 (File Generation)' below.

E : Word ignored by Executive, which writes standard information into the corresponding part of the header label (final contents = initial contents).

Word 1 (Primary Reply Word)

Bit 0 = 0 : Action called for has been performed successfully and remaining bits are meaningful.

= 1 : Action called for could not be performed satisfactorily and remaining bits are indeterminate.

Bits 1 to 23 : Contain 'Tape Serial Number' of tape concerned.

Word 6 (File Generation)

When opening a named tape, the object program has two options:-

(a) To call for a specific generation (other than zero) by specifying the required generation in this word.

(b) To call for 'any generation', by setting this word to zero.

In both cases, the final contents of this word indicate which generation of the file was actually opened.

Word 7 (Security Bit and Retention Period)

This word in both header label and control area takes the following form:-

Bit 0 : Security bit.

Bits 1 to 12 Undefined.

Bits 12 to 23 Retention period in days (binary count).

.2.2.3 Control Area (Close File Modes 512, 519)

Word 0 Type / Mode.

Word 1 Contents not significant, final contents indeterminate.

Words 2 and 3 Contents not significant, but must lie within the program's area. (Final contents = initial contents)

.2.2.4 Summary of File Security checks for Open File Modes

The table below shows which conditions are checked by Executive. Unless otherwise stated, the conditions listed must all be satisfied for a file to be opened successfully.

Mode / Security Check	Mode 64 (* 100)	Mode 128 (* 200)	Mode 192 (* 300)	Mode 256 (* 400)	Mode 320 (* 500)	Mode 384 (* 600)
File Name corresponds?	yes	yes	yes	--- Not examined ---		
Reel within File corresponds?	yes	yes	yes	--- Not examined ---		
File Generation corresponds?	yes	yes	yes	--- Not examined ---		
Security Bit	Not set@	Not examined	Not set@	Not set	Not examined	Not set
Write Permit Ring/Button	Absent@	Absent	Present	Present	Present	Present
Retention Period expired?	yes@	Not significant	yes@	yes	Not significant	yes
Property Request satisfied?	--- Not applicable ---			yes (with EX2M E4BM,E6BM)	Not applicable	yes (with EX2M E4BM,E6BM)

@ These conditions are optional; at least one of them must be satisfied for that mode (with E4BM and E6BM only).

Provided the action called for has been successfully completed, these modes leave the tape positioned ready for the object program to read or write the block immediately following the header label.

Any 'Open File' instruction to a program's unit number which is currently 'open' will be illegal, except on Cassette Tape used with Handswitch Executives. (On Cassette Tape with Handswitch Executives the effect is undefined, and may not be monitored as illegal).

.2.2.5 Action of Open File Modes (256, 320, 384, 14336)

Mode 256 (*400) 'Create named output tape'

The tape assigned as a result of this mode will have a new header label written on it in the following format:-
(Note: this action renders all previously recorded information inaccessible)

Word - 1	*20000000 (Block sequence word, Cassette Tape only)		
Word 0	The four characters 'HDDR'		
Word 1	Tape Serial Number	Copied from previous label.	
Word 2,3,4	File Name	As supplied by object program, in control area.	
Word 5	Reel number within file		
Word 6	File Generation number		
Word 7	Security Bit and Retention Period		
Word 8	Date written, (Current date)	Supplied by Executive	
Word 9	Engineering information		

On Industry Compatible (but not on Cassette) Tape, 'Properties required' may be specified when the 'replaced x ' form of the 157 instruction is used (in the same way as can be done in the 156 instruction when calling for a slow device). However, as there is no 'geographical and properties' reply to a 157 instruction, a 156 instruction should be used after assignment. For the relevant property codes, see 3. 'Industry Compatible Tape - Property Codes'.

Mode 320 (*500) 'Relabel Tape'

For a tape to be relabelled as a result of this mode, in addition to satisfying the checks outlined above (.2.2.4), it must already have a standard header label and have been previously assigned to the program.

This mode causes the tape to be 'rewound' (Industry Compatible) or 'aligned' (Cassette) and a header label written in the same format as for mode 256 (*400) above. Any previously recorded information is rendered inaccessible.

Mode 384 (*600) 'Create Scratch Tape'

The tape assigned as a result of this mode will have a new header label written on it in the following format:-

Word -1		*20000000 (Block Sequence Word, Cassette Tape only)	
Word	0	The four characters 'HDDR'	
Word	1	Tape Serial Number	Copied from previous label
Word	2,3,4	The 12 characters 'SCRATCH TAPE'	Supplied by Executive
Word	5	Zero	
Word	6	Zero	
Word	7	Zero	
Word	8	Current Date	
Word	9	Engineering information	

All previously recorded information is rendered inaccessible.

As with mode 256 'properties required' may be specified for Industry Compatible tape when using the replaced X form of the instruction. For details of property codes see.3.

Additive Mode 14336 (*34000) 'Own Monitor Open File'

The effect of this mode is to allow the issuing program member to continue even if the action demanded by the 'Open' mode cannot be satisfactorily completed. It does not inhibit suspension due to irrecoverable errors and inoperability during the search for a tape; it only allows the program member to continue in the simple 'no suitable tape' condition.

.2.2.6 Action of Close File Modes (512,519)

Both modes cause the termination of the assignment to the program of the tape concerned, and the tape to be rewound (Industry Compatible) or aligned (Cassette). In both cases, as far as the object program is concerned, the action is complete as soon as the next instruction is reached, and there is no significant reply.

With Cassette Tape, if the previous instruction was a 'write', Executive will, before aligning the tape, write a special guard block on the tape in order to simplify file extension procedure. (See .2.2.7).

.2.2.7 File Extension Procedure

When it is required to read up to some known block and then write further blocks, or overwrite blocks written during the current writing pass, the object program must ensure that:-

(a) The first 'write' (or 'write tape mark') instruction causes a block which is known to be correctly recorded to be overwritten. (Note: A tape mark, on Industry Compatible Tape, or a guard block (or erased area on track 7) on Cassette Tape may be overwritten in this way).

(b) The previous instruction to the tape must have caused it to have moved forward (Industry Compatible Tape only).

Failure to observe these requirements may render the extended file unreadable.

.2.3 Industry Compatible Tape - Property Codes

The property codes which appear in Bits 10 to 17 of the program's word 9 in response to a 156 order have the following structure:-

Bit 10	:	= 0	:	7 track tape
		= 1	:	9 track tape
Bits 11 to 12	:			Speed of tape movement
Bit 13	:			Availability of "Read Reverse" (mode 2)
Bit 14	:			Availability of "Skip backwards to Tape Mark" (mode 6)
Bit 15	:			Undefined
Bits 16 to 17	:			Data packing density

(.2.3 continued)

Specific Codes	Bit 10 to Bit 17
1970/1974 working at 200 rpi	: 01111X00
1970/1974 " " 556 rpi	: 01111X01
1974 " " 800 rpi	: 01111X10
1971 " " 200 rpi	: 00000X00
1971 " " 556 rpi	: 00000X01
1972/1973 " " 200 rpi	: 00101X00
1972/1973 " " 556 rpi	: 00101X01
1973 " " 800 rpi	: 00101X10
2504, in environments providing "read reverse"	: 10111X00
2504, in environments not providing "read reverse"	: 10101X00
2505, in environments providing "read reverse"	: 11011X00
2505, in environments not providing "read reverse"	: 11001X00

(X = contents indeterminate)

(rpi = rows per inch)

.3 TAPE UNIT CHARACTERISTICS

Tape Unit	Type Number	Transfer Rate* (1000 Chars/Sec)	Standard Packing Density (Chars/Inch)	Optional Packing Density (Chars/Inch)
1970	5	66.0	556	200
1971	5	20.8	556	200
1972	5	41.7	556	200
1973	5	60.0	800	200 or 556
1974	5	96.0	800	200 or 556
2504	5	80.0	2133	
2505	5	160.0	2133	

Tape Unit	Type Number	Transfer Rates (1,000 Chars/Sec)	Packing Density (Chars/Inch/Track)
2501	12	6 to 20 nominally 10**	56 (all ones), 83 (equal mix), 167 (all zeroes).

* Maximum instantaneous rate assuming no inter-block or other gaps.

** One bits and zero bits are recorded at different densities, thus the transfer rate will depend on the bit pattern. Nominal figures are calculated on the basis of an equal mix of zeroes and ones.

PART 3 CONTENTS Page No.

Section 0 EXECUTIVE

0.1	INTRODUCTION	1	
0.2	MULTIPROGRAMMING	2	
0.3	PERIPHERAL OPERATIONS	4	
0.4	LOADING AND DUMPING PROGRAMS	4	
0.5	EXTRACODES	4	
0.6	OPERATOR COMMUNICATIONS	7	
0.7	RESERVED LOCATIONS	7	AL 7

Section 1 EXECUTIVES

.1 INTRODUCTION

(a) Nature and purpose of Executive

Executive is a special-purpose program (which for all practical purposes may be considered to be part of the hardware) consisting of a number of separate modules which can be assembled in variable packages, according to the hardware specification of a particular 1900 system.

This Executive program is used by all computers in the 1900-Series and is always present in a protected area of the store to control users' programs. Its size varies from one installation to another, one factor being the number of peripheral devices attached to the processor. (Details of Executive sizes are given in 1900 series computer guide No. 1053).

The general purpose of Executive is to take over from both programmer and operator the execution of a number of routine tasks, and to organize the running of each program in the most efficient manner possible.

Executive's main functions are:

i) Control of multiprogramming and subprogramming activity on those processors which have these facilities.

ii) Control of peripheral devices and execution of peripheral transfer requests.

iii) Loading and Dumping of Programs.

iv) Provision of Extracode Facilities.

v) Communication with the Operator and execution of Operator directives.

(b) Entries to Executive

An event which causes an object program to be left and Executive entered is termed an entry; there are two principal types, voluntary and involuntary.

A voluntary entry is one which occurs at a set predictable point in a program being caused by, for example, a program instruction which is actually executed by an extracode.

An involuntary entry is one which is caused by an occurrence which cannot be readily predicted by the programmer; it may be caused by operator action or by the end of a peripheral transfer, amongst other causes.

.2 MULTIPROGRAMMING

(a) 1902/3

1902/3 does not have full multiprogramming facilities. However 1902/3 with 16K or 32K core store does permit a Real Time Program to be run with an ordinary program. Further details of this facility will be published when available.

(b) 1904/5/9

These can have up to 4 programs in the machine at one time. Each of these programs can have up to 2 subprograms. Only the most significant 9 bits of the address are held in the DATUM and LIMIT registers so programs are rounded up to the nearest multiple of 64.

(c) 1906/7

These can have up to 16 programs each of which can have up to 3 subprograms. The DATUM and LIMIT registers hold the most significant 16 bits of the 22-bit address, so again, programs are rounded up to the nearest multiple of 64.

(d) Priorities (1904/5/6/7/9)

Each program is loaded with a priority assigned by the programmer. ~~01~~ 00 is regarded as the lowest priority and 99 as the highest. [AL/12]

Executive can be requested by the operator to type out the priorities of the programs currently in the machine (PRIORITY). In addition, the operator can ask Executive to revise the priority of a program (REVISE).

Executive tries to enter the highest priority program whenever possible. If two programs have the same priority Executive makes an arbitrary choice between them.

When Executive has formed a queue of instructions waiting to be given to a multi-unit channel e.g. a magnetic tape cluster, it disregards program priorities and initiates the operations in the order in which they are received.

(e) Store Protection

Prior to entering a program Executive on the 1904/5/6/7/9 sets the DATUM and LIMIT hardware registers for that program. In 1901/2/3 AL/16 LIMIT is wired in rather than existing as a hardware register. Before an instruction which addresses the core store is obeyed,

hardware adds the value of DATUM to the operand field, after modification, if appropriate, and checks that the result does not exceed the value of LIMIT.

(f) Program Deletion

When a program is deleted, either by the operator or by program instruction, the peripherals become available for other programs.

So that all the spare core is available as one continuous area when a program is deleted, Executive on the 1904/5/6/7/9 moves all programs in higher locations down to fill the gap and changes the values for DATUM and LIMIT accordingly. Before moving the programs, Executive completes any transfers which are in progress or queuing. This process can take several seconds.

.3 PERIPHERAL OPERATIONS

A program/subprogram communicates its peripheral action requests to Executive by means of the Group 15 orders.

Further details of these can be found in PART 2 of this manual.

.4 LOADING AND DUMPING OF PROGRAMS

Part 5 of this manual deals fully with this aspect of Executive.

.5 EXTRACODES

(a) General

An Extracode is a routine, contained in Executive, which carries out a defined function which, for various reasons (such as hardware cost on a small processor), is not provided in the logic design of the central processor. To the Object Program, Extracodes appear

to be normal instructions, thus programmers need not concern themselves about Extracodes any more than about any other instruction in the order code.

(b) Extracodes on the 1900 series

On the 1900 series the functions which are performed by Extracode vary between different processors and fall into several groups as follows:

i) Peripheral Extracodes (15 group)
The instructions which the Object Program uses to control peripheral devices are, on all processors, carried out by Extracode. Full details will be found in Part 2 (Peripheral Operations)

ii) Subprogram Control Extracodes (162, 163 and 164).
Details will be found in Part 4 (Subprograms)

iii) Programmer to Operator Communications (160 and 161).
Details will be found in Part 1 (Order Code) and Part 6 (Operator Communications).

iv) Program/Executive Communication (165)
The 165 (GIVE) Extracode enables the Object Program and Executive to communicate with each other.

v) Extracodes which, on some processors, are substitutes for hardware functions provided on other processors (see below).

(c) Extracodes in lieu of Hardware

Certain processors, mainly those at the 'small' end of the series, perform some of the more complicated functions by Extracode, thus saving on the cost of the Central Processor

hardware, at the expense of extended run times for Object Programs. The definitions of such functions remain as defined in the 1900 series Order code (see Part 1). The following table shows which functions are, sometimes, performed by Extracode and on which processors.

Function(s)	1901	1902/3	1904	1905/9	1906	1907
040 - 047	X	X	H	H	H	H
076	NA	NA	NA	NA	XX	H
111 and 113	X	X	H	H	H	H
114 and 115	Q	XX	H	H	H	H
126 and 127	X	X	H	H	H	H
130 and 131	Q	XX	XX	X	XX	H
132 - 135	Q	XX	XX	FH	XX	FH
136 and 137	Q	XX	XX	H	XX	H

References: NA - function not available on this model of processor.

H - function performed by hardware

Q - function not available on the 1901 but may be provided by Extracode, at a later date, under category 'XX'

X - function performed by Extracode

XX - function performed by Extracode provided that a special Executive is used. Otherwise it is not available.

(Continued over)

FH - function performed by Autonomous Floating Point Hardware Unit.

.6 OPERATOR COMMUNICATIONS

Part 6 is devoted to this subject.

.7 RESERVED LOCATIONS

Part 0 Section 6 is devoted to this subject.

CONTENTS PAGE

Introduction and abbreviations used	2
EX1H Executive	3
EX1T "	4
EX1V "	5
E1HS "	6
E1TS "	7
E1DS "	8
E1MS "	9
EX2L "	10
EX2$ "	11
EX2M "	12
EX2V "	13
E3TS "	14
E3TE "	15
E3TM "	16
E3DS "	17
E3DE "	18
E3DM "	19
E3DG "	20
E4BM "	21
E4G3 "	23
E6BM "	24
Dual Processor Executive	26
Paged Executive	27
E6G3 "	28

Introduction

This section gives detailed descriptions of all available Executives and projected Executives for use with the 'A' Series central processors. It should be noted that the lists given for permissible peripheral connections to central processors using the stated Executives is only for the standard Executives; if further facilities are required then a non-standard Executive may be requested. The Hardware Plan describes which peripherals may or may not be connected to which Executive whether standard or non-standard.

Abbreviations used

E.D.S.	Exchangeable Disc Store
T.E.D.S.	Twin Exchangeable Disc Store
F.D.S.	Fixed Disc Store
O.C.R.	Optical Character Reader
O.M.R.	Optical Mark Reader
U.D.T.	Universal Document Transport
V.D.U.	Visual Display Unit
S.I.	Standard Interface

EX1H

Processor: 1901

Description of Executive:

Single program with handswitch control.

Optional Features:

Floating Point facilities.

Configuration Restrictions:

Minimum store requirement of 4K words.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	1911; 2102; 2103.
Card Punches	4	1920/2; 1922/1; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1932/1, /2; 1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
1004 Local Link	2,3,4	7201.
Cassette Tape	12	2501.

EX1T

Processor: 1901

Description of Executive:

Single program, typewriter controlled.

Optional Features:

Floating Point, Automatic Operator facilities.

Configuration Restrictions:

Minimum store requirement of 8K words; the configuration must have a console typewriter.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	1911; 2102; 2103.
Card Punches	4	1920/2; 1922/1; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1932/1, /2; 1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
1004 Local Link	2,3,4	7201.
Magnetic Tape (7-track)	5	1971.
Cassette Tape	12	2501.
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
U.D.T./O.C.R. + O.M.R.	23	8101 + 8201 + 8301.
Interrogating Typewriter	10	7070 + 7071.

EX1V

Processor: 1901

Description of Executive:

Single program, typewriter controlled, E.D.S. overlaid, Automatic Operator.

Optional Features:

Floating Point facilities.

Configuration Restrictions:

Minimum store requirement of 8K words, must have console typewriter, and at least 2 E.D.S. drives.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	1911; 2102.
Card Punches	4	1920/2; 1922/1; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1932/1, /2; 1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
E.D.S.	6	2801/0, /2; 2802/0, /2.
Magnetic Tape (7-track)	5	1971.
Cassette Tape	12	2501.
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
U.D.T./O.C.R. + O.M.R.	23	8101 + 8201 + 8301.

E1HS

Processor: 1901A

Description of Executive:

Single program Executive for Card/Paper Tape systems controlled by handswitches.

Optional Features:

Floating Point facilities.

Configuration Restrictions:

Minimum store requirement of 4K words, maximum of 8K words.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	2102; 2104; 2105; 2106.
Card Punches	4	1920/2.
Paper Tape Readers	0	1915/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	2404; 2405.

E1TS

Processor: 1901A

Description of Executive:

Single program Executive for Magnetic Tape systems, controlled by console typewriter.

Optional Features:

Automatic Operator, Floating Point facilities.

Configuration Restrictions:

Minimum store requirement of 8K words, maximum of 16K words. The configuration must have a console typewriter.

Permissible peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Punches	4	1920/2.
Paper Tape Readers	0	1915/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Graph Plotter	20	1934/1 to 6.
1004 Local Link	2,3,4	7201.
Magnetic Tape (7-track)	5	1971.
Line Printers	2	2404; 2405.
Card Readers	3	2105; 2106.

E1DS

Processor: 1901A

Description of Executive:

Single program Executive for E.D.S. systems, with Automatic Operator facilities included. Controlled by console typewriter.

Optional Features:

Floating Point facilities.

Configuration Restrictions:

Minimum of 8K words of store, must have one E.D.S. drive and a console typewriter.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Punch	4	1920/2.
Paper Tape Reader	0	1915/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Graph Plotter	20	1934/1 to 6.
1004 Local Link	2,3,4	7201
E.D.S.	6	2801/0,/2; 2802/0, /2.
Magnetic Tape (7-track)	5	1971.
Line Printers	2	2404; 2405.
Card Readers	3	2105; 2106.
Various single channel Devices (Uniplexer)	10	

E1MS

Processor: 1901A

Description of Executive:

Single program Executive for twin E.D.S. systems, with Automatic Operator and console typewriter included.

Optional Features:

Floating Point facilities.

Configuration Restrictions:

Minimum of 8K words of store, must have Twin E.D.S. drives and a console typewriter.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Punch	4	1920/2; 2105; 2106.
Paper Tape Reader	0	1915/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printer	2	2404, 2405.
Graph Plotter	20	1934/1 to 6.
1004 Local Link	2,3,4	7201.
Magnetic Tape (7-track)	5	1971.
T.E.D.S.	6	2820; 2821.
Various single channel devices (Uniplexer)	10	

EX2L

Processor: 1902

Description of Executive:

Single program with limited facilities.

Optional Features:

Floating Point facilities.

Configuration Restrictions:

Minimum store requirement of 4K words. If store size is only 4K words, then only the 'slow' peripherals may be attached.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	1911; 1912; 2102; 2103.
Card Punches	4	1920/2; 1922/1; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Line Printers	2	1931/1, /2; 1932/1, /2; 1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
1004 Local Link	2,3,4	7201.

EX2S

Processors: 1902, 1903.

Description of Executive:

Single program.

Optional Features:

Floating Point, Automatic Operator facilities.

Configuration Restrictions:

Minimum store requirement of 8K words.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	1911; 1912; 2102; 2103.
Card Punches	4	1920/2; 1921/1; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1931/1,/2; 1932/1,/2; 1933/1,/2,/3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
1004 Local Link	2,3,4	7201.
Magnetic Drums	9	1962; 1963 (1903 only); 1964 (1903 only)
Magnetic Tape (7-track)	5	1971; 1972; 1973.
Cassette Tape	12	2501.
1100 Mag. Tape to S.I.	59	1198; 1199.
Document Sorter/Reader	57	8500.
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
U.D.T./O.C.R. + O.M.R.	23	8101 + 8201 + 8301.
Interrogating Typewriter	10	1938/1; 7070 + 7071.

EX2M

Processors: 1902, 1903.

Description of Executive:

Multi-program, with sub-programming facilities (up to 2 programs, each of which may consist of up to 3 members, can occupy the processor at one time).

Optional Features:

Floating Point, and Trusted Program facilities.

Configuration Restrictions:

Minimum store requirement of 16K words.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	1911; 1912; 2102.
Card Punches	4	1920/2; 1922/1; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1931/1, /2; 1932/1, /2, 1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
M.C.F.	7	1958; 1959.
E.D.S.	6	2801/0, /2; 2802/0, /2.
Magnetic Drums	9	1962; 1963 (1903 only); 1964 (1903 only).
Magnetic Tape (7-track)	5	1971; 1972; 1973.
U.D.T./O.M.R.	23	8101 + 8301
Interrogating Typewriter	10	1938/1; 7070 + 7071.
Various other single channel devices (Uniplexer)	10	
Multiplexer	11	7007/2 (1903 only).
Local Control V.D.U.	14	7154.

EX2V

Processors: 1902, 1903.

Description of Executive:

Single program, overlaid on E.D.S.. Automatic Operator feature is always included.

Optional Features:

Floating Point feature.

Configuration Restrictions:

Minimum store requirement of 8K words. Must have at least 2 E.D.S. drives (or one drive provided that Industry Compatible Magnetic Tape, NOT cassette, is also attached).

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	1911; 1912; 2102.
Card Punches	4	1920/2; 1922/1; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601; 2602/2.
Line Printers	2	1931/1, /2; 1932/1, /2; 1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
E.D.S.	6	2801/0, /2; 2802/0, /2.
Magnetic Tape (7-track)	5	1971; 1972; 1973.
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
U.D.T./O.C.R. + O.M.R.	23	8101 + 8201 + 8301.
Interrogating Typewriter	10	1938/1; 7070 + 7071 (1903 only).
Various other single channel devices (Uniplexer)	10	

E3TS

Processors: 1902A, 1903A

Description of Executive:

Single program Executive for Magnetic Tape systems, with Automatic Operator facilities included.

Optional Features:

Floating Point facility.

Configuration Restrictions:

Minimum store requirement of 8K words, plus a console typewriter.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	2101; 2104.
Card Punches	4	1920/2; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
Magnetic Tape (7-track)	5	1971; 1972; 1973.
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
Various other single channel devices (Uniplexer)	10	
Local Control V.D.U.	14	7154.

E3TE

Processors: 1902A, 1903A.

Description of Executive:

Single program Executive for Magnetic Tape systems, similar to the E3TS except that it includes the Trusted Program package. Also includes the Automatic Operator facility.

Optional Features:

Floating Point facility available.

Configuration Restrictions:

Minimum of 16K words of store must be available.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	2101; 2102; 2104.
Card Punches	4	1920/2; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader / Punch	0,1	2601/1; 2602/2.
Line Printers	2	1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
Magnetic Tape (7-track)	5	1971; 1972; 1973.
Magnetic Tape (9-track)	5	2504/1, /2, /3; 2505/1, /2, /3.
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
Various other single channel devices (Uniplexer)	10	
Local Control V.D.U.	14	7154

E3TM

Processors: 1902A, 1903A.

Description of Executive:

Multi-program Executive for Magnetic Tape systems, includes the Trusted Program package. (4 programs with up to 3 members each.)

Optional Features:

Floating Point facilities.

Configuration Restrictions:

Minimum core requirement of 16K words.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	2101; 2102; 2104.
Card Punches	4	1920/2; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotter	20	1934/1 to 6.
Magnetic Tape (7-track)	5	1971; 1972; 1973.
Magnetic Tape (9-track)	5	2504; 2505; 2506; 2507; (/1, /2, /3 all).
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
Interrogating Typewriter	10	7070 + 7071.
Various other single channel devices (Uniplexer)	10	
Local Control V.D.U.	14	7154.

E3DS

Processors: 1902A, 1903A.

Description of Executive:

Single program Executive for use with E.D.S. - based systems.

Optional Features:

Floating Point facility.

Configuration Restrictions:

Minimum of 8K words of core must be available. Two E.D.S. drives.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	2101; 2102; 2104.
Card Punches	4	1920/2; 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1933/1, /2, /3; 2401/1, /2; 2402.
Graph Plotters	20	1934/1 to 6.
E.D.S.	6	2801/0, /2; 2802/0, /2.
Magnetic Tape (7-track)	5	1971; 1972; 1973.
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
Various other single channel devices (Uniplexer)	10	
Local Control V.D.U.	14	7154.

E3DE

Processors: 1902A, 1903A.

Description of Executive:

Single program Executive for use with E.D.S. -based systems, with the Trusted Program feature incorporated; similar to E3TS.

Optional Features:

Floating Point facility.

Configuration Restrictions:

Minimum store requirement of 16K words, two E.D.S. drives.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	2101; 2102; 2104.
Card Punches	4	1920/2; 2151.
Paper Tape Readers	0	1915/2; 1916/2,/3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1933/1,/2,/3; 2401/1,/2; 2402.
Graph Plotters	20	1934/1 to 6.
E.D.S.	6	2801/0,/2, 2802/0,/2.
F.D.S. (Bryant 2A)	13	2803; 2804; 2805.
Magnetic Tape (7-track)	5	1971; 1972; 1973.
Magnetic Tape (9-track)	5	2504/1,/2,/3; 2505/1,/2,/3.
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
Interrogating Typewriter	10	7070 + 7071.
Various other single channel devices (Uniplexer)	10	
Local Control V.D.U.	14	7154.

E3DM

Processors: 1902A, 1903A.

Description of Executive:

Multi-program Executive for use with E.D.S. systems, includes the Trusted Program feature. (4 programs with up to 3 members each.)

Optional Features:

None.

Configuration Restrictions:

Minimum store requirement of 16K words, maximum of 32K words, must have at least two E.D.S. drives available.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	2101; 2102; 2104.
Card Punches	4	1920/2; 2151.
Paper Tape Readers	0	1915/2; 1916/2,/3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1933/1,/2,/3; 2401/1,/2; 2402.
Graph Plotters	20	1934/1 to 6.
E.D.S.	6	2801/0,/2; 2802/0,/2.
F.D.S. (Bryant 2A)	13	2803; 2804; 2805.
Magnetic Tape (7-track)	5	1972; 1971; 1973.
Magnetic Tape (9-track)	5	2504; 2505; 2506; 2507; (/1,/2,/3 all).
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
Interrogating Typewriter	10	7070 + 7071.
Various other single channel devices (Uniplexer)	10	
Local Control V.D.U.	14	7154

E3DG

Processors: 1902A, 1903A.

Description of Executive:

Disc-based overlaid Executive for use with GEORGE 3.

Optional Features:

None.

Configuration Restrictions:

Minimum of 32K words of store must be available, and all conditions demanded by GEORGE 3 Operating System, see specifications.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	2101; 2102; 2104.
Card Punches	4	1920/2; 2151.
Paper Tape Readers	0	1915/2, 1916/2,/3.
Paper Tape Punches	1	1925/2.
Paper Tape Reader/Punch	0,1	2601/1; 2602/2.
Line Printers	2	1933/1,/2,/3; 2401/1,/2; 2402.
Graph Plotters	20	1934/1 to 6.
E.D.S.	6	2801/0,/2; 2802/0,/2.
Magnetic Drum	9	1964; 2851/1,/2.
F.D.S. (Bryant 2A)	13	2803; 2804; 2805.
F.D.S. (Enhanced Capacity)	13	2803/2; 2806/2,/3,/4.
Magnetic Tape (7-track)	5	1971; 1972; 1973.
Magnetic Tape (9-track)	5	2504; 2505; 2506; 2507; (/1,/2,/3 all).
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
U.D.T./O.C.R. + O.M.R.	23	8101 + 8201 + 8301.
Interrogating Typewriter	10	7070 + 7071.
Various other single channel devices (Uniplexer)	10	
Local Control V.D.U.	14	7154.

E4BM

Processors: 1904, 1905, (1909).

Description of Executive:

Standard Executive for manual operating environments. Includes multi-programming and sub-programming (up to 4 programs of up to 3 members each may be in the processor at one time).

Optional Features:

Floating point extracodes on the 1904 only. Trusted program feature. Direct Response Mode available for certain peripherals.

Configuration Restrictions:

Minimum store requirement of 16K words, and typewriter control.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	1911, 1912, 2103.
Card Punches	4	1920/2; 1922/1, 2151.
Paper Tape Readers	0	1915/2; 1916/2, /3.
Paper Tape Punches	1	1925/2.
Line Printers	2	1931/1, /2; 1932/1, /2; 1933/1, /2, /3; 2401/1, /2.
Graph Plotter	20	1934/1 to 6.
M.C.F.	7	1958; 1959.
E.D.S.	6	2801/0, /2; 2802/0, /2.
Magnetic Drum	9	1962; 1963; 1964.
F.D.S. Bryant 2A	13	2803 2805.
Magnetic Tape (7-track)	5	1971; 1972; 1973.
1100 Mag. Tape to S.I.	59	1198; 1199.
Document Sorter/Reader	57	8500.

Cont'd overleaf

E4BM Cont'd.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
U.D.T./O.C.R. + O.M.R.	23	8101 + 8201 + 8301.
Interrogating Typewriters	10	1938/1; 7070 + 7071.
Various other single channel devices (Uniplexer)	10	
Multiplexer	11	7007/2.
Local Control V.D.U.	14	7154.

E4G3

Processors: 1904, 1905.

Description of Executive:

Executive for use with GEORGE 3 Operating Systems. It includes multi- and sub-programming facilities.

Optional Features:

None

Configuration Restrictions:

Minimum store availability of 16K words, other restrictions as determined by the requirements of GEORGE 3, see specification.

Permissible Peripheral Attachments:

See the Permitted Peripherals for the E4BM Executive (pages 21 and 22).

E6BM

Processors: 1904E, 1904F, 1905E, 1905F, 1906, 1907.

Description of Executive:

Typewriter-controlled Multi-program Executive. Trusted program feature. (16 programs with up to 4 members each.)

Optional Features:

Floating point extracodes on 1906 only.

Configuration Restrictions:

Minimum of 16K words of store must be available.

Permissible Peripheral Attachments:

PERIPHERAL	TYPE	CATALOGUE NUMBER
Card Readers	3	$191\underset{a}{1}$; $191\underset{a}{2}$; 2101; 2102; $210\underset{a}{3}$; $2104_{a,c}$
Card Punches	4	1920/2; $1922/1_{a}$; 2151.
Paper Tape Readers	0	1915/2; 1916/2,/3.
Paper Tape Punches	1	1925/2.
Line Printers	2	1933/1,/2,/3; $2401/1,/2_{a,c}$; 2402_{c}.
Graph Plotters	20	1934/1 to 6.
M.C.F.	7	1958_{a}; 1959_{a}.
E.D.S.	6	2801/0,/2; 2802/0,/2.
Magnetic Drums	9	$1962_{a,c}$; $1963_{a,c}$; 1964; $2851/1,/2_{b,c}$.
F.D.S. (Bryant 2A)	13	2803; 2804; 2805.
F.D.S. (Enhanced Capacity)	13	$2803/2 + 2806/2,/3,/4_{b,c}$.
Magnetic Tape (7-track)	5	1971; 1972; 1973; $1974/7,/8_{a,b}$.
Magnetic Tape (9-track)	5	2504; 2505; 2506_{c}; 2507_{c}; (/1,/2,/3 all).
U.D.T./O.M.R.	23	8101 + 8301.
U.D.T./O.C.R.	23	8101 + 8201.
U.D.T./O.C.R. + O.M.R.	23	8101 + 8201 + 8301.
Interrogating Typewriters	10	$1938/1_{a}$; 7070 + 7071.

Cont'd overleaf

E6BM cont'd.

PERIPHERAL	TYPE	CATALOGUE NUMBER
Various other single channel devices (Uniplexer)	10	
Multiplexer	11	$7007/2_{a,b}$
Local Control V.D.U.	14	7154

Note:

a indicates that this peripheral design is not available with the 1904A central processor with this standard Executive.

b indicates that this peripheral design is not available with the 1904/05 E + F central processors with this standard Executive.

c indicates that this peripheral design is not available with the 1906 & 1907 central processors with this standard Executive.

Dual Processor Executive.

Description of Executive:

Multi-program for use with large dual-processors, with trusted program and floating point facilities. This Executive is overlaid. The name has not yet been announced.

Optional Features:

None.

Configuration Restrictions:

Must have at least 32K words of core store.

Permissible Peripheral Attachments:

Similar to those for the E6BM Executive (see pages 24 and 25).

Paged Executive.

Description of Executive:

Multi-programming and sub-programming Executive for use with the GEORGE 4 Operating System. Paging facility.

Optional Features:

None.

Configuration Restrictions:

Must have at least 32K words of core store, and any other GEORGE 4 conditions.

Permissible Peripheral Attachments:

Similar to those for the E6BM Executive (see pages 24 and 25).

E6G3

Processors: 1904E, 1904F, 1905E, 1905F, 1906, 1907.

Description of Executive:

Multi-programming and sub-programming Executive for use with the GEORGE 3 Operating System.

Optional Features:

None.

Configuration Restrictions:

Must have at least 32K words of core store, and any other GEORGE 3 conditions.

Permissible Peripheral Attachments:

Similar to those for the E6BM Executive (see pages 24 and 25).

Section 0 SUB-PROGRAMMING

0.1	Introduction	1
0.2	Availability	1
0.3	Terminology	1
0.4	General	~~2~~ 3
0.5	Storage	3
0.6	Reference to common storage areas	~~3~~ 4
0.7	Program Dump and Reload	4
0.8	Initial loading states	5
0.9	The original system	5
0.10	The new system	6

[10] [10]

.1 Introduction

The original sub-programming facilities provided on 1904, etc., have proved inadequate, particularly where 'real-time' peripheral devices are involved. As a result, a new, more versatile, system has been provided, in such a form that any program which has been written on the basis of the original system will continue to work on 1904 etc., correctly, provided that the programmer observes the 'range-compatible' rules. It is essential that any one program use only one of the two systems (old or new), as any attempt to mix both systems is liable to produce unpredictable results; furthermore, the original system is not provided in all those Executives which include the improved system.

.2 Availability

Processor(s)	Executive	Old System	New System
1901	EX1H	No	No
1901	EX1T	No	No
1902/1903	EX2S	No	No
1902/1903	EX2M	No	Yes
1904/1905/1909	ALL	Yes	Yes
1906/1907	ALL	Yes	Yes

.3 Terminology

.3.1 'member'

Any unit of a program (previously 'master-program' or 'sub-program', but since the new facilities give the same power to each unit, the term 'member' is to be used).

.3.2 'current member'

The member which issued the instruction under discussion.

(.3, continued)

.3.3 'exists'

A member is deemed to exist when the appropriate priority word of the program's request slip is non-zero, and of the correct form (see P.D.R.M., Part 5, Section 1, Page 5)

.3.4 'sub-program control instruction'

Any of the instructions 162 (SUSMA), 163 (AUTO) and 164 (SUSAR/SUSIN)

.3.5 'suspension'

Within Part 4 of this Manual, except where otherwise indicated, this term refers to suspensions caused by sub-program control instructions only.

.3.6 'active'

A member is deemed to be active unless it is suspended for reasons of sub-program control. (A member is still active if it is suspended for some other reason, such as waiting for completion of a peripheral transfer).

.3.7 'flag-setting peripherals'

A class of peripherals which are defined as being capable of 'waking-up' an object program. A 'flag-setting interrupt' is an interrupt from such a peripheral which indicates that attention by the object program is required.

.3.8 'illegal'

A condition which will cause Executive to suspend the current member and to type an 'ILLEGAL' message. & halt??

.3.9 'restrictions'

Rules which must be observed if the program is to work correctly under all conditions, but are such that Executive is not normally able to trap violations. The effect of violating such rules is undefined and may vary depending on external conditions.

.3.10 'indeterminate/undefined'

Effects which cannot be predicted, and may vary, between different Executive/Processor combinations or, even when used in the same apparent situation from time to time (N.B. illegality may sometimes result).

.4 General

The various members of a program are permitted, by the multi-programming feature of Executive, to time-share with each other. Differing priorities will normally be assigned to each member, and each member of a program may become suspended for the usual reasons (awaiting peripheral termination or operator message) independently of the others. When a program is first loaded member 0 (previously 'master program') will be active (actually it will usually be suspended awaiting operator action), and all other members will be suspended awaiting activation by a '163, N(M)≠0' instruction, regardless of which system is to be used.

0.5 Storage

All members of a program share a common storage are defined by a common value of datum, and, normally, by a common value of limit also. However, in effect the Floating Point Accumulator and words 0-15 of each member (viz the eight accumulators, and eight additional words, two of which serve as a 'dump' for the contents of the F.P.A. during such times as the member is inactive because some other member, or a member of another program, is active) are distinct; each member's own set of these words will be accessed when references are made to them during execution of instructions from that member. When a program consists of more than one member, the storage area from Word 32 to word 31 + 16N of that program will be reserved for the temporary storage of words 0-15 of suspended members, (N is the number of members), and no member may use these words for any purpose (see also ~~1900 series Programming Notice No. 34, item 53~~). Part 0, Section 6.)

The actual allocation of this area is as follows:-

Words 32-47 : temporary storage for words 0-15 of member 0
48-63 : " " " " " " " 1
64-79 : " " " " " " " 2
80-95 : " " " " " " " 3

Note: The foregoing reference to N members assumes that the members are consecutively numbered. Should this not be the case, the reserved area must be calculated on the basis of (Q+1) members, where Q is the highest number used to refer to a member. (If Q=0 no temporary storage area is used).

0.6 Reference to common storage areas

There is no storage protection between data areas used by different members of a program. When data is placed by one member in a storage area for use by another member, the first member must ensure that it does not overwirte the data until the second member has finished with it. The sub-program control orders can be used for this purpose. Alternatively, indicators may be defined in storage for control communication between members. However, when data areas are used in this way, i.e. without the protection of the sub-program control orders, no assumption should be made as to the relative priorities of members, since these are not under a member's control. Accordingly, it is always possible that another member's orders may be obeyed between the obeying of any pair of successive orders in a member's routine, thereby causing alterations in common data items.

Some of the consequences of these factors are given here for guidance:-

(a) If under certain conditions an indicator is set by a member, and is nowhere reset by that member, then if another member determines that the indicator is set, it can assume that the setting conditions existed in the first member: however, if it detects that the indicator is not set, it cannot assume that the setting conditions did not exist.

(b) The alteration of an indicator, counter, etc., that is also altered by another member, and whose altered value is in some way dependent upon its original value, must be carried out by a single instruction that alters the quantity directly in its commonly accessed location, otherwise loss of information may occur. (Instruction 162, in the new system, provides a convenient means of overcoming this problem). See subsections 0.9 20.10 0.9 .10 [AL14]

(c) In general, a subroutine may not be common to two or more members of a program.

0.7 Program Dump and Reload

Orders 154 (CONT) and 155 (SUSDP) are illegal if issued by any member other than member 0. All members of a program

(0.7, continued)

are suspended whilst either of these orders are being carried out, hence, when a multi-member program is dumped, words 0-15 (including the F.P.A.) of all members will be in their respective temporary storage areas and thus will be dumped correctly for reload.

In the case of an operator initiated dump, it is the operator's responsibility to ensure that all members are suspended, prior to initiating the dump. This may be ensured by the use of a SUspend directive in respect of each member.

0.8 Initial loading states

When a program is first loaded member 0 (previously 'master-program') will be active (actually it will usually be suspended awaiting operator action), and all other members will be suspended awaiting activation by a '163, N(M) ≠ 0' instruction.

0.9 The original system

0.9.1 General

A program written in accordance with the original system will continue to work correctly on those Processor/Executive combinations which provide both systems, provided that the original rules have been strictly observed. However, as a result of the increased flexibility now provided, some mis-uses of the sub-program control instructions (e.g. instruction 162 (SUSMA) issued by other than member 0) which were, in the past, monitored by Executive as illegal may now be obeyed, producing an indeterminate effect.

0.9.2 Valid instructions, when using this systems:

.9.2.1 SUSMA : 162 X 0

"Suspend me if member X is active".

Restrictions: (a) the address N(M) must be zero.
(b) may only be issued by member 0 (master).

.9.2.2 AUTO : 163 X N(M)

"Activate member X at instruction in word N(M)"

Restrictions: (a) the address N(M) must not be zero.
(b) may only be issued by member 0 (master).

Illegal if : member X already active

(0.9, continued)

.9.2.3 SUSAR : 164 0 0

"Suspend me, reactivating member 0 if it was suspended due to my activity"

Restrictions: (a) the address X must be zero
(b) the address N(M) must be zero
(c) may not be issued by member 0 (master)

Notes: (a) all the foregoing instructions are illegal if they refer to a member that does not exist.
(b) on initial loading all members except member 0 are suspended as if they had just issued a SUSAR (164) instruction.

.10 The new system

.10.1 General

This system is much more versatile than the original one, having been designed both to overcome its weaknesses and to permit easy, yet efficient, means of handling peripheral devices that require prompt servicing by object programs. The following points are specially important:

(a) There is a special instruction for the alteration of the contents of common store areas (162, SUSMA) which helps the program to take care of some of the problems outlined in section 0.6.

(b) This system is symmetrical, i.e. any member can issue a sub-program control instruction in respect of any other member (in the original system all control had to be routed through member 0). This results in member 0 only being the 'master' when the program is first loaded.

(c) It is possible, as a result of misuse of the control instructions, for all members of a program to become suspended in such a way that none of them can be reactivated.

(.10, continued)

(d) When a member suspends itself, it does so awaiting reactivation by any other member, or by a flag-setting interrupt (see 10.2.3), and when reactivated it must ascertain the reason for this.

(e) Each member possesses two single-bit indicators M and P. M is the indicator set by "163, N(M) = 0" instructions given whilst the member is active; it remains set until cleared by a 164 instruction (X=1 or 2). P is the indicator set by flag-setting interrupts arising whilst the member is not in state 'SMP'; it remains set until cleared by a "164, X=2" instruction.

Both these indicators function in an accumulative fashion, being cleared only by an appropriate 164 instruction (which cancel the memory of all events of the type(s) concerned which have occurred since the last equivalent 164).

.10.2 Valid instructions. when using this system

10.2.1 SUSMA : 162 X N(M)

"Conditional alteration of one word-pair in common storage"

If the contents of word N(M)+1 are zero : make them non-zero, copy (X) to word N(M), and omit the instruction which immediately follows this instruction.

If the contents of word N(M)+1 are non-zero: do nothing and proceed to the instruction which immediately follows this instruction.

Restrictions: the address N(M) must not point to any word which lies within the restricted area at the lower end of object area. (The size of this area is dependent on the number of members in the program and may be determined from the details in Part 0 Section 6.Note that this restriction applies to all restricted locations regardless of general alleviations which are given in Part 0 Section 6.

(.10, continued)

Illegal if : N(M)+1 outside reservations.

10.2.2 AUTO : 163 X N(M)

"Activate member X"

This instruction takes two forms, one for the first activation of a member after the program is loaded, the other for subsequent reactivation of a member which has suspended itself by a 164 (SUSAR or SUSIN) instruction.

.10.2.2.1 For first activation, address N(M)≠ 0

Member X is activated, the instruction in word N(M) being the first instruction to be obeyed by that member.

Restrictions: (a) Because N(M) is the address of an instruction which is actually obeyed it must not be an address within the restricted area at the low address end of object storage.

(b) This form may only be obeyed once for each member (i.e. only when a member is being activated for the very first time after program loading).

(c) Must never refer to member 0 (implicit in restriction 'b')

Illegal if : (a) X is the current member.

(b) Member X does not exist.

(c) Member X is not suspended.

.10.2.2.2 For subsequent reactivations, address N(M) = 0.

Member X is, if currently suspended due to a 164 (SUSAR or SUSIN) instruction, reactivated at the instruction following that 164. If member X is active at the time this 163 instruction is issued, its indicator M will be set and will remain set until member X issues a 164, which will clear it but not suspend member X.

(.10. continued)

Restrictions: (a) Address N(M) must be zero.
(b) This form must not refer to a member which has never been activited since the program was loaded.

Illegal if: (a) member X does not exist.
(b) member X is the current member.

.10.2.3 SUSAR/SUSIN: 164 X 0

"Conditional suspension of current member"

.10.2.3.1 address X = 1 (SUSAR)

If the current member's indicator M is set, clear it and proceed with the next instruction.

If its indicator M is not set, suspend the current member until such time as a "163, N(M)=0" instruction, referring to it, occurs.

.10.2.3.2 address X = 2 (SUSIN)

If either of or both the indicators M and P are set, clear them and proceed to the next instruction.
If neither is set, suspend the current member until either a "163, N(M) = 0" instruction referring to it, or a flag-setting interrupt (from any peripheral assigned to this program) occurs.

Restrictions: (a) Address N(M) must be zero.
(b) Address X may only take one of the specified values.

(.10, continued)

.10.3 Flag-setting interrupts.

When a peripheral device interrupts because prompt action is required from the object program, every member that is suspended due to a "164, X=2" instruction will have that suspension removed and will continue at the instruction immediately following the 164 instruction. For every member that is not so suspended the interrupt will set its indicator P which remains set until that member issues a "164, X=2" instruction, which will clear indicator P but not suspend the member concerned.

.10.4 State transition diagrams

The two diagrams that follow summarize the effect of various events under all possible valid conditions. The following explanatory notes should be read in conjunction with these diagrams.

(a) The word 'undefined' indicates that the condition concerned is not valid, i.e. it is a voilation of the restrictions stated in section 10.2, and that the effect is unpredictable.

(b) The use of forms of the control instructions not defined in 10.2 (e.g. "162, N(M)=0" and "164, X=0") will have 'undefined' effect in this new system.

(c) At any time a member is in one of the following states:

NS : not suspended (active)

SL : suspended because it has not yet been activated at all since the program was loaded (member 0 cannot be in this state).

SM : suspended because it has issued a 164, X=1 instruction.

SMP: suspended because it has issued a 164, X=2 instruction.

(d) When a program is loaded, member 0 assumes state 'NS', and all other members assume state 'SL', and both indicators M and P are clear.

(e) The 'GO AT' directive, which may only refer to member 0, may only be used at a time when all other members are in a state of suspension.

(.10, continued)

(.10.3 continued)

Diagram 1 : The effect of the AUTO (163) instruction and flag-setting interrupts on suspension states and the indicators M and P

Event, or instruction issued by member 'p'	Applying to member	Member 'q' being in state;			
		NS	SL	SM	SMP
163, N(M) = 0	q	Mq=1	Undefined	q →NS	q →NS
163, N(M)≠0	q	Illegal	q →NS	Undefined	Undefined
flag-setting interrupt	q = all members	Pq=1	Pq=1	Pq=1	q →NS

Diagram 2 : The effect of the SUSAR and SUSIN (164) instructions on the suspension state and indicators M and P of the issuing member.

Member 'p' issued a 164 with	indicators set:			
	Neither	P only	M only	M and P
X = 1 (SUSAR)	p→SM	p→SM	Mp = 0	Mp = 0
X = 2 (SUSIN)	p →SMP	Pp = 0	Mp = 0	Mp = 0 Pp = 0

PART 5 CONTENTS Page No.

Section 0 INTRODUCTION

Section 1 BINARY PROGRAM FORMATS

1.1 Binary Program Formats (General) 1

1.2 Block Types 1

1.3 Layout of Block Types 2

Section 2 LAYOUT OF PROGRAM DATA ON VARIOUS MEDIA

2.1 Paper Tape 1

2.2 80-Column Cards 1

2.3 Magnetic Tape 2

Section 3 DUMPING OF PROGRAMS

3.1 Dumping 1

3.2 General 1

3.3 Dumping of zeroes 1

3.4 Devices which may be used for dumping 1

3.5 Program Initiated Dumps 2

3.6 Operator Initiated Dumps 2

Section 4 LOADING PROGRAMS

4.1 Devices which may be used for program loading 1

4.2 Program Initiated Loading 1

4.3 Operator Initiated Loading 2

4.4 Action of the FInd message with Magnetic Tape and Cassette Tape 4

4.5 Action of the FInd message with E.D.S. or F.D.S. 5

4.6 Executive treatment of Request and Supplementary Request Blocks 7

Introduction

Although programs are normally written in a convenient language e.g. PLAN, FORTRAN, COBOL, the central processor, and Executive can only handle programs in machine code. This form is known as Binary Program. It is a function of the compilers and associated routines such as the General Purpose Loader to convert, in suitable steps, the source language program into Binary Format.

Some compilation processes convert the source program into Binary Form and output in it on a suitable medium for preservation and loading. Others leave the Binary Program in the processor store, ready for use. In the latter case it is possible to make Executive output the Binary Program onto a suitable medium, such a process being known as Dumping.

This part of the manual describes the formats used on various media for Binary Programs and the means by which loading and dumping can be carried out.

.1 Binary Program Formats

All programs produced or input directly be Executive have the following standard format:-

(a) A request block heads the program indicating to Executive the program's size, peripheral requirements and program/sub program priorities. (Normal 1902/3 Executives disregard the priorities).

(b) The number of binary program data blocks following. (Type 5 block pairs are counted as one block).

(c) The program is terminated by an entry block which indicates to Executive the required action on the program.

.2 Block Types

All blocks consist of an integral number of 24-bit words, the first of which invariably indicates the block type and size. The block type numbers are as follows:-

Request block	-	Type 1
Data blocks	-	Type 0 or 5
Entry blocks		Type 2,3 or 4
Supplementary request block	-	Type 6
Pre-request block sentinel	-	Type 62 (#76) for Cassette Tape only. (See Part 2 Sect. 4.7.4)

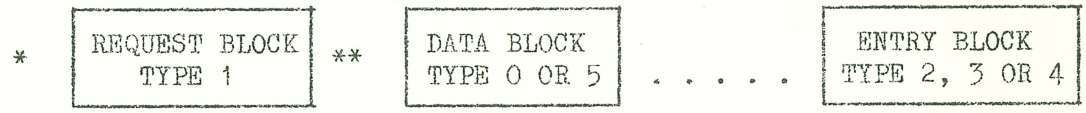
The first word of all block types has the following information content:

Char 0	-	Always octal 73
Char 1	-	Number of words in block (for sum check)
Char 2	-	Unused (normally zero)
Char 3	-	Block type.

The foregoing paragraphs may be summarized in diagrammatic form, thus, (see next page).

(.2 cont'd)

Format common to all programs.



* In the case of Cassette Tape, a pre-request block sentinel will be found here.

** In some environments, a supplementary request block will be found here.

First word of each block (block specifying word).

Octal 73	No. of words in block	Unused	Block Type
Ch.0	Ch. 1	Ch. 2	Ch. 3

Note: In overlay programs, there is one extra entry block per overlay plus the normal one for the initial entry following the permanent program.

.3 Layout of Block Types.

(a) (i) Type 1 - Request Block.

Word 0 Normal specifying word as shows above. Character 1 (number of words in record) will be 14 or 16.

Word 1 Four character program name which is used in Operator/Executive communication. Optionally a further 8 program name characters may be placed in words 14 and 15 of the block.
The program name must be composed of ~~alphanumeric characters~~ letters & numbers, the first character being a letter.
The program name extension must be composed of ~~alphanumeric characters~~ letters & numbers and left justified.

AL/17

AL/17

(.3 cont'd.)

~~The characters must be chosen from the range of characters which can be output on the console typewriter; i.e. the 64 character internal code less ∅,], ↑, ←.~~ AL17

Word 2 Peripheral request and trusted program status word. It is laid out thus:-

Bits 0, 1	indicate "retain load peripheral"
Bits 2 to 5	indicate trusted program status
Bits 6 to 20	peripheral device requests
Bits 21 to 23	undefined.

Peripheral Device requests. Up to 7 units of any peripheral of types 0 to 4 may be requested. The assignments are made to the program's unit numbers starting at 0 for each type.

Bits 6 to 8	requirement for type 0 (paper tape readers)
Bits 9 to 11	" " " 1 (paper tape punches)
Bits 12 to 14	" " " 2 (line printers)
Bits 15 to 17	" " " 3 (card readers)
Bits 18 to 20	" " " 4 (card punches)

"Retain load peripheral". If either Bit 0 or Bit 1 is set to 1, the peripheral from which the program is loaded will be assigned to the program as unit 0 of its type. (If one or more peripherals of the same type are requested in bits 6 to 8 or bits 15 to 17, the setting of Bit 0 or Bit 1 will not result in an additional unit being assigned; it will, however, ensure that the load peripheral is the one retained as unit 0).

The distinction between Bit 0 and Bit 1 is that when the program is dumped Bit 0 is always zero, but Bit 1 is in the same state as on loading.

Bit 0 is used with programs in GPL form to ensure that the loader reads the semi-compiled program from unit 0. It is set automatically by compilers and consolidators.

(.3 cont'd.)

Word 2 (cont'd)

Trusted program status.

Bit	Status	Single slot trusted program	1904/5 scheme
2	Q	Writes to direct access system/ directory files	
3	R	SSTP facilities used	167 order used
4	S	Ditto; status lost on dumping	160 x = 6, 7 used
5	T	Requires use of George 3 interface	

Word 3 Core store request word. This word contains the initial core store requirement for the program. It is expressed in two parts:-

(a) the number of units of 32,768 words, as a binary number in bits 17 to 23.

(b) the additional number of units of 64 words, as a binary number in bits 0 to 8.

Bits 9 to 16 must be zero.

Word 4 Undefined.

Word 5 Overlay directory word. It is laid out thus:-

	Value	Meaning
Bits 0 and 1		Undefined
Bits 2 to 23	$n \neq 0$	Overlay directory to start at word n.

If word 5 is zero the bootstrap will take no action. If it is non-zero, the program's subfile description is scanned, and a directory giving the layout of its overlay and permanent units is built up starting at word n.

(.3 cont'd.)

Word 6 Undefined.

Word 7 Priority of member 0 (see below).

Words 8, 9, 10,11,12 Priority of member 3, 1, 4, 2, 5 respectively (see below).

Priorities of members. The words in the request block which define the priorities of the various members of the program are each laid out as four characters thus:-

Characters 0,1 - member's priority (normally two decimal digits)

Character 2 - octal 77

Character 3. - member number (i.e. 0 for member 0 etc.)

An entirely zero priority word indicates that a members does not exist.

Word 13 Negative check sum.

Words 14,15 Program name extension. Optional. If present, it is included in the length character of word 0 and the check sum, and is output on dumping.

(.3 cont'd.)

The layout of this block is summarised in the following table:-

WORD	CONTENTS								FUNCTION	
0	#73		Block count		Not used		Block type		Block-specifying word	
1	Ach		ANch		ANch		ANch		Four character program name	
2	*	**	TR	TP	LP	CR	CP		Peripheral requests etc.	
3	Units of 64 words		Zero		Units of 32K words				Core Store request	
4									Undefined	
5	U	Address							Overlay directory start address	
6									Undefined	
7	Nch		Nch		#77		0		Priority of member 0	
8	Nch		Nch		#77		3		Priority of member 3	
9	Nch		Nch		#77		1		Priority of member 1	Zero if
10	Nch		Nch		#77		4		Priority of member 4	member
11	Nch		Nch		#77		2		Priority of member 2	conerned not
12	Nch		Nch		#77		5		Priority of member 5	present
13									Negative check sum	
14	ANch		ANch		ANch		ANch		Optional program	
15	ANch		ANch		ANch		ANch		name characters	

Note: ch = character (A = alpha; N = numeric)

* : Bits 0,1 indicate "retain load peripheral"

** : Bits 2 to 5 indicate trusted program status

U : Undefined.

(.3 cont'd)

(a) (ii) Type 6 - Supplementary Request Block.

Word 0 Normal specifying word.

Char 0 - #73.

Char 1 - Number of words in record (8)

Char 2 - Unused

Char 3 - Type (ie. 6)

Word 1 Mode word. The mode word contains two bits which convey the setting of 22 bit Address Mode and Extended Branch Mode.

Bit 21 = 0 indicates DBM (direct branch mode)

= 1 " EBM (extended branch mode)

Bit 23 = 0 " 15AM (15 bit address mode)

= 1 " 22AM (22 bit address mode)

The remainder of the word is undefined.

Words 2-6 Undefined.

Word 7 Negative check sum.

The supplementary request slip, if present, would normally follow immediately after the standard request slip.

On Executives without the EBM and 22AM facility, the presence of a supplementary request slip will cause the program to be rejected, even if words 0 to 6 are set to zero. AL17

(b) Data block type 0

This is a block of x words such that $4 \leq x \leq 20$. It is principally used with media other than magnetic tape, since, in the latter, considerable economies result from the use of type 5 blocks. Block layout is as follows.

Word 0	Normal block specifying word
Word 1	Destination address, relative to datum, for the first word of program in the block (word 2).
Word 2	First program word in a binary format suitable to the medium in use.

Up to 15 further program words may follow.

Word following the last program word	Negative sum check
Next word	Optional block sequence number (see 1.3(g))

Notes: (i) The 'number of words' in Word 0 is always the number of program words plus 3 (it never includes the optional block sequence word, which, is always ignored by Executive).

(ii) If all the program words in a type 0 block contain zero the block is usually omitted.

(c) Data block type 5

This is a data block pair and consists of,

(i) a 5-word specifying block which gives the parameters of the following data block, and

(ii) the data block itself which has a maximum word length of 512

The layout of the specifying block is as follows:

Word 0 Normal specifying word in which both the word count character and the block type character are set to 5

Word 1 Destination address, relative to the datum point, of the first word of the data block.

Word 2 Word-length of data block (maximum 512)

Word 3 Negative sum check of data blocks.

Word 4 Negative sum check of specifying block.

N.B. For the purpose of block counts, each type 5 block pair is counted as one block; this only applies to block counts in entry blocks and not to block counts in trailer labels and end of sub-file sentinels.

(d) Entry Block type 2

This is a 3 or 4 - word block which instructs Executive to GO at the specified starting address of the main program. Block layout is as follows.

Word 0 Normal specifying word giving the block word count 3 (or 4) and the block type 2.

Word 1 Indicates the entry point of the program.

Word 2 Negative sum check or block ~~word~~ count (but see 1.3(g).)

Word 3 Negative sum check if block ~~word~~ count in Word 2.

(e) Entry block type 3

A 3 or 4 - word block which instructs Executive to SUSPEND the main program, after loading, awaiting operator action. Block layout is as follows:-

Word 0 Normal specifying word, block word count 3 (or 4), block type 3.

Word 1	Entry point of program.
Word 2	Negative sum check or block ~~word~~ count
Word 3	Negative sum check if block ~~word~~ count in Word 2, (but see 1.3(g)).

AL15

(f) Entry block type 4

A 3 or 4 word block normally used to terminate an overlay (i.e. additional program), which is intended to be read in by a 154(CONT) instruction. When Executive encounters a Type 4 block, it returns control to the instruction in the word following that from which the 154 (CONT) instruction was obtained. The additional program read must not include a request block .

Block layout is as follows:

Word 0	Normal specifying word, block word count 3 (or 4), block type 4.
Word 1	Zero
Word 2	Negative sum check or block ~~word~~ count
Word 3	Negative sum check if block ~~word~~ count in Word 2. (but see 1.3(g)).

AL15

(g) Block Counts and Sequence Numbers

When Executive dumps a program it produces a 4-word entry block, putting in word 2 the total number of blocks dumped, using word 3, instead of word 2, for the negative check sum. It also, on some media (e.g. punched cards), extends type 0 blocks to 20 words, using word 19 to provide a block sequence number, but it does not alter the block specifying word, which always quotes 19 words.

On loading, Executive will, if the entry block is 4 words long, check that the total number of blocks read is correct. Type 0 blocks are always treated as being 19 words long,

the sequence number being intended for external purposes only (e.g. for identifying a card).

Type 5 block pairs count as one block.

N.B. When the program read in does not have a request block (i.e. program read as a result of a 154 instruction or a type 4 entry block) the block count in the entry block must be one greater than the actual number of blocks. (This is to allow the request block to be removed from the result of a dump, so that it can be read back in the continuation mode).

.1 Paper Tape

Binary blocks are punched in 8-track tape in graphic set mode with even parity, each block being terminated with a NEWLINE character and followed by a gap of, at least, three blank tape characters. The program tape will have the previously detailed format i.e. request block, data blocks and entry block.

.2 80-Column Cards

Blocks are stored one per card, in decode mode, each column representing a 6-bit internal machine code. The maximum number of columns required to hold a block will thus be 76 (in the case of a 19-word data block type 0). The last four columns of each card are therefore available and are used to contain a card count which allows card sequence checks to be made. The card count is not included as a word to be sum-checked and thus does not affect the previously detailed block sum check procedure. It should be noted that this sequence number is not punched in decimal form, but consists of the card code characters produced by the binary value of the sequence number.

In programs produced on card by Executive, the entry block layout is modified by an additional word which indicates the number of program blocks thus,

Word 0 Normal specifying word

Word 1 Entry Point

Word 2 Number of program blocks

Word 3 Negative sum check

.3 Magnetic Tape

Binary blocks of 5-word minimum length are written on magnetic tape in odd parity mode. Nothing can be assumed about data (which will be redundant to Executive) beyond normal binary block length. The format of a program tape which is requested by means of the FInd directive is as follows:

(a) A header label with the file name (Words 2 to 4) 'PROGRAM name' (where 'name' is the four characters stored in Word 1 of the request block) and with reel sequence number (Word 5) zero.

(b) Tape mark

(c) Request block type 1.

(d) A number of program blocks (preferably type 5).

(e) Entry block.

Optionally, other blocks may be written between the label and the ~~end of file~~ tape mark. AL/15

When a program is loaded from magnetic tape either by the LO directive or the 154 order, the tape must be positioned immediately before the first block to be read, i.e. the request block if LO is used, the first data block if the 154 is used. Following a LO directive, the tape will be rewound and closed, unless B0 or B1 in word 2 of the request block called for the load peripheral to be retained.

The DUmp directive finds a scratch tape on an unallocated deck and constructs a tape with the format shown in items (a) to (e) above. The tape is then rewound and disengaged (i.e. removed from Executives list until re-engaged). ~~If the DUmp directive specifies a particular deck, only the request block, data blocks and an entry/block (continued on next page).~~ [AL/13]

~~block type 3 are constructed on the tape~~. On most processors, [AL/13]

the data blocks dumped onto Magnetic Tape are of type 5, but

1902~~/3~~ 1/2/3 Executives always dump type 0 blocks. AL/15

A dumping action by means of the 155 order produces a similar

tape format, no ~~end of file~~ tape marks being written. [AL/12]

Dumping

.1 Dumping

The dumping of a program can be initiated by the operator, or by the program itself (by means of a 155 order)

Warning

When Executive is instructed to dump a program it does so without checking the state of any peripheral devices other than the one to be used for the dump. As a result, should the program concerned have any peripheral active at dump time, the results of the dump may be indeterminate.

.2 General

In all cases the request block output will be a copy of the original except that bit 0 of word 2 will always be zero (see Section 1.3 (a)). The core-store figure will always be the original value (which may differ from the actual current size of the program due to a 165/4 order, or because the operator specified the amount of store to be used at load time). The entry block produced will always be of type 3.

.3 Dumping of zeros

A dump which is in the form of type 0 data blocks (i.e. all except a magnetic tape dump on 1904 and above) will omit any blocks in which all 16 program words are zero, as, on loading, the store is initially zeroised.

.4 Devices which may be used for dumping

These are Paper Tape Punches, Card Punches, Magnetic Tape and, in the future, other types of File-Storage devices.

.5 Program Initiated Dumps

The 155 (SUSDP) introduction, which is only acceptable if initiated by the master program, causes the master and its sub-programs to be suspended, and then dumps the complete program area. Such a dump is written to the specified device without any prior repositioning of the output medium. On completion of the dumping, the program continues with the instruction following the 155 instruction, without operator intervention. (See the 155 instn.) AL/16

.6 Operator Initiated Dumps

(a) General

Usually, the program concerned, and its sub-programs, will already be suspended when the operator calls for a dump. Should this not be the case, they will be suspended immediately and, on completion of the dumping, operator action will be required to restart them.

(i) If a particular device is specified by the operator

The device specified may only be a paper tape or card punch, and it must either be a device currently assigned to the program to be dumped, or a device which is not assigned to any program.

(ii) If no device is specified by the operator

Executive locates, on any tape deck which is not currently assigned to any program, a tape which is available for use as an output tape (i.e. retention period exceeded etc.,). It labels this tape 'PROGRAM_name', giving it reel sequence and generation numbers of zero. In order to give reasonable protection to the dump a retention period of seven (7) days is given. Should Executive not be able to locate a suitable tape, the operator will be notified.

On completion of the dump, the tape is closed but the operator is not notified which tape was used; any retrieval procedure must either use the FInd directive or a program which opens the dump by name.

Loading

In general, the loading of a program is operator initiated, but under certain circumstances it may be program initiated.

.1 Devices which may be used for program loading

These are Paper Tape Readers, Card Readers, Magnetic Tape and, in the future, other types of File-Storage devices.

.2 Program Initiated Loading

(a) During the running of a program

The 154 (CONT) instruction, which is only acceptable if initiated by the master program, provides the programmer with a simple type of overlay facility. It causes the master program, and all its sub-programs, to be suspended immediately. Executive, then, reads Binary Program (which must not have a request block) into the program's area, the locations overwritten being determined by the data blocks read. The area read into is not zeroised before loading, hence care must be taken in respect of areas which were zero when the Binary Program was created. On completion of the loading, the action taken will depend on the type of entry block read. Usually this will be of type 4 ~~2~~, which will cause the program to continue with the instruction in the location following that which contained the 154 instruction (either or both of these locations may have been overwritten by the program just read in). ~~If the entry block is of type 3, the program will be suspended awaiting operator action. Should an entry block of type 4 be encountered still more program will be read in (again no request block is allowed).~~ [AL/12] [AL/12]

The input medium must be correctly positioned, immediately before the first data block, at the time that the 154 instruction is given.

Warning

When a 154 instruction is initiated Executive starts the loading process without checking the state of any peripheral devices, other than the one to be used in the loading process. As a result, should the program which issues the 154 instruction have any peripheral active at the time, the results of the loading may be indeterminate.

(b) When a program deletes itself

If a program deletes itself by means of a 160/2 (DELTY) instruction, Executive obeys the message output by the program as if it had been typed by the operator. (This enables one program to initiate the loading of another). The effect of such an instruction in no way differs from the effect of the operator typing the same message. (see below for further details).

.3 Operator Initiated Loading

There are two operator directives which can initiate program loading .

(a) LOad

This directive causes Executive to read Binary Program from the specified device, which is normally a Paper Tape Reader or Card Reader. Such a program must be in the standard form commencing with a request block. The program bearing medium must be correctly positioned in the reading device, so that the first block read is the request block.

The LOad directive will be used in respect of Magnetic Tape only as a result of a 160/2 order.

(b) FInd

The FInd directive is used to load a program from backing storage. In some cases Executive calls in a search program which locates and loads the actual program required.

In what follows any information about the actions of search programs describes conventions and not rules; for details see the specifications of search programs.

The backing storage medium primarily used is specified when an Executive is compiled, and the action of the FInd directive depends on the medium chosen.

The FInd directive on processors with console typewriters has two main forms, one specifying one, and one specifying two names. In either case a decimal integer, equal to 64 or more may follow, which specifies a core store reservation which is to be used instead of that specified in the request block. On 1904 processors and upwards decimal integers less than 64 may follow which represent the absolute numbers of peripherals which are to be allocated to the program to be loaded, each as unit 0 of its type (hence there must be not more than one of any type allocated by this method). Not more than three integers in total may be specified in a FInd directive. Direct access peripherals cannot be allocated in this way.

For the format of messages see Part 6 Sect. 1.

.4 Action of the FInd directive with Magnetic Tape (Industry Compatible) and Cassette Tape.

A. FInd ≠JACK

In response to FInd ≠JACK (where ≠JACK represents any legal program name), Executive searches all the currently unassigned tape decks or cassette stations for a tape with file name PROGRAMVJACK and reel sequence number zero (as if opening it in mode ≠100 as unit 0). If it finds one it loads the binary program from that tape, and further action is determined by the program loaded. In the case of cassette tape, Executive searches for a block with block sequence word negative (this should be a pre-request block) before loading the program.

If the name ≠JACK is that of a search program, the latter on entry will conventionally halt, and output the message HALTED SL. The operator should then type a GO AT directive (see Part 6 Sect. 1.4(b)) to indicate whether a steering line to specify the program actually required is provided on cards or paper tape.

If Executive does not find a file called PROGRAMVJACK it will instruct the operator to load one. Formerly, under these circumstances, after a 160/2 (DELTY) instruction, Executive would search for a magnetic tape called PROGRAMVTAPE and use ≠TAPE as a search program to locate and load PROGRAMVJACK. This can still be achieved, if, after the LDT PROGRAM JACK console message, the operator types FI ≠JACK ≠TAPE (followed by the integers of the original FInd directive).

On processors without console typewriters the FInd process (handswitch setting 20/0) reads from the cassette station currently allocated as the program's unit 0 until it finds a block with block sequence word negative (this should be a pre-request block), and then loads the program.

B. FInd #JACK #SSSS

In response to FInd #JACK #SSSS (where #JACK and #SSSS represent any legal program names), Executive will search all the currently unassigned decks or cassette stations for a tape with file name PROGRAMVSSSS. If no such tape is found the message LDT PROGRAM SSSS will result. If the tape is found Executive will load the first program on it (which must be called SSSS and is conventionally a search program), and will pass the name #JACK, any core requests and the additional information in the FInd directive to it, as follows:-

Word 10	Name of program required (in this example JACK)
Words 11 to 13	
Bits 2 to 23	Integers specified in the FInd directive
Bits 0 and 1	Indeterminate

If less than three integers were specified, bits 2 to 23 of the remainder of words 11 to 13 may be set to zero or left as loaded (conventionally zero in any case).

Conventionally, the search program will locate the program specified, position the tape before the request block and obey a 160/2 (DELTY) instruction which implements the directive LOad #JACK followed by the geographical address of the tape deck and any integers in the FInd directive.

5. Action of the FInd directive with E.D.S. or F.D.S.

In response to any FInd directive eg. FInd #JACK #DISC (where #JACK and #DISC represent any legal program names) Executive loads a special search program from the reserved area on the disc into words 10 to 137 of the store, assigns it the first name in the FInd directive, and hands over the information in the FInd directive as follows:-

Word 10	First name in directive, if there are two (in this example JACK)
Word 11	Second or only name in the directive (in this example DISC)
Words 12 to 14	Integers specified in the FInd directive.

If only one name or less than three integers were specified in the FInd directive, the remainder of words 11 to 14 will either be set to zero or left as loaded (conventionally zero in any case).

If the directive is of the form FInd /JACK, the search program will conventionally try to open a file with name PROGRAM/JACK as unit 0 and load the program from it, taking the usual action on the integers.

If the directive is of the form FInd //JACK //DISC, the search program will conventionally try to open a file with name PROGRAM/DISC as unit 0, and load the program from its subfile named PROGRAM/JACK, taking the usual action on the integers.

In both cases, when the program is loaded, the search program will change its request slip in order to convert itself into the program JACK. Words 10 to 19 and 31 to 34 may be corrupted in the loading process.

The action taken if either the file or the subfile is not found will depend on the search program, for example, some search programs will proceed to repeat the attempt on another medium e.g. magnetic tape (for details see the specification of the search program used).

.6 Executive Treatment of Request and Supplementary Request Blocks

.1 EX1H. With this Executive all store and peripherals are regarded as being permanently assigned to the object program. Request blocks are ignored on program loading (though if one is present its checksum must be correct).
A supplementary request block is signalled as an error.
The RRQ (166) order is null.
There is no storage of request block details, and a program dump has no request block. Such a dump on re-loading is acceptable to EX1H, but not to any other Executive unless a request block is added.

.2 EX2L. As with EX1H, there is no assignment of peripherals. The request block must sum check correctly, and the name must agree with the name on the console LO message. A check is then made that the core store requirement stated on the request block does not exceed the available storage. The request block is stored with bit 0 of word 2 zero.
A supplementary request block is signalled as an error.
The RRQ (166) instruction is null; the stored request block therefore remains unchanged during the running of the program.
A dump by Executive of the object program is preceded by a copy of the stored request block.

.3 EX1T, EX1V, EX2S, EX2V. These are single program Executives in which peripherals are assigned to the program's unit numbers on demand. All available storage (outside Executive) is regarded as being available to the object program.

EX2M, E4BM. These are multiprogramming Executives in which core store and peripherals are assigned and released on demand.

Program loading - these 6 Executives check sum all binary program records on reading. The following checks and actions are then carried out on the request block:-

(.6.3 cont'd)

a) Check that this is the first record of the program.

b) Check that the program name in the request block agrees with the name on the console LO message.

c) Check that the core store requirement (from the request block unless overridden by LO message) is available.

d) Assign load peripheral if bit 0 or 1 of word 2 is set.

e) Check that the peripherals called for in the request block are available; these units are then assigned to the program.

In addition, EX2M and E4BM establish the priorities of members; E4BM also monitoring the program name extension.
The request block is stored with bit 0 of word 2 zero.
The appearance of a supplementary request block during input of the program is signalled as an error.

Changes during execution - no alterations are made to the stored request block as a result of assigning or releasing peripherals, variations in core store used, etc. Changes arise as the result of 166/1 instructions, which completely replace the old request block by the new.

Program dumping - a dump by Executive of the object program is preceded by a copy of the latest stored version of the request slip.

.4 E6BM. This Executive handles the request block in the same way as E4BM. The supplementary request block (type 6) is recognised, and causes 22AM and/or EBM to be set as indicated in the mode word. If the program switches these modes during execution, the new setting is used in any program dumping operation. On dumping, a supplementary request block is produced immediately following the request block proper unless its contents would have been zero.

PART 6 CONTENTS Page No.

Section 0	Input from Operator Executive	
	0.1 Introduction	1
	0.2 Initiation of Messages	1
	0.3 Interpretation of Messages	2
	0.4 Identification of Messages	3
	0.5 Form of Messages	3
	0.6 Notation	5
Section 1	Typewriter Input Messages	
Section 2	Output Messages	
	2.1 General	1
	2.2.1902/3 Messages from Executive	1
	2.3 1902/3 Messages from the Program	2
	2.4 1902/3 Messages output during the running of Magnetic Tape Programs	3
	2.5 1904/5 Messages from Executive	5
	2.6 1904/5 Messages from the Program	7
	2.7 1904/5 Messages output during the running of Magnetic Tape Programs	8
TABLE A	1902/3 ERROR CODES	
TABLE B	1904/5 OPERATOR ERRORS	
TABLE C	1904/5 PROGRAM ERRORS	

Input from Operator to Executive

.1 Introduction

Instructions to Executive from the Operator are given by certain fixed messages which may be input from the console typewriter, or from the paper tape reader(s) or card reader(s).

On the 1902/3 only the principal Executive-loading peripheral may be used as an alternative to the console typewriter for the input of Executive messages.

.2 Initiation of Messages

A message is first initiated by means of pressing the INPUT button on the console typewriter. This causes an entry into Executive.

N.B. If any other specific input device is used for reading input messages,

(i) firstly the INPUT button is pressed followed, when the INPUT button light comes on, by the CONTROL key. While the latter key is still depressed the 'A' button is pressed. (The actual device assigned to 'A' will vary according to the installation). - 1902/3

or (ii) one of the F buttons is pressed. - 1904/5

If a) Executive can be entered and

b) the typewriter is not required for an output message,

the INPUT button will light up (informing the operator that the typewriter is in an input mode).

In the case of the 1904/5, the key board will unlock after the the INPUT button lights up, and the operator will be able to type the message. There is no locking device on the 1902/3 and the operator must wait until the INPUT button light goes on before proceeding with the message. If, however, the operator dees proceed before the INPUT light goes on, Executive types 'CANCEL'.

The message which is typed must observe prescribed format rules and must be one of the standard messages described in Section 1 When the message has been typed, the operator must

check this for accuracy. If an error is discovered the operator must press the:-

RUB-OUT key (situated on the key-board itself) - 1902/3
CANCEL button (situated on the panel to the right of the key-board) - 1904/5

This will cause the INPUT light to go out and a message 'CANCEL' to be printed via the typewriter.

The whole procedure, commencing with pressing the INPUT button (or the corresponding procedures for any other specific input devices described above) must now be repeated.

When satisifed as to the correctness of the message, the operator must press the:-

ACCEPT key (key-board) - 1902/3
ACCEPT button (panel) - 1904/5

The message will not be given to the computer to analyse unless this is done. On pressing ACCEPT the INPUT light will go out.

Messages readin from a tape reader must be terminated by a new-line character, in lieu of pressing ACCEPT. This will inform Executive whento start the analysis. It will be almost impossible to press any other button during the input of a message from the tape reader, but inthe event of a button or key being pressed the message will be cancelled.

3. Interpretation of Messages

(a) Acceptable Messages

If Executive is satisfied that a message from the typewriter is acceptable it is acknowledged by the message:-

O.K. - 1902/3
OK - 1904/5

which is output on the same line as the message. The instructions contained in the message are then carried out. Messages from devices other than the console typewriter are acknowledged by printing, on the typewriter, the message read from the input device. Except on the 1902/3 Executive will not output 'OK' if the message is correct.

(b) Unacceptable Messages

If Executive detects that a message is non-standard or if, for other reasons, it is impossible to put into action, it will be ignored by Executive after printing an appropriate error indication immediately after the message (regardless of the input device concerned). The INPUT light will then go out. A look-up table will give the reason for rejection of the message (see Tables at the end of PART 6).

The operator must then examine the original message and try again.

If, while a message is being typed, the typewriter is required by Executive for output, Executive will take priority; the input message will be cancelled and 'CANCEL' typed; the message being typed will be ignored. The operator will have to start the whole procedure over again, once the output message has been typed.

.4. Identification of Messages

On the 1902/3 Executive generates a newline automatically at the end of each line and precedes output message by three spaces. 'O.K.' and error messages, however, follow immediately the input message concerned.

On the 1904/5, when the INPUT button is pressed Executive generates an outward transfer of newline and precedes the input message by three spaces. The INPUT button then lights up informing the operator to start typing.

Messages other than those input from the console typewriter are printed on a new line.

These facilities make it easy for an observer to see, from the console typewriter print out, which messages are from Executive and which have been typed in.

.5 Form of Messages

All messages are of standard form and are constructed from the following components:-

(a) Directive

The first two characters of the directive, which must be typed correctly, form the first two characters of the first word. These characters must be the first ones typed after pressing the INPUT button, e.g. LO, DU, etc. The remaining part of the directive is optional e.g. LOAD, DUMP, etc.

(b) Program Name

A program name must consist of 4 characters of which the first must be alphabetic and the last three alphanumeric. [AL/14]
The name EXEC is reserved.

(c) Sub-program Number

When there are several sub-programs to a program, most messages require an indication of the appropriate sub-program on which to act. The indication is given in the form of a numeric character which immediately precedes the #of the program name. This number must be omitted if sub-programs are not used.

The value of this character should be 1, 2 or 3 for the appropriate sub-program and for the main program, rather than typing 0, this character should be omitted. Some directives such as 'LOAD' and 'FIND' must not have a sub-program number.

(d) Additional Information

A program name is followed by one or more groups of numeric characters. Each group must be separated by non-numeric characters (usually spaces) from the next. Characters within a group must not be separated from one another. These groups of numeric characters represent information such as absolute peripheral numbers, core store allocation, entry points of a program etc.

Any number required in an input message may be given in decimal form. If the octal form is used the number must be immediately preceded by an asterisk (*).

·6 Notation

The following notation will be used for the description of Input and Output messages:-

(i) Actual alphabetic characters, which are input by the Operator or which are output from Executive, will be denoted by Capital letters.

(ii) The variable part of the message, such as the program name and entry points, will be denoted by lower case letters.

An example of this notation is given below:-

OU ;s=#prog m

s being the sub-program number,
prog. " " program name,
and m being an address of the sub-program s.

The following paragraphs describe the effect of the Typewriter Input Messages.

N.B. The inclusion of the program name in the message is optional except in the case of LOAD and FIND where it is mandatory.

.1 LOAD (1902/3/4/5)

(a) LO =#prog n

Load the program which is on device n where n constitutes the absolute number of one of the acceptable load devices appropriate to the installation. The action is to read, from the device specified, a program in normal binary form, allocate peripherals and core store according to the request slip block; generate the priority of the program and sub-programs, if any, and enter the program according to the entry block. This entry block may or may not suspend the main program.

(b) LO =#prog n m x y

This message is an expansion of the above which allows the operator to specify up to 3 further parameters (on 1902/3 only one extra parameter is permitted). These parameters are numbers which may represent the size of store to be allocated to the program and/or peripheral devices to be assigned, (on 1902/3 peripheral assignment is not allowed in this form). If the decimal value of a parameter exceeds 63 that parameter is deemed to represent storage allocations; otherwise it represents the geographical address of a peripheral device. Only one parameter may refer to the store size, and only one peripheral belonging to any particular internal type group may be assigned in this way.

The amount of store quoted in such a parameter takes precedence over the figure in the request slip, but does not alter it. AL/15

Peripherals assigned in this way are assigned as "program's unit 0" of their type, and replace the first unit of the type(s) concerned, called for in the request slip (if any).

.2 FIND (1902/3/4/5)

There are two forms of this message:-

(a) FI #prog

The message causes Executive to search all unallocated and available magnetic tape decks for the specified program (or the Program Library Tape (PLT), in which case control is handed over to a suitable search routine, at the head of the PLT, which locates the desired program and then instructs Executive to load it) and then load it according to its request block.

(b) FI #prog m x y

This is very similar to the message 'LO #prog n m x y'. (See paragraph .1(b))

N.B. For further details of the LOAD and FIND messages see PART 5, Section 4.3.

.3 DUMP (1902/3/4/5)

This has two forms:-

(a) DU #prog

This message will cause Executive to search through the available unallocated tape decks until it finds a scratch tape, which it will label 'PROGRAM_name', the program then being written in binary program form onto tape.

(b) DU #prog n

This message causes the program to be output in normal binary form, on the device n, headed by a request slip block and terminated by an entry block. The device may ~~only~~ be a paper tape punch or card punch, ~~or magnetic tape unit.~~

N.B. For further details of the DUMP message see PART 5, Section 3.6.

.4 GO (1902/3/4/5)

There are two acceptable forms to this message:-

(a) GO s #prog

The action is to start (or re-start) the program or sub-program at the first (or next) instruction. In general this form of the message will be used to restart a program or sub-program after a suspension by a 160/0 or by the operator directive 'SUSPEND'. Typing "GO" after the program has been suspended due to an illegal instruction causes the offending instruction to be repeated.

(b) GO # prog m

If the program stops for any reason and cannot be continued, this message will cause the main program to restart at the storage location m, e.g. to commence an appropriate error recovering procedure. This type of instruction cannot be used to enter a sub-program directly. If the stop occurs during the running of the sub-program, the sub-program will be suspended until it is re-entered by the main program.

5. SUSPEND (1902/3/4/5)

SU # prog

This message will cause the program (or one of its Sub-programs) to be suspended until the operator wishes to restart it.

6. DELETE (1902/3/4/5)

DE # prog

This message will cause the program to be removed from the computer and, on multi-program processors, all programs in higher numbered store locations to be moved to fill in the space originally occupied by the deleted program. This directive will not succeed if there is an active peripheral which has been disengaged. In this case the program is suspended and a print out is given which will indicate the peripheral responsible. After the peripheral has been dealt with the message will have to be re-typed.

7. ALTER (1902/3/4/5)

AL # prog m a

Alter the contents of location m to a, a being an octal representation of the new contents to be placed in location m.

On the following line the 1904/5 prints the contents of the location being; the function number, the X number, the modifier (where necessary) and the value of N in octal, followed by an octal representations of the word being altered.

.8 PRINT PRIORITY LIST (1904/5)

PR

This message will output on the typewriter the priority of each program/sub-program in the machine.
e.g. 1 JACK02 etc.
The first numeral is the sub-program number, 0 if it is the master, 1 and 2 for the subprograms, and 7 if being processed by Executive (e.g. during load or dump). This numeral is followed by the four character name followed by two characters which denote the priority of the program/sub-program. The bigger the value of this priority then the higher the position in the priority list.

In the event of two programs having the same priority value being active at the same time Executive makes an arbitrary choice when deciding which one will be allowed to actually run.

If there are no programs in the machine then the reply to the directive 'PR' will be 'NIL'.

.9 REVISE PRIORITY (1904/5)

REs=#prog ab
Revise the priority of the subprogram specified by s (being 0, 1 or 2) to the value ab and change its position in the priority list according to its new priority. Instead of printing out OK this will print out the priority list in the same format as that specified for the 'PRINT PRIORITY' message.
'ab' should always be decimal.

.10 OUTPUT (1902/3/4/5)

This message has two forms. It is used as an aid in the commissioning and development of new programs.

a) OU s# prog m

Print on the console typewriter the contents of address m of the subprograms s (0, 1 or 2) of the program.

b) OU s# prog m a n

Output on device n (absolute) core store region of s# prog from word m to word m + a - 1.

If any word being output is zero, all reference to it is omitted from the output, except on 1904/5 when outputting to a paper tape punch, one 'newline' character is produced for each group of one or more consecutive zeros, or, on the printer, one blank line is produced.

The peripheral n must be a paper tape punch, or line printer and must be allocated to the program or must be a free device. In the latter, Executive temporarily allocates the device to the program.

11. GIVE (1902/3/4/5)

GI # prog n m y

Allocate peripheral n to the program as that program's unit number m. This peripheral will only be allocated if it is available. If the peripheral is magnetic tape, y is used when it is required to change the parity mode, gap length or packing density of the tape. The use of y is optional.

On the 1902/3 additional options are available with the GIVE message.

(i) If m is given the value - 1 i.e. typed as "*77777777", the peripheral n is made available to the computer for general use (being removed from an assignment to a program). This is equivalent to the TAKE or RIGHT directive on 1904/5

.11 cont'd

(ii) If m is given the value - 2 i.e. typed as '*77777776', the peripheral n is removed from the list of available peripherals and interrupts from it are ignored.
This is equivalent to the WRONG directive on the 1904/5.
(iii) A particular mode may be set for a peripheral, provided it is not assigned, by a 'GIVE' message with m = -1 and y having the appropriate value.
'y' never reverts to Standard unless changed by message or a new Executive is loaded. This is roughly equivalent to 'CHANGE' on 1904/5.
Executive does not check the action demanded by a 'GIVE' message.

.12 SPACE (1904/5)

SP

This message will cause Executive to print out, on the console typewriter, a list containing the following.

a) Core storage available for use (in words)

b) the type and numbers of available peripherals thus

SP

CORE	12928
TR	2
TP	2
LP	1

.13 TAKE (1904/5)

TA #prog n

Remove from the program's use the peripheral, of which the absolute number is n, and make this peripheral available for general use.

On 1902/3 a modified form of 'GIVE' replaces 'TAKE'

.14 ON (1902/3/4/5)

ON #prog n m p q

.14 cont'd

Give each of the bits, n, m, p. q, counting from the most significant end of program location 30, a value one. Up to 4 bits may be switched in one message on any processor.

.15 OFF (1902/3/4/5)

OF #prog n m p q

Give each of the bits n, m, p, q, counting from the most significant end of program location 30, a value zero. Up to 4 bits may be switched in one message on any processor.

.16 RIGHT (1904/5)

RIn

Make the peripheral n available to the computer for general use. This directive will only be used after the directive 'WRONG' On the 1902/3 a modified form of 'GIVE' replaces 'RIGHT'.

.17 WRONG (1904/5)

WRn

Remove the peripheral n from the computer's list of peripherals and ignore an interrupt from that device. This directive is used to inform the computer that the particular device is out of action and must not be used. On the 1902/3 a modified form of 'GIVE' replaces 'WRONG'

.18 CHANGE (1904/5)

CH n y

The switching status of the unit n is altered to that indicated by y. If a "GIVE" message containing a new y is now entered, the new y will apply only until the file is closed, when the status of the deck reverts to that which it had prior to the 'GIVE' message. On the 1902/3 a modified form of 'GIVE' replaces 'CHANGE'.

.1 General

All input messages are acknowledged by an Executive print-out message which indicates whether the input message is acceptable or otherwise.

If the message has correct format and can be obeyed the characters 'O.K.' are printed out, on the same line as the input message.

If the message has incorrect format or cannot be obeyed, Executive will cause the typewriter to print out, on the same line as the input message, 'ERROR', This is followed by, on the 1902/3, two characters, the first being alphabetic and the second numeric. On the 1904/5 'ERROR' is followed by an alphabetic character. The error conditions which Executive can recognise are given in an Error look-up table to determine the type of error which has been made. (See tables A and B at the end of PART 6).

The operator may find that the message he is typing is incorrect, in which case he stops typing and presses the RUBOUT Key. 'CANCEL' is then typed out on the same line as the incorrect message.

.2 1902/3 Messages from Executive

Peripheral occurences are reported by Executive by means of the following messages:-

(continued)

UNIT n FIX	This is the result of a peripheral incident such as a machine getting low in cards, or paper running out. The operator will attend to the device and re-engage unit n.
UNIT n ERR	This indicates that the transfer on this device is in error e.g. a parity failure occurs or a peripheral becomes faulty. The program is automatically suspended.
UNIT n FREE	This is printed out as a result of a successful 151 order, requesting the release of a peripheral, or close file instructions.
UNIT n USED	This is printed out as a result of a successful 156 order requesting the allocation of a peripheral.

.3 1902/3 Messages from the Program

There are three messages from Executive which are printed out to inform the operator of the state of the program currently held in the central processor.

(continued)

HALTED	This is the result of a 160/0 or 161/0 order, requesting the suspension of the current program.
DISPLAY	This is the result of a 160/1 or 161/1 order, reporting some event taking place in the program.
DELETED	This is the result of a 160/2 or 161/2 order, requesting the deletion of the current program.

Each of these messages may be followed by a number of alphanumeric characters except in the case of the 161 order where the number of characters is limited to two.

.4 1902/3 Messages output during the running of Magnetic Tape Programs

The following messages may be printed out during the running of programs that use magnetic tapes.

(continued)

UNIT n 10R	This is a purely informative message which appears after every tenth repeat in a tape pass. It does not mean 10 repeats of any particular block (it may be 1 repeat on each of 10 blocks or any combination of repeats making up 10).
LDT SCRATCH TAPE LDT file-name ~~TAPE~~ LDT PROGRAM TAPE UNIT n LDT ~~LTD~~	All these message are results of a program calling for a tape which is not yet on a tape deck. The operator should load the tape and type GO. The programmer should as far as possible make sure that his operating instructions ask for tapes to be loaded in good time.
unit number - file name	This message informs the operator that a file has been opened on the specified unit in the message and that the specified name has been given to that file.

Output Messages

.5 1904/5 Messages from Executive

The following messages are reported by Executive on peripheral occurrences.

s # prog; UNIT q :- FIX	Message requesting operator attention on the specified unit e.g. disengaged.
s # prog; UNIT q :- BUSY	A message informing the operator that some action, particularly the deletion of a program, cannot take place immediately, due to the activity of the device stated.
s # prog; UNIT q :- R&FX	Requests that the operator repositions the medium on the specified unit
s # prog; UNIT q :- FREE	This is the result of a 151 order asking for the peripheral to be released from a programmer's allocation.
s # prog; UNIT q :- USED AS X	This is the result of a 156 order requesting the allocation of a specified peripheral as program's unit X of the appropriate type.
s # prog; UNIT q :- RPTD	Following a parity failure on the card Punch, this indicates that Exec. has repeated the transfer data for re-punching.

s # prog; UNIT q :- FAIL	Max no of attempts have been made (card punch)
s # prog; UNIT q :- FAILw	w=0 Max no of attempts have been made on the block. w=1 It is suspected that Executive has lost its place on the tape due to 'block-splitting'. w=2 A read transfer has been in progress for five seconds and, since nothing has been read from the tape, it is assumed that the tape is a blank. w=3 A major failure due to (a) an attempt to read back beyond beginning of tape (b) an attempt to write with W.P.R. absent. (c) unit being put off line after being assigned to the program.
s # prog; CLOCKED:- ab	Message to indicate that the program has used ab seconds of central processor time. It will not appear if ab < 01.
s # prog; ILLEGAL: e zzzz	This message reports a programming error. e is the type of error (See Table C) and zzzz is the address of the rejected instruction On the following line the contents are then printed out as a) an instruction b) an octal number.

.6 1904/5 Messages from the Program

There are three messages from Executive which are printed out to inform the operator of the state of a particular program currently held in the central processor.

s #prog; HALTED:- message	The result of a 160/0 or 161/0 order, requesting the suspension of the specified program/sub-program.
s #prog; DISPLAY:-message	The result of a 160/1 or 161/1 order, reporting some event taking place in the program.
s #prog; DELETED:-message	The result of a 160/2 or 161/2 order, requesting the deletion of the specified program/sub-program.

Each of these messages may be followed by a message in the form of a number of alphanumeric characters except in the case of the 161 order where the number of characters is limited to two.

.7 1904/5 Messages output during the running of Magnetic Tape Program

The following messages may be printed out during the running of programs that use magnetic tapes.

CLOSED; UNIT q :- file name, n REPEATS	This message indicates that the current program has completed its operation on the specified tape deck and that the tape is no longer assigned to that program. The deck concerned is still engaged and the tape can be opened by an appropriate 157 order from any program.
UNLOAD; UNIT q:- file name, n REPEATS	AS CLOSED except that the deck is disengaged, usually so that the tape can be removed to storage.
s # prog; LOAD file name	A request from the named program to load the specified tape.

TABLE A

1902/3 ERROR CODES

ERROR	M0	Format Error.
ERROR	M1	Item Error.
ERROR	M2	Timing Error.
ERROR	L0	Incorrect name in data.
ERROR	L1	Data not in binary format.
ERROR	L2	Request slip received at wrong time.
ERROR	L3	Store Limit failure caused by current action.
ERROR	L4	Sum-check Error.
ERROR	D0	Illegal Character.
ERROR	D1	No such day.
ERROR	D2	Impossible date.
ERROR	D3	No such month.
ERROR	D4	Improbable year.
ERROR	D5	Day and date do not match.

TABLE B

1904/5/9 EXECUTIVE ~~MK. 2~~ AL11

OPERATOR ERRORS

ERROR	
A	Program or sub-program not in
B	Incorrect data after message
C	No such message
D	Program in Executive action
E	Program name in use
F	Illegal message from program
G	Four programs in the machine
H	Not a load, dump or output device
I	-
J	Peripheral not available
K	No core available
L	Peripheral busy
M	Peripheral of this type and unit no. already allocated
N	Address beyond program's limit.

TABLE C

PROGRAM RUNNING ERRORS

ILLEGAL	GROUP 15	OTHERS
A	Illegal order	Illegal order
B	Reservation violation	Reservation violation
C	Peripheral or file not allocated to program	
D	Not a correct device for this order	
E	-	
F	Transfer too big for device	163, N(M) ≠ 0, with member X active
G	Unit X of type N(M) already allocated	162, 163 with member X non-existent
~~H~~	~~Illegal from sub-program~~	
I	-	-
J	Peripheral or file not available	-
K	Zero or null transfer	Core request inadequate (165 X/4)
L	Protected file (Not to be written on)	160, 161, 164 with X or 165 with N(M) too large
M	Illegal mode	-
*X	Illegal order or Reservation Violation (* Detected by Hardware)	

TABLE D

PROGRAM LOADING ERRORS

FAULT	
A	Not Binary block
B	Block destination beyond program's limit
C	Block checksum fail
D	Not a Request block when expected
E	A Request block when not expected
F	Block count fail
*J	Peripheral not available
*K	Core request inadequate or not available
*L	Name in Request block not name in message
*	Program is deleted

After a rectifiable error, the loading action may be continued by typing the message GO 7 #PROG. In other cases, Executive may be correctly reset by deleting the program.

PART 7 CONTENTS PAGE No.

Section 0	Instruction Times 1901/2/3/4/5/6/7	1
Section 1	Scientific Attachment Times 1901/2/3	1
Section 2	Floating Point Times (Hardware)	1
	0.1 1905	1
	0.2 1907	3

INSTRUCTION TIMES 1901/2/3/4/5/6/7

TIMES IN MICROSECONDS

CENTRAL PROCESSOR	1901	1902	1903	1904/5	1906/7			
STORE CYCLE TIME	6	6	2 (1.8*)	2	1.25		2.25	
					A	B	A	B
000 LDX	34	18	7	7	2.5	1.875	4.5	2.875
001 ADX	34	18	7	7	2.5		4.5	3.5
002 NGX	34	18	7	7	2.5		4.5	3.5
003 SBX	34	18	7	7	2.5.		4.5	3.5
004 LDXC	34	18	7	7	2.5	1.875	4.5	2.875
005 ADXC	34	18	7	7	2.5		4.5	3.5
006 NGXC	34	18	7	7	2.5		4.5	3.5
007 SBXC	34	18	7	7	2.5		4.5	3.5
010 STO	34	18	7	7	2.5	1.875	4.5	2.875
011 ADS	34	18	7	7	3.125	2.5	5.125	3.5
012 NGS	34	18	7	7	3.125	2.5	5.125	3.5
013 SBS	34	18	7	7	3.75	2.5	5.75	3.5
014 STOC	34	18	7	7	2.5	1.875	4.5	2.875
015 ADSC	34	18	7	7	3.125	2.5	5.125	3.5
016 NGSC	34	18	7	7	3.125	2.5	5.125	3.5
017 SBSC	34	18	7	7	3.75	2.5	5.75	3.5
020 ANDX	34	18	7	7	2.5		4.5	3.5
021 ORX	34	18	7	7	3.125	1.875	5.125	2.875
022 ERX	34	18	7	7	2.5		4.5	3.5
023 OBEY	25	7	3	3	1.875		2.875	
024 LDCH	34	18	7	7	2.5		4.5	3.5
025 LDEX	34	18	7	7	2.5		4.5	3.5
026 TXU	34	19	7	7	2.5		4.5	3.5
027 TXL	34	19	7	7	2.5		4.5	3.5
030 ANDS	34	18	7	7	3.125	2.5	5.125	3.5
031 ORS	34	18	7	7	3.75	2.5	6.75	3.5
032 ERS	34	18	7	7	3.125	2.5	5.125	3.5
033 STOZ	31	12	5	7	2.5	1.875	4.5	2.875
034 DCH	31	18	7	7	3.125	2.5	5.125	3.5
035 DEX	32	18	7	7	3.125	2.5	5.125	3.5
036 DSA	32	18	7	7	3.125	2.5	5.125	3.5
037 DLA	34	18	7	7	3.125	2.5	5.125	3.5
FOR MODIFICATION ADD	62	6	2	2	0.625		0.625	

[AL12]

A Times in this column are for functions operating between accumulators and core store.

B Times in this column are for functions operating between two accumulators.

* Since January 1966, new 1903 machines have been fitted with a 1.8μs store Machines with a 1.8 μs store are about 3% faster than those with a 2μs store.

INSTRUCTION TIMES 1901/2/3/4/5/6/7

CENTRAL PROCESSOR	1901	1902	1903	1904/5	1906/7			
STORE CYCLE TIME	6	6	2(1.8)	2	1.25		2.25	
					A	B	A	B
040 MPY N = 0	MULTIPLY 1902 TIMES BY 2 (EXTRACODE)	210	80	40	10.9		12.9	11.9
N ≠ 0		435+52s+30t	235+19s+12t					
041 MPR N = 0		210	80	40	10.9		12.9	11.9
N ≠ 0		435+52s+30t	235+19s+12t					
042 MPA N = 0		222	84	41	10.9		12.9	11.9
N ≠ 0		447+52s+30t	239+19s+12t					
043 CDB CD		390	155	29	11.9		13.9	12.9
CND		120	50					
044 DVD		2440	960	45	21.65	21.025	23.65	22.025
045 DVR		2590	1020	43	24.55	23.925	26.55	24.925
046 DVS		2420	950	44	21.65	21.025	23.65	22.025
047 CBD		455	180	28	15.0	13.75	19.0	16.75
050 BZE	21/19	13	5	5	2.5+ 3.75*		3.5+ 5.75*	
052 BNZ	21/19	13	5	5	2.5+ 3.75*		3.5+ 5.75*	
054 BPZ	21/19	13	5	5	2.5+ 3.75*		3.5+ 5.75*	
056 BNG	21/19	13	5	5	2.5+ 3.75*		3.5+ 5.75*	
					** CMPCT STORE	** EXT. STORE	** CMPCT STORE	** EXT. STORE
060 BUX	24/22	12	5	5	2.7+ 3.95*	2.5+ 3.75*	3.7+ 5.95*	3.5+ 3.75*
062 BDX	24/22	12	5	5	2.7+ 3.95*	2.5+ 3.75*	3.7+ 5.95*	3.5+ 3.75*
064 BCHX	24/22	12	5	5	2.7+ 3.95*	2.5+ 3.75*	3.7+ 5.95*	3.5+ 3.75*
066 BCT	ILLEGAL	ILLEGAL	ILLEGAL	ILLEGAL	2.7+ 3.95*	2.7+ 3.95*	3.7+ 5.95*	3.7+ 5.95*
MODIFICATION ADD	2	6	2	2	0.625		0.625	

A Times in this column are for functions operating between accumulators and core store.
B Times in this column are for functions operating between two accumulators.
\+ This is the time for jumps in compact jump mode and relative jumps in extended jump mode
* This is the time for replaced jumps in extended jump mode.
Where two figures are given, the second one applies to unsuccessful jumps.
s is the number of significant bits in the operand.
t is the number of 1's contained in the operand
CD = Character a digit
CND = Character not a digit
Extracode times are underlined
** ~~These headings refer to the physical nature of the store and not modes.~~

1900 SDORM/JRWA
AL9

** These headings refer to the physical nature of the store and not modes.

INSTRUCTION TIMES 190/2/3/4/5/6/7

TIMES IN MICROSECONDS

CENTRAL PROCESSOR STORE CYCLE TIME	1901 6	1902 6	1903 2(1.8)	1904/5 2	1906/7 1.25	 2.25
070 CALL	26	13	6	8	2.5 + 3.75 *	3.5 + 5.75 *
072 EXIT	19	12	4	5	2.5	3.5
074						
x=o BRN	14	8	4	3	2.5 + 3.75 *	3.5 + 5.75 *
x=1 BVS	16/14	8	4	3	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=2 BVSR	16/14	8	4	3	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=3 BVC	16/14	8	4	3	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=4 BVCR	16/14	8	4	3	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=5 BCS	16/14	8	4	3	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=6 BCC	16/14	8	4	3	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=7 BVCI	16/14	8	4	3	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
076 BFP						
x=o	ILLEGAL	ILLEGAL	ILLEGAL	ILLEGAL	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=1	ILLEGAL	ILLEGAL	ILLEGAL	ILLEGAL	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=2	ILLEGAL	ILLEGAL	ILLEGAL	ILLEGAL	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=3	ILLEGAL	ILLEGAL	ILLEGAL	ILLEGAL	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=4	ILLEGAL	ILLEGAL	ILLEGAL	ILLEGAL	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
x=5	ILLEGAL	ILLEGAL	ILLEGAL	ILLEGAL	2.5/1.875+ 3.75/3.125*	3.5/2.875+ 5.75/5.125*
FOR MODIFICATION ADD	82	6	2	2	0.625	0.625

[AL/12]

\+ This is the time for jumps in compact jump mode and relative jumps in extended jump mode

* This is the time for replaced jumps in extended jump mode
Where two figures are given the second one applies to ~~successful~~ jumps

[4]

1900 SDORM/JRWA

AL9

INSTRUCTION TIMES 1901/2/3/4/5/6/7

TIMES IN MICROSECONDS

CENTRAL PROCESSOR	1901	1902	1903	1904/5	1906/7	
STORE CYCLE TIME	6	6	2 (1.8)	2	1.25	2.25
100 LDN	28	12	5	5	1.875	2.875
101 ADN	28	12	5	5	1.875	2.875
102 NGN	28	12	5	5	2.5	3.5
103 SBN	28	12	5	5	2.5	3.5
104 LDNC	28	12	5	5	1.875	2.875
105 ADNC	28	12	5	5	1.875	2.875
106 NGNC	28	12	5	5	2.5	3.5
107 SBNC	28	12	5	5	2.5	3.5
110 Nt = 0 SLC	25 + 5N	20 + 2N	10 + 2N	6 + N	2.5 + .625r	3.5 + .625r
Nt = 1 SLL	25 + 5N	20 + 2N	10 + 2N	6 + N	2.5 + .625r	3.5 + .625r
Nt=2, 3SLA	25 + 5N	20 + 2N	10 + 2N	6 + N	2.5 + .625N	3.5 + .625N
112 Nt = 0 SRC	29 + 5N	20 + 2N	10 + 2N	6 + N	2.5 + .625r	3.5 + .625r
Nt = 1 SRL	29 + 5N	20 + 2N	10 + 2N	6 + N	2.5 + .625r	3.5 + .625r
Nt = 2 SRA	29 + 5N	20 + 2N	10 + 2N	6 + N	2.5 + .625N	3.5 + .625N
Nt = 3 SRAV	29 + 5N	20 + 2N	10 + 2N	6 + N	2.5 + .625N	3.5 + .625N
114 NORM VSET		339	139	15 + N	3.525 +.825N	4.525 + .825N
VCLEAR		422 + 4N	174 + 4N			
111 N = 0		50 TO 110	20 To43			
Nt = 0 SLC N < 24		340 + 4N	140 + 4N	15 + N	2.5 + 1.25r	3.5 + 1.25r
N ≥ 24		295 + 4N	65 + 4N			
Nt = 1 SLL N < 24		280 + 4N	115 + 4N	15+N	2.5 + 1.25r	3.5 + 1.25r
N ≥ 24		120 + 2N	20 + 2N			
Nt=2,3,SLA N <23		335 + 4N	135 + 4N	15 + N	2.5 + .625N	3.5 + .625N
N ≥ 23		240 + 2N	100 + 2N			
113 N=0		58 TO 97	24 TO 39			
Nt = 0 SRC N <24		340 + 4N	140 + 4N	15 + N	2.5 + 1.25r	3.5 + 1.25r
N ≥ 24		295 + 4N	65 + 4N			
Nt = 1 SRL N <24		280 + 4N	115 + 4N	15 + N	2.5 + 1.25r	3.5 + 1.25r
N ≥ 24		120 + 2N	20 + 2N			
Nt = 2 SRA N <24		335 + 4N	140 + 4N	15 + N	2.5 + .625N	3.5 + .625N
N ≥ 24		140 + 2N	30 + 2N			
Nt=3 SRAV N <24		395 + 4N	165 + 4N	15 + N	2.5 + .625N	3.5 + .625N
N ≥ 24		205 + 2N	55 + 2N			
115 NORM VSET		447	187			
V CLEAR N <23		636 + 6N	267 + 6N	16 + N	6.225 +.825N	7.225 + .825N
N ≥ 23		469 + 4N	154 + 4N			
For Modification Add	62	6	2	2	0.625	0.625

MULTIPLY 1902 TIMES BY 2/3 (EXTRACODE)

[11]

[AL12]

r = S + u where S is the rounded result of $\frac{N}{4}$, and u = 0 for N EVEN
and u = 1 for N ODD

N is the number of places of shift
Extra code times are underlined

INSTRUCTION TIMES 1901/2/3/4/5/6/7

TIMES IN MICROSECONDS

		1901	1902	1903	1904/5		1906/7			
CENTRAL PROCESSOR STORE CYCLE TIME		6	6	2 (1.8)	2		1.25		2.25	
116	MVCH	ILLEGAL	ILLEGAL	ILLEGAL	ILLEGAL		8.125+3.75N		11.125+5.75N	
							A	B	A	B
117	SMO	ILLEGAL	ILLEGAL	ILLEGAL	ILLEGAL		2.5	2.5	4.5	3.5
120	ANDN	28	12	5	5		1.875		2.875	
121	ORN	28	12	5	5		2.5		3.5	
122	ERN	28	12	5	5		2.5		3.5	
123	NULL	19	8	4	3		2.7		3.7	
124	LDCT	74	12	5	6		1.875		2.875	
125	MODE	19	7	3	4		1.875		2.875	
126	MOVE	MULTIPLY 1902 TIMES BY 2/3 (EXTRACODE) →	145+72N	60+28N	14+4N		**7.5+2.5N		**10.5+4.5N	
127	SUM		170+36N	70+14N	14+3N		4.37+1.875N		6.375+2.875N	
					1904	1905	1906	1907	1906	1907
130	FLOAT		855 - 6N	425-6N	300	300	125	SEE SECTION 2 PAGE 3	226	SEE SECTION 2 PAGE 3
131	FIX		1110-4N	500-4N	325	325	125		226	
132	FAD		1900	750	540	SEE SECTION 2 PAGE 1	185		334	
133	FSB		1900	750	540		185		334	
134	FMPY		4000	1600	830		290		523	
135	FDVD		7250	2900	820		260		469	
136	LFP		550	220	265		115		208	
137	SFP		575	230	285		110		199	
150	SUSBY			TIMES ARE NOT AVAILABLE FOR THESE FUNCTIONS AS THEY ARE DEVICE OR SITUATION DEPENDENT						
151	REL									
152	DIS									
153	-									
154	CONT									
155	SUSDP									
156	ALLOT									
157	PERI									
160				TIMES ARE NOT AVAILABLE FOR THESE FUNCTIONS AS THEY ARE DEVICE OR SITUATION DEPENDENT						
x=0	SUSTY									
x=1	DISTY									
x=2	DELTY									
161										
x=0	SUSWT									
x=1	DISP									
x=2	DEL									
162										
x=1	SUSMA	ILLEGAL								
x=2	SUSEN	ILLEGAL								
163	AUTO	ILLEGAL								
164	SUSAB	ILLEGAL								
165	GIVE									
FOR MODIFICATION ADD		6/2	6	2	2		0.625		0.625	

[11]

[AL/12]

A AND B - SEE FOOTNOTE ON PAGE 1 OF SECTION 0.

** THESE ARE FOR MOVES FROM EVEN TO ODD AND FROM ODD TO EVEN STORE ADDRESSES.
FOR EVEN TO EVEN AND ODD TO ODD STORE ADDRESSES THE FOLLOWING TIMES APPLY.
1,1 CORE STORE 10.0 + 1.25N 2.25 CORE STORE 15.0 + 2.25N

EXTRACODE TIMES ARE UNDERLINED

The scientific attachment is available in two versions; one for performing the multiply, divide, and shift functions, and the other for performing all the functions shown below:

All times in microseconds:

Instruction	Central Processors 1901	1902	1903
040 MPY 041 MPR	97.2	91.5	71.7
042 MPA	99.8	96.7	72.7
044 DVD	101.9	99.4	75.4
045 DVR	105.5	103.0	79.0
046 DVS	101.9	99.4	75.4
111 SLC, SLL, SLA 113 SRC, SRL, SRA, SRAV	54 + 1.8 N	48.5 + 1.8 N	28.5 + 1.8 N
114 NORM	56 + 1.8 N	50.5 + 1.8 N	30.5 + 1.8 N
115 NORM	56 + 2.7 N	50.5 + 2.7 N	30.5 + 2.7 N
132 FAD 133 FSB	59.6*	52.1*	40.1*
134 FMPY	149.4	143.9	129.9
135 FDVD	159.3	153.8	143.8
136 LFP	36.0	30.0	18.0
137 SFP	41.0	31.0	19.0
For Modification Add:	2	6	2

*These figures include estimates for a total of 4 normalizing shifts. The figures given for the shift and normalize functions are average times. N is the number of places of shift.

FLOATING POINT TIMES 1905

1905 FLOATING POINT UNIT

(1) The process of performing a floating point instruction takes place in a number of distinct phases. In phase 1 of the instruction is fetched from core store by the central processor; in phase 2 the central processor loads the operand into the floating point unit, and in phase 3 the floating point unit is busy with that instruction. If modification is called for, this is done by the central processor and takes place between phases 1 and 2.

(2) The floating point unit on the 1905 is an autonomous part of the computer. As soon as phase 1 and phase 2 of a floating point instruction have been completed, the central processor is free to continue with fixed point instructions while the floating point unit is, at the same time, carrying out phase 3 of the first floating point instruction.

(3) If the central processor should meet a second floating point instruction while the floating point unit is busy with phase 3 of the first, the central processor will be able to execute the second instruction only up to the end of phase 1 (and modify if necessary). Since the floating point unit on the 1905 has no buffer, the central processor will not be able to continue with further orders, either fixed or floating point, until the floating point unit has completed phase 3 of the first instruction.

(4) Since neither the LFP nor the SFP instructions involve the floating point unit in actual calculation, there can be no overlap for any instruction (either fixed point or floating point) that follows them. If either LFP or SFP follow one of the other floating point orders however, then overlap applies in the normal way.

FLOATING POINT TIMES 1905

TIMES IN MICROSECONDS

	PHASE 1	PHASE 2	PHASE 3	TOTAL
	INSTRUCTION FETCH	OPERAND LOAD	FLOATING POINT WORKING	(WITH NO OVERLAP)
132 FAD	2	4	9(7)	15(13)
133 FSB	2	4	9(7)	15(13)
134 FMPY	2	4	25(14)	31(30)
135 FDVD	2	4	47(45)	53(51)
136 LFP	2	4	-	6
137 SFP	2	6	-	8

MODIFICATION of a floating point instruction takes place between phases 1 and 2 and may be taken to add 2μs to the time for phase 1 of that instruction.

PLEASE NOTE THE FOLLOWING 2 POINTS

(1) The times given in the table are typical, those shown in brackets are minimum times.

(ii) FLOAT and FIX are done by extracode-for times see section 0 page 5.

1907 FLOATING POINT UNIT

(1) The process of performing a floating point instruction takes place in a number of distinct phases. In phase 1 the instruction is fetched from core store by the central processor; in phase 2 the central processor loads the operand into the floating point unit, and in phase 3 the floating point unit is busy with that instruction. If modification is called for, this is done by the central processor and takes place between phases 1 and 2.

(2) The floating point unit on the 1907 is an autonomous part of the computer. As soon as phase 1 and 2 of the first floating point instruction have been completed the central processor is free to continue with fixed point instructions while the floating point unit is at the same time, carrying out phase 3 of the first floating point instruction.

(3) Assuming phase 3 of the first instruction is still in progress then when a second floating instruction is encountered it will be executed up to the end of phase 2. That is to say, the operand will be loaded into the floating point unit, there being a buffer available on the 1907 unit to hold this operand until the unit has finished working with the first order. If, however, the second instruction is FIX or SFP or the first has been FMPY the second instruction will be executed only to the end of phase 1.

(4) Assuming phase 3 of the first instruction is still in progress, then when a third floating point instruction is encountered it will be executed to the end of phase 1. The central processor is now held up as regards further fixed point instructions or floating point instructions, until phase 3 of the first floating point instruction is completed.

(5) Since neither the LFP or the SFP instructions involve the floating point unit in actual calculation, there can be no overlap for any instructions (either fixed or floating point) that follows them. If either LFP or SFP follow one of the other floating point orders however, then overlap applies in the normal way.

	PHASE 1		PHASE 2		PHASE 3		TOTAL	
	INSTRUCTION FETCH		OPERAND LOAD		FLOATING POINT WORKING		(WITH NO OVERLAP)	
STORE CYCLE TIME	1.25	2.25	1.25	2.25	1.25	2.25	1.25	2.25
130 FLOAT	1.25	2.25	2.5	3.5	6.5*	6.5*	10.15*	12.15*
131 FIX	1.25	2.25	2.5	3.5	3.0+	3.0+	6.95*	8.95+
132 FAD	1.25	2.25	2.5	3.5	3.5**	3.5**	7.25**	9.25**
133 FSB	1.25	2.25	2.5	3.5	3.5**	3.5**	7.25**	9.25**
134 FMPY	1.25	2.25	2.5	3.5	10.5 ≠	10.5 ≠	14.25 ≠	16.25 ≠
135 FDVD	1.25	2.25	2.5	3.5	24.5 ≠	24.5 ≠	28.25 ≠	30.25 ≠
136 LFP	1.25	2.25	1.25	2.25	-	-	2.5	4.5
137 SFP	1.25	2.25	1.25	2.25	-	-	2.5	4.5

* Assumes 4 Normalising Shifts ** Assumes 3 Normalising Shifts
\+ Assumes 2 Normalising Shifts ≠ Assumes 1 Normalising Shift.

MODIFICATION of a floating point instruction takes place between phases 1 and 2 and may be taken to add 0.625μs to the time for phase 1 of that instruction. Since modification is carried out by the central processor, modification of a second or third floating point order can be completed while the floating point unit is busy with the first.

PLEASE NOTE THE FOLLOWING 4 POINTS:

(i) The times given assume $N \geq 8$ i.e. they do not refer to the hardware accumulators. If $N < 8$ add 3.75μs (fast store) or 4.75μs (slow store) to phase 2.

(ii) When a floating point instruction is held up (i.e. a second or third instruction is waiting for phase 3 of a first order to be completed) 0.5μs must be added to phase 3 of the waiting instructions.

(iii) The times given in the table assume optimised operands. If non-optimised operands add 1.25μs (fast store) or 2.25μs (slow store) to phase 2.

(iv) Times for the 076 (BFP) instruction are given in section 0 page 3.