UNIVAC
DIVISION OF SPERRY RAND CORPORATION

INTERCOMMUNICATION

TO: Branch Managers
Area Managers
Regional Managers
International Division
Washington Office

FROM (NAME): W. G. Crowell

LOCATION & DATE: Whitpain - August 12, 1964

DEPARTMENT: UNIVAC 1050 Marketing

CARBONS:

SUBJECT: PRELIMINARY DOCUMENTATION FOR UNIVAC 1050 FREESTANDING SYSTEM.

Enclosed is a manual which represents the software package that currently exists on the UNIVAC 1050.

This is not the official documentation of the UNIVAC 1050 Software package and it doesn't represent the total scope of UNIVAC 1050 software. The information contained within this manual is preliminary and is subject to changes by Systems Programming as other items and features are added.

This software is being used by early Customers and UNIVAC Field Personnel at Whitpain today. It has been very effective in the construction of Bench Mark Demonstrations.

It is suggested that any person programming the UNIVAC 1050 today study this manual.

Two (2) copies of this manual have been sent to each Branch and Area Office. There are no more copies available and additional copies cannot be ordered.

This information and any possible revisions to this information will eventually be published in New York by Systems Programming Library Services as part of the official UNIVAC 1050 documentation.

W. G. CROWELL

Enclosure
WGC/gmd

TABLE OF CONTENTS

This Manual consists of excerpts from other documentation and therefore, there is no consistency or sequence in the page numbering.

The sections contained in this manual are in the following order:

CALLS FOR ROUTINES ON SYSTEMS TAPE

CALL		
$AJAX	Tape Utility	80 or 90 column row or serial
$ŌUS1	Object Utility Service	80 Column row or serial
$ŌUS4	Object Utility Service	90 Column row or serial
$TDMP	Prints OPS Tape Dump(s) from Servo 1	
$TPŌX	Prints Data Tapes from Servo 1	
$0001	PAL Assembler	80 or 90 column row
$0101	PAL Assembler	80 column serial
$1001	PAL Assembler	90 column serial

PAL TAPE ASSEMBLER NOTES

1. There is no limitation on the number of labels used in a PAL assembly.

2. The size of the combined PROC and NAME table and FORM table is 800 locations for 8K of memory. For each module of memory over 8K add 4096 to this table capacity. A PROC or NAME entry requires 13 locations. 315/4K

   If an assembler tape has more PROCS or NAMES than the table can hold, the PROCS that are not stored in the table will be missing when a call is made for those PROCS; however, if an asterisk is placed in column 18 of a call line the PROC will be found regardless of whether it is in the table.

3. An additional directive has been added to the PAL Tape Assembler.

   SEGJP

   It is often necessary to divide a program into segments which may operate at different times in store. The SEGJP directive provides this facility. The SEGJP is similar to the END directive in that it causes a "T" block to be produced containing a jump instruction to the label or address contained in the operands field.

   In addition an "S" block also produced similar to the "R" block of the first segment but without the high address, starting location and number of characters fields.

   The assembler will create a segment ID field for the next segment by adding a decimal "01" to the program or last segment ID obtained from columns 7-10 of the BEGIN directive. If column 9 of the BEGIN directive is greater than 9, a decimal "1" will be added. See OPS Instructions to load a segment.

4. The PAL Tape Assembler produces output object code on tape 1 only. See documentation on "OPUS" to produce cards from tape output.

5. UNIVAC software uses labels that start with the letters X, Y and Z. The letters U, V and W are also reserved for future expansion.

APPENDIX 3

Operating Instructions for 1050 Two or Three Tape Assembler

Console Sense Switches:

1. Sense Switch 1 controls source code location.
   Off - Assembler expects source program or call card (See console) in reader.

   On - Assembler will stop to allow key in via the trace switches of the 4 character program ID, appearing on the BEGIN line of the program to be assembled from tape. (See console).

2. Sense Switch 2 controls printing of assembly listing.
   Off - Entire listing will be printed.
   On - Procedure generated lines will not be printed.

PRINTER:

1. Depress off-line button until light is on.
2. Set paper 3 holes above sprocket.
3. Depress off-line button until light is off.

READER:

1. Place assembler call card ($0001 in columns 1 to 5) in front of BEGIN card. Place program to be assembled in input magazine followed by about 1/2 lb. blank cards.
2. Place cards in reader face down, 9 edge leading.
3. Depress power-on button until light is on.
4. Depress magazine-load button.
5. Depress clear button after magazine-load light is off.

UNISERVOS:

1. Mount PAL assembler on servo Ø with write enable ring in.
2. Mount blank tape on servo 1 with write enable ring in.
3. If source program is to be assembled from tape, mount source code library on servo 2 with write enable ring removed.
4. Bring all tapes to load point.

CONSOLE:

1. Depress clear button.
2. Depress load tape MODE button.
3. Depress program start.
4. Depress continuous MODE button.
5. Depress program start button (loads OPS). STOP (30 070001 60) Operating System ready to load.
6. If Assembler call card and source program cards are in the reader depress program start to assemble.

   STOP (30 070001) End of Assembly, Operating System ready to load.
7. If program to be assembled is from tape and card reader is available.

   a. Place source program call card in reader after $0001 card. This card contains $RRRR in columns 1 to 5 and RRRR must be identical to columns 7 to 10 appearing on the BEGIN line of the program to be assembled.

   b. Depress program start to assemble.
8. If program to be assembled is on tape and card reader is not available.

   a. At STOP (30 070001 60) depress Operator Request button.

   b. Set trace mode to PROC.

   c. Set trace switches to (000303_8) depress program start.

   d. Program stops (30 077000 60) to allow key in of next two characters of program ID. Set trace switches to (000304_8).

   e. Depress Sense Switch 1.

   f. Depress program start.

   g. STOP (30 017001 60) assembler ready for key in of program ID.

h. Set up binary value of first 2 characters of program ID in trace switches. Depress program start.

i. STOP (30 077000 60) Set up binary value of next 2 characters of program ID in trace switches. Depress program start (assemble).

OUTPUT

Object Code - output servo 1.

Listing - printer.

APPENDIX 6 - PAL STOPS

Display	Pass	Description	Recovery
017771	1	No BEGIN card or improper call card.	Reload proper BEGIN card or Call card. Depress start to continue assembly.
017771	2	Bad tape	Label not on tape. Restart.
11000X	1	Reader error	Replace 'X' cards in the input magazine. Depress clear on the reader. Depress program start. NOTE: 'X' does not always match number of cards in error stacker. Remove excess cards from normal stacker.
100000	3 & 4	Printer error	Turn printer off-line. Return printer to normal condition. Depress clear on printer. Turn printer on-line assembly will continue.
070001	OPS	Operating System	Read to load.
077000	1	ID KEY IN	Set up 2nd 2 characters of ID KEY IN trace switches. Depress program start.
017001	1	ID KEY IN	Set up 1st 2 characters of PID of program to be assembled from tape.
040U44	ALL	Tape error	Tape block count wrong, restart.
140U55	ALL	Tape error	Restart.
140U66	ALL	Tape parity	Assembler has rocked tape 5 times. To continue depress program start. Assembler will attempt to recover.

SQEEZE ROUTINE

This routine provides a method for making corrections to absolute object code decks produced by the PAL assembler.

Input

All input corrections are punched in octal starting in column 20 as follows:

1 19	20 25	26-27	28 79	80
BLANK	AAAAAA	LL	CCCC CCCCC	BLANK

Columns 20-25 contain the six digit octal address of the first character to be corrected. Columns 26-27 contain the octal number of characters to be changed. Columns 28-79 contain the data in octal to be converted to PAL format. Up to 26 characters located in contiguous memory may be altered by one change card.

Output

The output is punched in PAL format. Input cards will be compressed as much as possible until a change to a non-contiguous location is encountered or until the output card is filled.

The squeezed output is placed immediately preceding the last card of the object deck.

The following fields on the output card will be blank:

1. Card Sequence Number
2. Relocation Mask
3. Check Sum Field
4. Utility
5. Program Identification

2 CARD LOAD ROUTINE

The source card deck for the 2 card loader is supplied for an 8K configuration. This may be adjusted for any configuration by altering the second card of the source deck as follows:

CARD	LABEL		OP'N	OPERAND
(2nd card)	CARDΔ	Δ	EQUΔΔΔΔ	0X7200

X is 1 for 8K store (017200)
X is 2 for 12K store (027200)
:
:
X is 7 for 32K store (077200)

After altering the source deck, assemble the source deck for the desired 2 card loader. After assembly remove the R and T cards (first and last cards respectively) of object deck. The resultant 2 cards will load any program within the designated store configuration.

Operating Instructions

1. Place 2 card loader ahead of program to be loaded in the input hopper of the High Speed Reader.
2. Depress 'load card' - program start.
3. Depress 'CONT' - program start.
4. Program will be loaded and executed.

MEMORY PRINT ROUTINE (CARD SYSTEM)

The Source Card Deck for the Memory Print is supplied for an 8K configuration. This may be adjusted to any other size store by the alteration of two cards as follows:

first card:

Card	label		op'n	operands
00010Δ	DUMPΔ	Δ	BEGINΔΔ	0X5200

X is 1 for 8K store (015200)
X is 2 for 12K store (025200)
X is 3 for 16K store (035200)
:
X is 7 for 32K store (075200)

second card (seventh card from end of deck):

card	label		op'n	operands
01810Δ	PARΔΔ	Δ	+8ΔΔΔΔ	00bbbbbb00eeee00

00bbbbbb is the octal address where printing is to begin, usually 00000520 (see note).
00eeee00 is the octal address of the last row of memory to be printed. (for 8K this would be 00017700, for 12K - 00027700, etc.)

After altering the source deck, assemble the memory print to obtain the desired object card deck.

Operating Instructions:

1. Place the appropriate 2 card load routine ahead of the memory print object deck in the input hopper of the High Speed Reader.
2. Depress "LOAD CARD" - program start.
3. Depress "CONT" - program start.
4. To execute immediately, place printer "on line" and depress "Program Start".
5. To manually execute at any other time, set up octal address 015400 (8K) in M switches, depress "INST", "CLEAR", "DISPLAY", "CC", "M", and "Program Start".
6. To print all of memory as indicated on the PAR card (source deck) Sense Switch 1 should be set on, otherwise the print will start from the address in Tetrad 30 and end with the address in Tetrad 31. If limits are desired when operating the dump manually, Tetrads 30 and 31 must be set up from the console using the alteration procedure.
7. To operate the dump under program control, fix Tetrads 30 and 31 to the desired parameters and execute a JR to octal 015412 (8K). Control will be returned to the running program after the dump has been executed.

NOTE: Regardless of the parameters in Tetrads 30 and 31, location 0 through 0517g will always be printed in 4 character groups. The locations specified by the parameters are printed in 5 character groups. If Tetrad 30 contains 0, location 0 through 0517g will be repeated in 5 character format.

OPERATING INSTRUCTIONS FOR THE UNIVAC 1050 CARD SYSTEM

APPENDIX 3

OPERATING INSTRUCTIONS FOR 1050 CARD ASSEMBLER

CONSOLE SENSE SWTICHES.

1. Sense switch 1 controls punching of object deck.

   OFF - Object deck will be punched.
   ON - Object deck will not be punched.

2. Sense switch 2 controls printing of assembly listing.

   OFF - Listing will be printed.
   ON - No printing.

3. Sense switch 3 controls punching of label table.

   OFF - Label table will not be punched.
   ON - Label table will be punched.

PUNCH.

1. Turn punch on (depress power-on button until light is on).
2. Turn punch off-line (depress on-line switch until light is off).
3. Depress clear-manual feed card(s) until card appears in output stacker (remove card(s) from output stacker).
4. Turn punch on-line (depress on-line button).

PRINTER.

1. Depress off-line button until light is on.
2. Set paper 3 holes above sprocket.
3. Depress off-line button until light is off.

READER.

1. Place program to be assembled behind 1st pass assembler deck followed by about ½lb. blank cards.
2. Place cards in reader face down, 9 edge leading.
3. Depress power-on button until light is on.
4. Depress magazine-load button.
5. Depress clear button after magazine-load light is off.

CONSOLE.

1. Depress clear button.
2. Depress load card mode button.
3. Depress program-start button.
4. Depress con. mode button.
5. Depress program-start button (assemble 1st pass).

STOP (30 017777 60)$_8$.

READ UNIT.

1. Remove cards from output stacker, separating source code from 1st pass assembler deck.
2. Place source code behind 2nd pass assembler deck, followed by blank cards and weight, in the input magazine, 9 edge leading.
3. Depress magazine-load button.
4. Depress clear button after magazine-load light is off.

CONSOLE.

1. Depress program-start button (assemble 2nd pass).
   Program-Stop (30 017777 60)$_8$.
   End of Assembly.

OUTPUT.

Object code - output punch stacker.
Listing - printer.

ERROR STOPS

DISPLAY	PASS	DESCRIPTION	RECOVERY
30 017771 60	1 & 2	No BEGIN Card	Refeed source deck with valid BEGIN card. Depress program start to continue assembly.
30 010077 60	1	Label Table Exceeded	Depress program start to continue assembly. All labels that exceed the label will appear on the output listing with "L" errors. After assembly is finished use $ option to reduce number of labels.
30 110000 60	2	Reader Error	Refeed cards in error stacker. Depress clear on READER. Depress program start.
30 11000x 60	1	Reader Error	Replace 'X"cards in the input magazine. Depress clear on READER. Depress program start. NOTE: "X" does not always match number of cards in error stacker. Remove excess cards from normal stacker.
30 120002 60	Label Table Print & Punch	Punch Error a) No card in error stacker. b) 1 card in error stacker.	a) Remove last card in normal stacker. b) Last card in normal stacker will be followed by the next proper card. NOTE: All cards selected into error stacker may be discarded.

DISPLAY	PASS	DESCRIPTION	RECOVERY
Channel 2 Abnormal on Console will be lit.	2	Punch Error	Clean all cards out of punch. Turn punch off-line, manual feed cards until blank card appears in output stacker. Depress program start. Remove any blank cards from output stacker. NOTE: Read check is an unrecoverable error.
Channel 0 Abnormal on Console will be lit.	Label Table Punch & Print, 2	Printer Error	Turn printer off-line. Return printer to normal condition. Depress clear on PRINTER. Turn PRINTER on-line. Assembly will continue.
30 017777 60	1 & 2	Completion	Pass in operation is completed.

ADDITIONAL INSTRUCTIONS

SEND	Name of source field, Name of destination field, Number of characters to be transferred.

This operation causes a block transfer of up to 1024 characters.

Example:

SEND	IN, OUT, 80

The 80 characters of information beginning at IN is transferred to OUT and successively higher positions.

ALTER	Label of GOTO instruction, Name of operation

This operation permanently replaces the operand specified in the GOTO instruction with the name of a new operand.

X. REGENT OPERATING INSTRUCTIONS

A. Tape Regent

REGENT source cards are used as direct input to the PAL Tape Assembler, and the operating instructions for the assembler should be followed. No further processing of the output object programs is required.

B. Card Regent

The Card REGENT program produces a PAL source deck which is subsequently assembled using the PAL Card Assembler, after the desired I-O control routines are added. The steps required to produce an object program are as follows:

1. The Card REGENT object program is loaded from the card reader, followed by the source cards.

   The intermediate source output is punched and the input cards are listed on the printer. (This listing may be eliminated by setting Sense Switch 2.)

2. The intermediate output cards are removed from the punch. The I-O control routines required for the programs are selected from those supplied with Card REGENT. These are inserted after the first (BEGIN) card of the intermediate deck. The deck is then ready to be used as direct input to the PAL Card Assembler.

   Output cards will have been sequenced by Card REGENT beginning with 05000 (the I-O routines will have sequence numbers lower than 5000.) for the purpose of future reference or in case the need for sorting arises. Input statements will be punched in the output as comment cards containing a (.) in column 7, and blanks in the sequence number field. The original sequence number will appear in the I.D. field.

3. The operating instructions for the PAL Card Assembler should be followed. After both passes of the assembly have been completed, the I-O control cards should be removed and stored for future use.

CARD REGENT OBJECT-SPACE ESTIMATES (Approximate)

LOW ORDER MEMORY	400
USE	200
PAGE	250
INPUT	400
OUTPUT DTAIL	275
OUTPUT CARD	400
OUTPUT NONDT	150
READER CONTROL	650
PUNCH CONTROL	650
PRINTER CONTROL	650
READ	5
PUNCH	5
PRINT	10*
CLOS	*
ADD	*
SUB	*
MPY	50
DIV	80
ROLL	5*
RESET	*
ROUND	100
SHIFT	40
MOVE	50

SEND	20
CLEAR	20
IFDEC	*
IFALP	*
IFCHR	*
IFDIG	*
IFZON	80
IFNEG	*
LEV	130
RTN	5
EXIT	5
XCUTE	5
GOTO	5
STOP	5
ALTER	10

*Add 5 for each operand present.

UNIVAC 1050 OPERATING SYSTEM (OPS)

The operating system is divided into three major functions:

A. Input-Output Coordination, Program Switching
B. Tape I/O Order Handling
C. Program Loading and Memory Allocation

For the free standing system the Coordination and Tape Handler functions are substantially the same as for satellite systems. This description is primarily concerned with (1) the various parameters which may be used to assemble different versions of OPS for specific purposes and configurations; (2) description of loading and locating function; (3) program communication with OPS, and (4) console operating instructions.

I. Assembly Options.

OPS is available in source code in the standard library and may be assembled by the user to fit his needs. From 3 to 5 parameters may be written as follows:

OPS	p1	p2	p3	p4	p5
	Card Type,	Tape Type,	No. of Program,	Memory Dump,	Translation

Following is a description of each parameter:

A. Parameter 1, card type.

1. 80

   Provides for 80 column card loading from row or serial readers. Loader will accept 80 column "Call" cards for tape locating.

2. 90

   Provides reading of 90 column call cards from row or serial readers, with translation.

3. 90LS

   Same as (2) above but includes ability to load 90 column object cards from a serial reader.

4. 90LR

Same as (3) for row reader.

Note: One of the above must appear as parameter 1. If one of the 90 column parameters is used a translate table is generated for 90 column card code beginning in location 01500_8.

B. Parameter 2, Tape Type.

1. A

Provides tape order handling and error recovery for Uniservo IIIA tapes; up to 6 units.

2. C

Same as (1) for IIIC tapes.

C. Parameter 3, No. of Programs.

1. CONC

Provides for loading and running of 2 relocatable programs concurrently, or a single absolute program.

2. SING

Eliminates the portion of OPS which provide for concurrent processing (approx. 1000 char.).

D. Parameter 4, Memory Dump Option.

1. PDMP

Provides for inclusion of a memory print routine which can be executed by the operator. (Aprox. 1100 char.).

2. TDMP

Provides the ability to write all of memory on servo 1 for future printing. (65 char.).

Note: Parameter 4 is not required.

E. Parameter 5, Translation.

Since translate tables must be located in the first 4096 characters of memory, absolute locations must be set aside by the operating system for use by relocatable programs. The parameter TRNSn (where n is 1, 2, or 3) provides up to 3 open rows beginning in location 01500_8 into which translate tables may be transferred and used. In 90 column versions of OPS an input translate table is automatically generated in 01500_8 which may be used by the worker program, but not disturbed. In these cases, areas provided by the TRNSn parameter begin in 01600_8.

II. Program Loading Function.

All versions of OPS contain the ability to locate and load programs from a master instruction tape. Card loading ability is dependent on parameter 1, described above.

A. Program Call.

The Program ID may be provided by reading a "Call" card, or by trace switch settings (see Operating Instructions). A segment ID may be provided by these methods or by the worker program (see Section III). A call card must contain '$' in column 1 and the PID in columns 2-5. A blank PID from any source indicates that the next program is to be loaded from the card reader. A PID of (077777777) indicates that a load is not to be performed; the loader will release control to the Coordinator to continue a program already running, if there is one.

If a program ID other than blank or 00000 is received, the locator searches forward on servo 0 for a label block (R) containing a matching ID. If a match is not found the MIT is rewound and the system stops to await further instructions. The MIT is not rewound when a program has been located and loaded.

B. Program Memory Allocation.

Information in the R block (or card) enables the loader to determine whether or not the program will fit into available memory. In a concurrent system, the first relocatable program is assigned the lowest memory available, and the second is assigned the highest, except where the load key is 5, in which case it is always assigned the lowest.

The memory remains allocated until the program is released or jettisoned.

Absolute programs may be loaded only if all of the memory is available. If an absolute program has been loaded and not released or jettisoned, no other program may be loaded.

A list of the stop displays and procedures to be followed if the load being attempted is unacceptable is contained in Section IV, Operating Instructions.

C. Segment Loading.

Segments of either relocatable or absolute programs may be loaded from time to time using the methods described above, or a running program may access OPS for the purpose of loading its segment without operator intervention or knowledge. Relocatable segments are always assigned the same base address and memory allocation originally assigned the run.

III. Program Communication with OPS.

A. Class II Interrupt.

A program using decimal arithmetic instructions where the possibility of decimal overflow exists must load the address of the overflow routine into location 0775-0777. If a class II interrupt occurs which is not an operator request, control will be transferred to that address.

The actual class II interrupt entry channel must not be altered at any time.

B. Program Release.

When a program is completed, a JR to the Release Entry of OPS (0700) must be executed. OPS releases the memory allocated to the finished program and stops. Control is not thereafter returned to the program which has released.

The release entry should not be accessed unless all processing has been completed.

C. Program Stop.

In order to bring the computer to an orderly halt, all IO orders currently being executed must be completed and their interrupts processed. This is accomplished by a JR to the Stop routine (0736). OPS retains control until all pending IO interrupts have been processed.

The Stop routine must be accessed before executing a JD or JHJ instruction.

D. Segment Loading.

A running program may access OPS for the purpose of loading a segment by performing the following steps in order shown:

1. Execute the Stop Routine (JR 0736).

2. Set locations 0541 and 0542 to non-blank. This prevents OPS from stopping (0541) and reading a call card (0542). If it is intended that the segment ID be obtained from a call card, or trace setting, this step and following step 3 are not performed.

3. Store the Segment ID (4 characters) in AR2.

4. If the servo number on which the segment appears is other than 0, store the appropriate unit number in 0540.

5. Execute JR 0612. The segment will be located, loaded and executed. Locations 0540 thru 0542 will be reset to blank at completion of the load.

E. Trace Routine.

Information may be entered into memory from the operator console by using the console trace switches, when trace mode is set on PROC. The routine used by OPS for this purpose is a closed subroutine and is available to worker programs as follows:

1. Execute a display stop (JD) informing the operator of the need for a trace key in of 2 characters.

2. Execute JR 01245.

3. When control is returned, the two characters which have been set in the console trace switches will be in the least significant characters of AR2.

F. Translate Tables.

If translate table areas are included in the Operating System being used, they will begin in location 01500. Translate tables should be transferred into the areas as they are used, since they are not preserved when switching programs in concurrent operations. (See Section I, E).

G. Following is a summary of absolute locations in OPS which are available for program communication or information:

0540	Servo # of MIT
0541	OPS stop switch. (0 = stop, 1 = bypass stop 070001)
0542	OPS call card switch. (0 = read call card, 1 = bypass call card read)
0612	Load entry. Access by JR.
0700	Release entry. Access by JR.
0736	STOP entry. Access by JR.
01000-01244	Loader read image area.
01245	Trace switch routine. Access by JR.
01302-01355	Temporary storage of T16, T17, T18, T8, X1 through X7 Program A (low order). Concurrent systems only.
01356-01431	Same for Program B (high order).
01432-01435	PID of last program or segment loaded from tape.
01437-01441	Highest location of Program A. Do not alter.
01443-01445	Lowest Location of Program B. Do not alter.
01447-01451	Highest location in OPS + 1. Do not alter. (Lowest loc. PROG A).
01453-01455	Highest location in memory. Do not alter.
01500-01577	80 column system: 1st translate table area generated by parameter TRNS1. 90 column system: Input card code translate table. Subsequent translate tables generated by TRNS in parameter follow beginning in 01600.

IV. OPS Operating Instructions.

A. Initial Load.

The Master Instruction Tape is mounted on servo 0. Using the tape load facility, OPS is loaded and stopped (30 070001 60). Following is a description of the display stops which may be encountered, their causes and the action to be taken.

B. Display Stops.

070001 Ready to Load.

1. To load using call card, or preset PID, depress start. (Call card contains '$' in col. 1, PID in cols. 2-5.)

2. To load using trace switch setting; set first 2 characters in trace switches, depress Opr. Request, then Start. After stop 077000, set second 2 characters in trace switches, depress Start.

3. To return to program already running without performing load, set ID of '□□□□' (077777777) in trace switches, as in (2) above.

070002 No R card. Depress start to repeat load. Stop 070001 will be accessed. Correct input.

070003 Not enough memory available. Procedure same as 070002.

070004 Trying to load 3 programs. Procedure same as 070002.

070005 Card read is not call card. Procedure same as 070002.

070007 Program stopped by Operator Request. To attempt error recovery and/or continue running, depress start.

To exercise the following options at this time; depress Operator Request, set trace mode on PROC, and enter the appropriate key into the trace switches as follows: then depress Start.

077 Load program. Stop 070001 will be accessed.

076 Print memory.

075 Dump all of memory on servo 1 (see C below).

00 Jettison program using Fastrand.

01 Jettison program using Fastrand.

02 Jettison program using Servo 0.

03 Jettison program using Servo 1.

04 Jettison program using Servo 2.

05 Jettison program using Servo 3.

06 Jettison program using Servo 4.

07 Jettison program using Servo 5.

012 Jettison program using Reader.

013 Jettison program using Punch.

014 Jettison program using Printer. CH0

015 Jettison program using Printer. CH7

Note: Great care should be exercised in making the above settings. Incorrect key ins which are less than 017 may cause unrecoverable problems.

070010 Absolute program load is unacceptable (See Section IIB). Procedure same as 070002.

070104 Check sum error. Key 1 into loc. 0 to ignore. (Not recommended unless cause is positively known.) Otherwise, procedure same as 070002.

070105 Card or Block count error. Procedure same as 070104.

070106 Read Error during load. Depress clear, then Start. Loader will return to stop 070001.

Note: If a segment is being loaded from tape, the loader may attempt to restart the load while the MIT is rewinding in which case the error stop will be repeated. Wait for rewind to be completed, then repeat procedure.

070707 PID not found on tape. MIT rewinding. Wait for rewind, then depress start to try again.

077000 First half of trace switch ID acknowledged. Key in other half, depress start.

070013 Card load being attempted in 90 column system. This version of OPS does not contain a card loader.

077776 Tape abnormal during tape memory dump. Rewind Servo 1 and depress start to try again.

077777 Tape memory dump completed. Depress start to access stop 070007. Program (S) may be continued from this point.

C. Tape Memory Dump.

If the OPS in use contains the tape memory dump feature, it may be accessed by a trace switch key in of 075 at the appropriate time, following an operator request stop (070007). A blank tape should be on Servo 1 at load point. All of memory is written in maximum size blocks and the computer is stopped (077777). The tape is not rewound. Depress Start to return to normal operation. The standard library contains the routine TDMP which may be loaded using the normal call procedure for the purpose of printing the memory as written on Servo 1. The printout obtained is in the same format as the Print Dump routine.

D. Print Memory Dump.

The Print Memory, if provided in the system, produces an octal printout directly from memory and returns to stop 070007 when completed. It is accessed by a trace key in of 076 following the operator request stop. In view of its high memory requirement, the use of the print dump option is not recommended except for the initial stages of debugging.

E. Peripheral Error Recovery.

Each of the IO control routines used with OPS contains its own error display stops indicating the nature of the error which has occurred and identifying the channel unit, etc. Following is the procedure to be followed to attempt error recovery following a peripheral error stop. (Note exception tape single program).

1. Depress Start.

   If a second program is in memory and is not affected by the error, it will continue processing. If there is no second program OPS will loop until the operator intervenes. The program in which the error occurred will be by-passed until the condition is corrected.

2. Correct the error condition, if possible.

3. Depress Operator Request. The operator request stop (070007) will be accessed.

4. Depress Start.

   If the error condition has been properly corrected, the program (S) will be resumed from the point of error. If not, the error stop will reappear, and the procedure must be repeated.

   It should be noted here, that a retry will be attempted on each peripheral which has an error condition existing at this time whether or not an attempt has been made to correct it.

   Exception: If the single program version of OPS is being used, the above procedure does not apply to tape errors. If a tape error occurs, recovery is attempted immediately, when Start is depressed after the error display stop. Control remains in the tape error recovery routine until the situation is corrected.

F. Program Jettison (Concurrent Processing Only)

The jettison procedure enables the operator to release the memory allocation of a program which is unable to continue; and to replace it with another program and/or continue a program which has been running concurrently.

Normally, this will be done when an unrecoverable peripheral error has occurred and the program is unable to proceed to its normal conclusion and release.

Following is the procedure to be followed:

1. Depress operator request to access stop 070007.

2. While the computer is stopped, set the jettison code (see above, Section B) of any <u>one</u> of the peripherals being used by the program being jettisoned into the trace switches.

3. Set trace mode to PROC.

4. Depress Operator Request.

5. Depress Start. The program will be released and Stop 070001 will appear, and any of the options listed in Section B may be exercised.

There is not a jettison procedure in the single program versions of OPS. If a program cannot run to normal completion, OPS must be reloaded in order to substitute another program.

G. Operator Request.

The varied reasons for using the Operator Request button are described above. Following are some general remarks regarding its operation:

1. The button must be lit when it is depressed in order to take effect. There are times when, during the running of OPS and the peripheral control routines, it must be inhibited (light out). Sometimes this is obvious when looking at the console, but usually the inhibit periods are so brief that the light seems to be lit continuously or may be flickering. If depressing the button has no effect, it was probably inhibited at the instant it was depressed. Hesitate, then try again.

2. If a program contains an error which causes it to enter a loop which does not involve IO processing, OPS will never be able to secure control in order to process an Operator Request. This condition will usually be apparent in that the processor will be running (looping) and no I.O peripherals will be running. In order to obtain a memory dump and jettison the program:

a. Depress Program Stop.

b. Set top row of console switches to 1000072700_8

c. Set Display-Alter section to INST.

d. Depress ONE INST

e. Depress CLEAR

f. Depress ALTER

g. Depress START

h. Depress CONT

i. Depress START

If the computer does not now stop at display 070007, depress Operator Request. Follow memory dump and/or jettison procedure. If a second program had been running before the problem developed, it may now be continued.

3. When programs are running normally with OPS, operator request is the only safe way to stop the computer without risking the loss of IO images. Use of the Program Stop button is not recommended.

Assembly Instructions.

OPS source code is a part of the standard library and may be assembled using the parameters in Section I, by preparing three cards as follows:

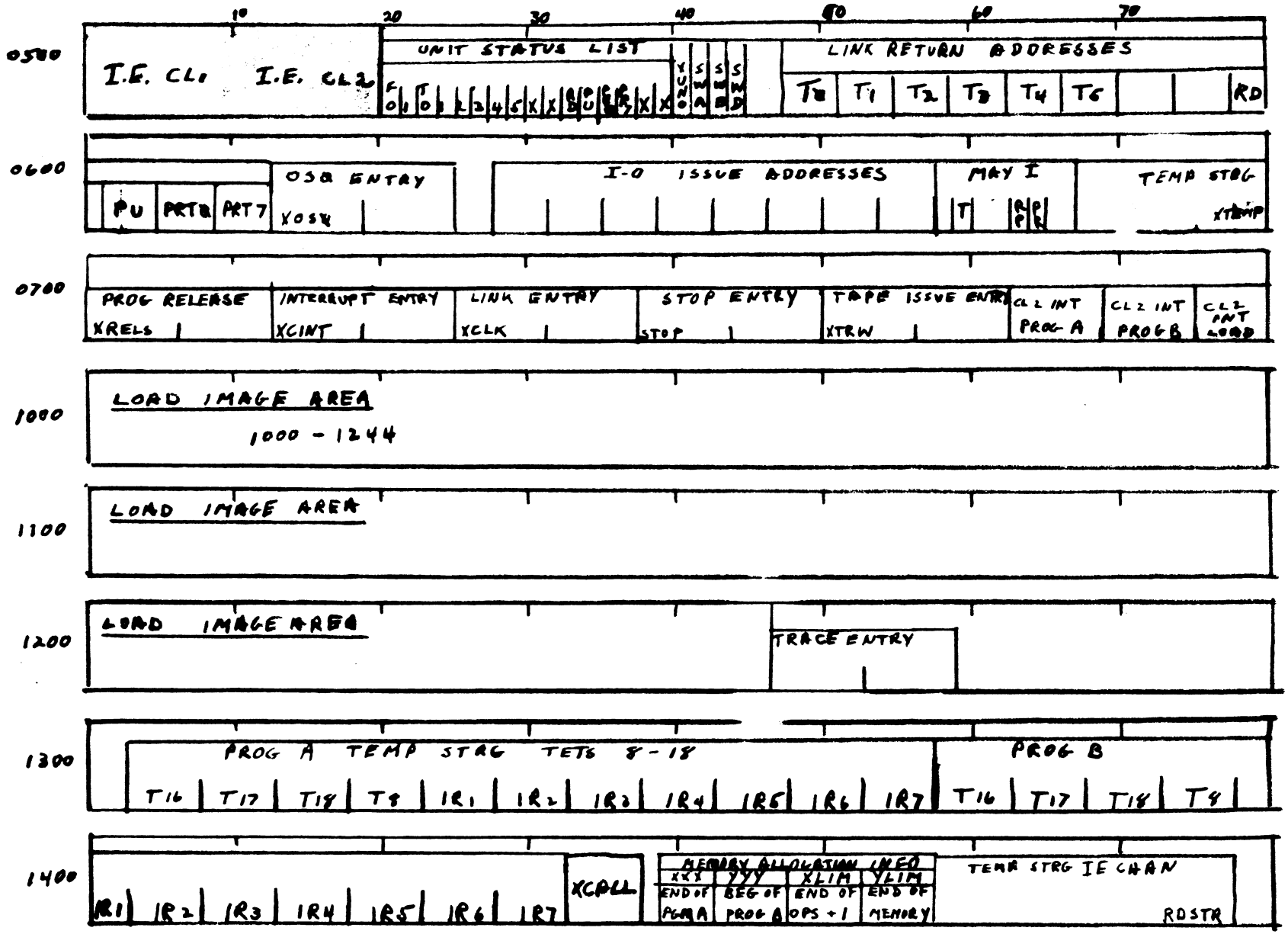
```
7          13          19
PID        BEGIN       0520

           OPS         p1, p2, p3, p4, p5

           END         START
```

Any 4 character PID may be used in the BEGIN card. However, Systems Programming has assigned names to 10 versions for future reference accordingly. They are:

Name	Parameter Combination
0S01 *	80, A, SING, TDMP
0S02*	80, A, CONC, TDMP
0S03*	90, A, SING, TDMP, TRNS1
0S04	90, A, CONC, TDMP, TRNS1
0S05	80, A, SING, PDMP
0S06	90, A, SING, PDMP, TRNS1
0S07	80, C, SING, TDMP
0S08	80, C, CONC, TDMP
0S09	90, A, CONC, PDMP, TRNS1
0S10	80, A, CONC, PDMP

* These three versions are in the standard system tape.

OPS PERMANENT STORAGE LAYOUT



I/O SPECIALIZER FOR UNIVAC 1050 CARD SYSTEM

The UNIVAC 1050 Card System I/O Specializer produces source decks (for PAL) of the reader, punch and printer routines as specified in directive cards. The I/O routines produced by the Specializer do not require the 1050 coordination function.

The directive card is in the following form. The label field is left blank. The "Operation field has RDR written for the reader routine, PCH" for the punch and "PRNT for the printer. The Operands" field contains a series of parameters separated by commas. The number, nature and interpretation of these expressions is determined by the particular routine being specified as follows:

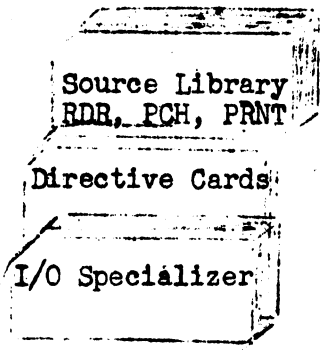
Routine / Parameter	RDR	PCH	PRNT
P1	LABEL of AREA associated with routine.		
P2	Number of reserve areas associated with routine. $3 < P2 < 32$	$3 < P2 < 32$	$2 < P2 < 32$
P3	Index register to contain the relative area address.		
P4	TRNSL(△) or UNTRN	TRNSL(△) or UNTRN	FULL (△) or HALF

Example: A reader routine is desired to read cards with the translated card images read into one of three reserve areas. The label of the first character of the area is CRDIN. The area relative address of the current card image area is to be supplied in index register 5. The directive card will contain the following information.

LABEL	OPERATION	OPERANDS
(blank)	RDR	CRDIN,3,5

The deck produced by the Specializer will contain the proper PAL coding for the reader routine requested for assembly with the worker program source deck.

Card input for Specialization is in three parts.



The first part is the I/O Specializer program deck. This is followed by the directive cards to specify any one or all of the I/O routines. The third part is the source library deck containing the reader, punch and printer routines. A blank card must follow the third part.

When loading of the first part (Specializer) is completed the computer will stop with the display 30 01 0000 60. Before hitting "Program Start" the operator may select any of the following options.

Depress Sense Switch 1 - No process of Specialization - punches and prints part three input cards.
Depress Sense Switch 2 - No punching.
Depress Sense Switch 3 - No printing.

These sense switch options may be combined if required.

Successful Completion - 30 010077 60

Error Stop - 30 01 0001 60 - Incorrect expression on directive card.

I/O ROUTINES FOR UNIVAC 1050 TAPE SYSTEM

The card reader, punch and printer routines for the tape system are essentially the same as those for the card system. Worker program communication is exactly as stated for the card system. The difference between the card and tape systems lies in the communication that takes place between these routines and the coordination function.

Error recovery is effected by the operator through the coordinator operator request. (See Coordinator Operating Instructions.)

For use of the I/O Library, specifications are included in the worker program in the following form for assembly. The label field is left blank. 'RDR' is written in the 'Operation' field for the reader routine, PCH for the punch and PRNT for the printer. The 'Operands' field contains a series of parameters separated by commas. The number, nature and interpretation of these expressions is determined by the particular routine being specified, as follows:

In a two printer system the call 'PRNT7' is available for a channel 7. print routine. The parameters are identical as those for the first printer routine.

Routine / Parameter	RDR	PCH	PRNT
P1	LABEL of AREA associated with routine.		
P2	Number of reserve areas associated with routine. 3 < P2 < 32	3 < P2 < 32	2 < P2 < 32
P3	Index register to contain the relative area address		
P4	TRNSL(Δ) or UNTRN	TRNSL(Δ) or UTRN	FULL(Δ) or HALF

Example: A reader routine is desired to read cards with the translated card images read into one of three reserve areas. The label of the first character of the area is 'CRDIN'. The area relative address of the current card image area is to be supplied in index register 5. The worker program will contain the following line of coding.

LABEL	OPERATION	OPERANDS
(blank)	RDR	CRDIN, 3, 5

The object program will contain the proper coding for the reader routine requested.

CARD PROC PARAMETERS

Routine / Parameter	RDR, RDR9, RDS RDS9, *REA *REA9 *RES *RES9	PCH, PCH9, PCHS9 *PUN, *PUN9, *PUNS9	PRNT, PRNT7, PRPL PRPL7, *PRT, *PRPL
P1	LABEL NAME OF AREA ASSOCIATED WITH ROUTINE		
P2	NUMBER OF RESERVE AREAS 3 to 21	3 to 21	2 to 21
P3	INDEX REGISTER TO CONTAIN RELATIVE AREA ADDRESS		
P4	TRNSL (Δ) or UNTRN	TRNSL (Δ) or UNTRN	Full (Δ), Half or 132 (buffered only)

TAPE SYSTEM 1050

I/O CARD PROCS

I/O CALL LINE	COL	ROUTINE	TOTAL SIZE INCLUDING MIN. NO. OF AREAS	MINIMUM NO. OF AREAS
RDR	80	Row Reader Routine - 80	989	3
RDR9	90	Row Reader Routine - 90	989	3
RDS	80	Serial Reader Routine - 80	637	2
RDS9	90	Serial Reader Routine - 90	672	2
PCH	80	Row Punch Routine - 80	928	3
PCH9	90	Row Punch Routine - 90	928	3
PCHS9	90	Row Punch(Serial)Routine - 90	968	3
PRNT	---	Channel Ø PRINT	908	2
PRNT7	---	Channel 7 PRINT	908	2
PRPL	---	Channel Ø W/Paper Low ability	983	2
PRPL7	---	Channel 7 W/Paper Low ability	983	2

CARD SYSTEM

*REA	80	Row Reader - 80	794	3
*REA9	90	Row Reader - 90	794	3
*RES	80	Serial Reader - 80	546	2
*RES9	90	Serial Reader - 90	576	2
*PUN	80	Row Punch - 80	748	3
*PUN9	90	Row Punch - 90	748	3
*PUNS9	90	Row Punch(Serial) - 90	788	3
*PRT	---	Channel Ø PRINT	763	2
*PRPL	---	Channel Ø W/Paper Low ability	823	2

1. ROW READER ROUTINE.

1.1 Subroutines. Referenced by Worker Program.

Initialize (XINRD). XINRD must be entered before there is any attempt to get a card image. Base address tetrad (tetrad 36), standby address tetrad (tetrad 37) and the channel interrupt entry are set to their appropriate values. All the indicators, counters and variable connectors are reset to their initial conditions. No feed card order is issued. Re-initialization takes place automatically on error recovery.

Execute (XCTRD). XCTRD must be entered when the worker program wants a new card image. The present reserve area is assumed to be released by the worker program. Thus RAC is increased by 1. A feed card instruction will be issued if there is no card in the track. The base address of the next reserve area available to the worker program is given in the IR specified by p3. If an error condition exists and there is neither a card image ready for processing nor an actual card in the track, the computer will be brought to an orderly stop with the following in the instruction register:

30 11000X 60 where X equals 1 ot 2,

the number of cards to be reloaded. When no images are available to the worker program, this routine will set the 8 and 4 bits of the unit status list and transfer control to the coordinator. Control will be returned to the worker program (when an image becomes available) at XB22, the link entry of XCTRD.

1.2 Subroutines Internally Referenced.

Feed Card (XBFCD). XBFCD is entered from the coordinator, interrupt or the execute subroutine when a feed card instruction can be issued. This section must not be entered directly from the worker program. A feed card instruction will be issued if there is no error condition in the reader, punch or printer and if its MAY I switch is on. This routine will immediately exit if operating under an error condition. It also turns on the 2-bit (operating bit) of the unit status list if a feed card instruction is executed.

Interrupt (XBITR). XBITR is automatically entered at each cycle point 18 or when a feed card instruction is issued while an error condition exists. In case of an error interrupt, the error subroutine is entered setting the 1-bit (error) of the unit status list. (Control is returned to the worker program without stopping the computer after an appropriate treatment for the error situation.) In case of a normal interrupt, the 2nd bit of PHI is tested against 1 and if equal, RAC is decreased by 1. XBFCD will be entered, if possible*. Further cyclical interrupts will be inhibited if there is no card in the track and if there is no error condition. If and

*The possibility is determined by testing RAC against PHI.

only if RAC > PHI, there is at least one reserve area available to the card reader, the 1 and 4 bits at the unit status list are reset after a successful read. The tape MAY I switch is turned on if the punch is not operating. If there are no cards in the track during interrupt, the 2-bit (operating bit) is also turned off. Coordinator interrupt is executed during the routine.

Error (XLERR). XLERR is entered from the interrupt subroutine when an error condition is detected. The correct number of error cards is inserted in the error card counter which is the 3rd l.s.d. of the reader error stop instruction. The 1-bit (error) of the unit status list is set. Feed card (XLFCD) is set to exit. The error switch in XCTRD is set to stop when all remaining good images are exhausted.

1.3 Indicators and Counters.

Phase Indicator (PHI). PHI consists of the two least significant bits of a character to which the tag XLPHI is assigned. A 1-bit is inserted in the l.s.b. of PHI when a feed card instruction is issued. At each cyclical interrupt, successful or not, PHI is shifted left one bit position and a 0-bit is inserted in the l.s.b. if there is no reserve area released by the worker program. The number of 1-bits in PHI shows the number of cards in the track. This number together with the number of reserve areas filled with data, not yet processed or under processing, determines the number of reserve areas unavailable to the card reader. At each successful interrupt, the 2nd l.s.b. of PHI is tested against 1 and if equal, a card image was read into memory during the previous card cycle.

Read Area Indicator (RAI). RAI consists of a character to which the tag XLRAI is assigned. RAI is initially set to the total number of reserve areas. Each time the feed card subroutine is entered, RAI is decreased by 1. A feed card instruction without memory advance is issued. In the latter case, RAI is reset to the initial value.

Workable Area Indicator (WAI). WAI consists of a character to which the tag XLWAI is assigned. WAI is initially set to 1. Prior to giving the base address of the next reserve area ready for processing to the worker program, WAI is decreased by 1 and tested against 0. If equal, the base address is reset to the initial value and WAI is set to the total number of reserve areas. Otherwise, the base address is advanced by 128 or 192.

Readable Area Counter (RAC), RAC consists of a character to which the tag XLRAC is assigned. RAC is initially set to the total number of reserve areas minus 1. RAC is increased by 1 each time the execute subroutine is entered and is decreased by 1 each time a card image is successfully read into memory. Prior to giving the base address to the worker program, a test is made to determine if RAC is equal to the total number of reserve areas. If equal, the execute subroutine will wait until a card image is read into memory. Otherwise the next working area is available to the worker program.

1A. Serial Reader Routine

In the serial read routine the communication with the worker program is exactly the same as it is with the Row Reader.

All the information stated concerning the Row Reader, Its subroutines and construction apply directly to the Serial Reader with the following exception:

The Phase Indicator (PHI) is non-existent in the serial reader routine. Only one card at a time may be in the track. Therefore, the error stop will contain a reload at Ø or 1 card as compared to 1 or 2 cards in the row reader routine.

There is no cyclical interrupt with the serial read routine. Card feeds are issued during interrupt approximately 0.5 ms after the interrupt is received.

The routines will recover from all errors with the exception of an output jam.

FOR ROUTINES *REA, RDR, *REA9, RDR9,

*RES, RDS, *RES9, RDS9

ROW & SERIAL

U 1050 READER

REASON FOR STOP	RESULTING CONDITION	RECOVERY PROCEDURE
STKR full Hopper empty Registration Marginal check	Recoverable	Clear problem at reader, reload number of cards as indicated by stop display even if this does not agree with the number in the error stacker. Reload hopper, depress ready and start buttons.
All others	Non-recoverable	Any error that causes the reader drive motor to be stopped is non-recoverable.

NOTE: The recovery procedure for the tape system using the coordinator is exactly the same as stated above with one exception. That being that after start button is depressed, the operator request button must be depressed to signal the coordinator that an error recovery attempt is being made. Depressing the start button after this will cause the program to attempt recovery.

2. PUNCH ROUTINE.

2.1 Subroutines. Referenced by the Worker Program.

Initialize (XINPH). XINPH must be entered before there is any attempt to edit data to be punched. All the reserve areas are cleared to spaces. Those areas between two punch areas are not altered. The channel interrupt entry is set to its appropriate value. All the indicators, counters and variable connectors are reset to their initial conditions. The base address of the first working area is given to IR2. Issue and link addresses are stored in coordinator.

Execute (XCTPH). XCTPH must be entered when the worker program finished the editing of data and wants it to be punched. PAC is increased by 1. A punch instruction will be issued if the previous one has been completed. The base address of the next reserve area available to the worker program is given to IR specified by p_3. If none exists the 8 and 4 bits of the unit status list are set and control is transferred to the coordinator. When an area becomes available, control is returned to the worker program through the link entry XC22.

Close Out (XCLPH). XCLPH must be entered when the worker program wants all the remaining images to be punched and the punch unit to be cleared of data cards. After all the data cards are punched, a feed instruction is issued to send the last valid card into the output stacker.

2.2 Subroutines Internally Referenced.

Error (XCERR). XCERR is entered from the interrupt subroutine when an error condition is detected. The computer is brought to an orderly stop with the following in the instruction register:

30 120000 60

When recovery is attempted through the coordinator, the follow-up punches for the two last cards will be done. One or two cards will be selected into the error stacker. The 1-bit (error) is set in the unit status list. The recovery switch is set in XCPCH. The coordinator stop routine is executed and after the stop display when the run depressed control is transferred to the coordinator.

Punch (XCPCH). XCPCH is entered from the coordinator, interrupt or the execute subroutine when a punch instruction can be issued. The punch instruction is always issued without memory advance. The base address (tetrad 40) is updated (advanced or reset) prior to issuing the punch instruction. This section must not be entered directly from the worker program. Error recovery is effected through this routine. The tape MAY I is turned off upon the issuance of a punch order. The 2-bit (operating bit) is turned on.

Interrupt (XCITR). XCITR is automatically entered when a punch instruction is completed successfully or not. In case of a successful interrupt, the contents of tetrad 40 is stored to a temporary storage so that it can be used for a follow-up punch when an error condition is caused by the next punch instruction. PAC is decreased by 1 and tested against 0. If PAC is not 0, XCPCH will be entered. In case of an error interrupt, the error subroutine is entered and the computer is brought to an orderly stop. The unit status list 2 and 4 bits of the punch are turned off upon successful interrupt. If the reader is not operating the tape MAY I is turned on. The coordinator interrupt entry is executed to enable other I/O units a chance to operate.

2.3 Indicators and Counters.

Punch Area Indicator (PAI). PAI consists of a character to which the tag XCPAI is assigned. PAI is initially set to 1. Each time the punch subroutine is entered. PAI is decreased by 1 and tested against 0. If equal, the base address of the reserve area next to be punched (tetrad 40) is reset to the initial value and PAI is set to the total number of reserve areas. Otherwise, the base address is advanced by 128 or 192.

Workable Area Indicator (WAI). WAI consists of a character to which the tag XCWAI is assigned. WAI is initially set to the total number of reserve areas. Prior to giving the base address of the next reserve area ready for processing to the worker program, WAI is decreased by 1 and tested against 0. If equal, the base address and WAI are reset to their initial values.

Punchable Area Counter (PAC). PAC consists of character to which the tag XCPAC is assigned. PAC is initially set to 0. PAC is increased by 1 each time the execute subroutine is entered and decreased by 1 each time a card image is successfully punched (but not yet check read). Prior to giving the base address to the worker program, a test is made to determine if PAC is larger than the total number of reserve areas minus 2. If larger, the execute subroutine will wait until a punch instruction is completed and a reserve area is released to the worker program. Otherwise, the next working area is available to the worker program.

FOR ROUTINES PCH, PCH9, PCHS9, *PUN, *PUN9, *PUNS9

REASON FOR STOP	INTERNAL INDICATOR	PUNCH PANEL LIGHT	RESULTING CONDITION	RECOVERY PROCEDURE	No. of cards that should be in error STKR at stop
Read check	Hole Ct error	Read check	Recoverable	Depress ready and start buttons.	
Stacker full	Non-ready	STKR full	Recoverable	Depress ready and start after emptying stacker	Ø
Hopper empty	Non-ready	Hopper empty	Recoverable	Load hopper with cards depress ready and start	Ø
Off-line	Non-ready	Off-line	Recoverable	Depress off-line, ready and start buttons	Ø - initially 1 - if it occurs while punching
All others	Non-ready	SKEW A & B ENTRY A & B EXIT A & B JAM POWER LOSS	*Non-recoverable (See Below)	_____	1 or Ø

*It is possible to recover from these errors at the risk of duplicating or losing a maximum of 2 images depending upon conditions. However, the recovery attempt will be successful in most cases. (No images lost of duplicated) for jam type errors (i.e., SKEW, ENTRY, EXIT and JAM panel lights). The punch track must be cleared and blank cards manually fed through all stations. After this is done depress ready and start. A read check will occur and the procedure for recovering from read-checks should then be followed. For other than jam type errors follow read-check procedure.

NOTE: The recovery procedure for the tape system using the coordinator is exactly the same as stated above with one exception. That being that after start button is depressed the operator REQUEST button must be depressed to signal the coordinator that an error recovery attempt is being made. Depressing the start button after this will cause the program to attempt recovery.

3. PRINTER ROUTINE.

3.1 Subroutines. Referenced by Worker Program.

Initialize (XINPR or (XINP7). XINPR must be entered before there is any attempt to edit data to be printed. All the reserve areas are cleared to spaces. The channel interrupt entry is set to its appropriate value. All the indicators, counters and variable connectors are reset to their initial conditions. The base address of the first working area is given to IR3. The link and issue addresses are stored in the coordinator.

Execute (XCTPR) or (XCTP7). XCTPR must be entered when the worker program finished the editing of data and wants it to be printed. XADVC must be supplied with the number of lines to be advanced by the worker program before entering XCTPR. (XADVC) is transferred to one of the temporary storages. PAC is increased by 1. A print instruction will be issued if the previous one has been completed. The base address of the next reserve area available to the worker program is given in the IR specified by p_3, if one exists, otherwise the 4 and 8 bits of the unit status list are set and control is transferred to the coordinator link. When an image area becomes available control will be transferred to the worker program from the coordinator through the printer link entry XA22.

Close Out (XCLPR) or (XCLP7). XCLPR must be entered when the worker program wants all the remaining images to be printed. This section is entered each time the remote print or the advance paper subroutine is entered.

3.2 Subroutines Internally Referenced.

Print (XAPRT) or (XAPR7). XAPRT is entered from the coordinator, interrupt or the execute subroutine when a print instruction can be issued. The print instruction is always issued without memory advance. Prior to issuing the print instruction the base address (tetrad 32) is updated (advanced or reset) and the line advance count (tetrad 33) is supplied with the appropriate number of lines to be advanced, if MAY I is off no print instruction is issued. Error recovery is effected through this routine. This section must not be entered directly from the worker program. The 2-bit (operating bit) is turned on when XF is issued.

Interrupt (XAITR). XAITR is automatically entered when a print instruction is completed successfully or not. In case of a successful interrupt, PAC is decreased by 1 and tested against 0. If PAC is not 0, XAPRT will be entered. In case of an error interrupt, the error subroutine is entered and the computer is brought to an orderly stop. The coordinator interrupt routine is executed during this time to enable other peripherals to operate. The 2 and 4 bits of the unit status list are reset upon successful interrupt.

Advance Paper (XCTAD) or (XCTA7). XCTAD must be entered when the worker program wants the paper to be advanced the number of lines specified. This number must be supplied to XADVC before entering XCTAD. XCLPR is executed before issuing the advance paper instruction.

Call (XCTOL) or (XCTO7). XCTOL must be entered when the worker program wants a 'remote area' to be printed. A 'remote area' means a print area which is not included in the reserve areas. The worker program can place any number of 'remote areas' anywhere he wants as far as the memory capacity permits. XCTOL can be used to print such things as heading lines, page numbers and so on from these 'remote areas' saving the worker program the trouble of transferring constants to reserve areas. Prior to entering XCTOL, the owrker program must supply the location XRMAR (3 characters) with the base address of the remote area and XADVC with the number of lines to be advanced. The size of a remote area must be the same as for a reserve area, that is, 128 characters in the full line mode or 64 characters in the half line mode. XCLPR is executed before issuing the print instruction.

Error (XAERR). XAERR is entered from the interrupt subroutine when an error condition is detected. Unit status list 1-bit is set. The computer is brought to an orderly stop with the following in the instruction register, after the XCCRD stop routine is executed:

30100000 60

When recovery is attempted through the coordinator, the previous print or paper advance instruction will be reissued. Then control is returned to the worker program.

NOTE: Communication between the worker program and the print routine is effected as previously noted through JR's to various tags. In the case of the channel 7 print routine, the tags involved in the JR's have the least significant digit changed to seven (7).

3.3 Indicators and Counters.

Print Area Indicator (PAI). PAI consists of a character to which the tag XAPAI is assigned. PAI is initially set to 1. Each time the print subroutine is entered, PAI is decreased by 1 and tested against 0. If equal, the base address of the next reserve area to be printed (tetrad32) is reset to the initial value and PAI is set to the total number of reserve areas. Otherwise the base address is advanced by 64 or 128.

Workable Area Indicator (WAI). WAI consists of a character to which the tag XAWAI is assigned. WAI is initially set to the total number of reserve areas. Prior to giving the base address of the next reserve area ready for processing to the worker program, WAI is decreased by 1 and tested against 0. If equal, the base address and WAI are reset to their initial values.

Printable Area Counter (PAC). PAC consists of a character to which the tag XAPAC is assigned. PAC is initially set to 0. PAC is increases by 1 each time the execute subroutine is entered and decreased by 1 each time a line is successfully printed. Prior to giving the base address to the worker program, a test is made to determine if PAC is equal to the total number of reserve areas. If equal, the execute subroutine will wait until a print instruction is completed and a reserve area is released to the worker program. Otherwise the next working area is available to the worker program.

Storage Inidcators (XADVC). There are two 3-character indicators which control the transferring the number of lines to be advanced (XADVC) to temporary storages and to tetrad 33. One of them appears in the execute subroutine and is used to store (XADVC) to one of the temporary storages. The other appears in the print subroutine and is used to bring the contents of the proper temporary storage into tetrad 33.

FOR ROUTINES PRNT, PRNT7, PRPL, PRPL7, *PRT, *PRPL

U 1050 PRINTER "CARD SYSTEM"

REASON FOR STOP	INTERNAL INDICATOR	PRINTER PANEL LIGHT	RESULTING CONDITION	RECOVERY PROCEDURE
Off-Line	Non-ready	Off-line	Recoverable	Depress off-line, ready and start buttons
Carriage out	Non-ready	Carriage out	Recoverable	Depress carriage in until carriage is completely in, then ready and start buttons.
Ribbon out	Non-ready	Ribbon out	Recoverable	Call technician. When ribbon restored depress ready and start buttons.
Paper low If paper low option has been called in the print PROC.	Paper low	Forms out	Recoverable	Depress manual print button and start. This will cause one line to be printed. Continue this procedure until all printing is finished for that page or until line advance between pages is executed, then reload new paper stock. Depress manual print and start, and the program will continue.
All others	Non-ready	Overheat D.C. fault forms runaway etc.	Non-recoverable	If recovery attempt is desired at risk of a lost or duplicated line, clear problem, depress ready and start buttons.

NOTE: The recovery procedure for the tape system using the coordinator is exactly the same as stated above with one exception. That being that after start button is depressed the operator REQUEST button must be depressed to signal the coordinator that an error recovery attempt is being made. Depressing the start button after this will cause the program to attempt recovery.

PAPER LOW MANUAL PRINT OPTION FOR ROUTINES PRNT, PRNT7, PRPL, PRPL7, *PRT, *PRPL

A. This option allows the worker program at the additional cost of 80 characters to print to end of a page (i.e., fixed forms, etc.,) when the printer stops in a paper low condition.

The stops are different from normal error stops and are as follows:

Channel Ø Printer (PRNT) 30 010700 60
Channel 7 Printer (PRNT7) 30 017700 60

To include this option in the print routine change the PROC call line as follows:

1. USE PRPL in place of PRNT
2. USE PRPL7 in place of PRNT7

B. The following procedure must be used when the paper low stop is reached:

Depress the manual print button, then the start button. This will cause one line to be printed and the program will return to the paper low stop. Continue this procedure until the last line for that page is printed or until page advance is executed (paper now past hammers). At this point reload paper stock depress the manual print and start buttons and the program will continue.

UNIVAC 1050

SYSTEM AND LIBRARY TAPE CONVENTIONS

TABLE OF CONTENTS

Note: 3C tapes will be in compatible mode at 556 BPI.

SYSTEM TAPE CONVENTIONS

1.0 LOAD BLOCK (165 Characters in Length)

The first two (2) blocks of the master tape are tape load blocks occupying memory location 17000_8 to 17600_8.

Position 0	L (Block Type)
Position 1-3	Binary block count
Position 4-25	Not used
Position 26-29	Starting address
Position 30-31	Numbers of characters to be loaded this block
Position 32-143	Data to be loaded
Position 144-164	Not used

These blocks are generated only by the AJAX tape utility routine as the first two blocks on tape (see Figure 1) when called for by parameter 3 of the Inout card of AJAX.

2.0 PROGRAM HEADER BLOCK (165 Characters in Length)

Position 0	♀ (053) Block Type
Position 1-3	Block number (binary)
Position 4-7	Run ID
Position 8-164	Not used

This block must precede all object programs on tape. It is automatically generated by the PAL Freestanding Tape Assembler.

3.0 'R' BLOCK (165 Characters in Length)

Position 0	R(054) Block Type
Position 1-3	Block number (binary)
Position 4-5	Not used
Position 6-9	Relative address of assembly
Position 10-12	Total number of locations assigned to program
Position 13-15	Number 1 higher than highest location into which information will be loaded
Position 16-135	Not used
Position 136-139	077777777
Position 140-143	Run ID
Position 144-145	Load Key
Position 146-147	02001 (R in col. 74)
Position 148-164	Not used

4.0 'S' BLOCK (165 Characters in Length)

Position 0	S (065) Block Type
Position 1-3	Block number (binary)
Position 4-135	Not used
Position 136-139	077777777
Position 140-143	Segment ID
Position 144-145	Load Key
Position 146-147	01200 (S in col. 74)
Position 148-164	Not used

Segment ID is Run ID (or last segment ID) plus decimal 01.

5.0 'INSTRUCTION' BLOCK (165 Character Block in Length)

Position 0	W (071) Block Type
Position 1-3	Block number (binary)
Position 4-5	Not used
Position 6-25	Relocation mask
Position 26-29	Starting address
Position 30-31	Number of characters to be loaded from this block
Position 32-143	Data to be loaded
Position 144-145	Check sum
Position 146-147	Blank
Position 148-164	Not used

6.0 'T' BLOCK (165 Characters in Length)

Position 0	T (066) Block Type
Position 1-3	Block number (binary)
Position 32-36	Not used
Position 32-36	Jump instruction to start of program
Position 37-38	Program (excluding routine header) or segment card count
Position 39-143	Not used
Position 144-145	Check sum
Position 146-147	01100 (T in col. 74)
Position 148-164	Not used

7.0 'LDKEY' BLOCK (165 Character Block in Length)

Position 0	Any allowable key supplied by the use of the 'LDKEY' assembler directive
Position 1-3	Block number (binary)
Position 4-5	Not used
Position 6-25	Relocation mask
Position 26-29	Starting address
Position 30-31	Number of characters to be loaded from this block
Position 32-143	Data to be loaded
Position 144-145	Check sum
Position 146-147	Blank
Position 148-164	Not used

8.0 PROGRAM SENTINEL BLOCK (165 Characters in Length)

Position 0	Y (073) Block Type
Position 1-3	Block number (binary)
Position 4-164	Not used

This block must follow all object programs on tape. It is automatically generated by the PAL tape assembler.

9.0 TAPE SENTINEL BLOCK (165 Characters in Length)

Position 0	Z (074) Block Type
Position 1-3	Block number (binary)
Position 4-164	Not used

This block is the last block on tape. There are always two (2) present.

10.0 SOURCE PROGRAM or PROC HEADER BLOCK (165 Characters in Length)

Position 0	D (027) Block Type
Position 1-3	Block number (binary)
Position 4-7	Run ID
Position 8-164	Not used

11.0 SOURCE PROGRAM or PROC SENTINEL BLOCK (165 Characters in Length)

Position 0	F (031) Block Type
Position 1-3	Block number (binary)
Position 4-164	Not used

11.1 Block types 10 and 11 are produced on tape from 80 or 90 column cards through use of AJAX (which see). The tape block positioning of information corresponds with column positioning on cards.

11.2 Type 10 must precede all source programs or PROCS and type 11 must follow all source programs or PROCS that are being filed on tape from cards.

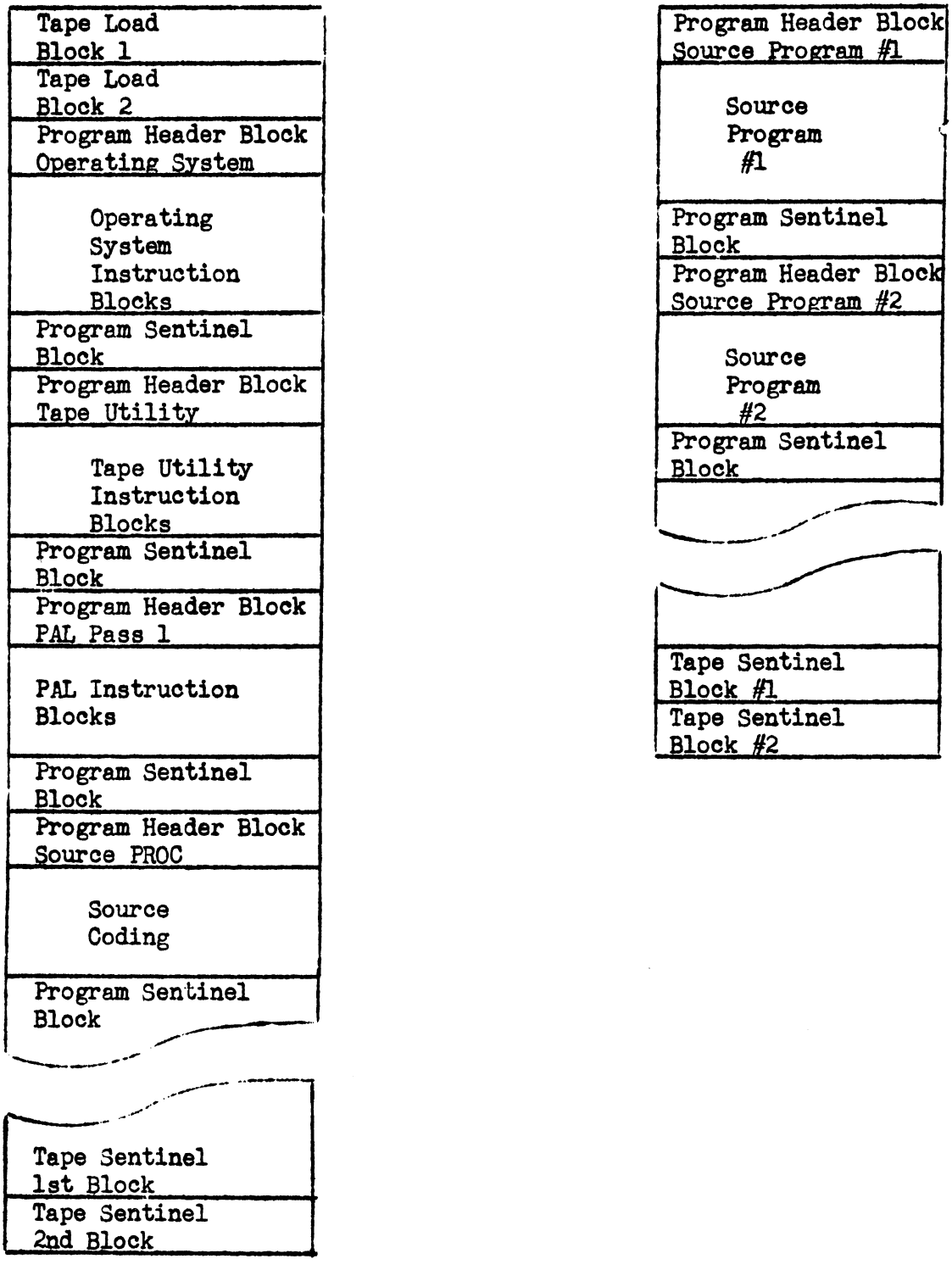
12.0 SOURCE CODE BLOCK (87 Characters in Length)

Position 0	E (030) Block Type
Position 1-3	Block number (binary)
Position 4-5	Unused
Position 6-79	Characters 7-80 of source code
Position 80-86	Page-Line-Insert number

12.1 NOTE:

Object header and sentinel cards correspond character for character with object header and sentinel blocks when filing cards on tape.

TAPE LAYOUTS - FIGURE 1



Systems tapes and libraries are constructed through the use of AJAX and OPUS utility programs. (See Figure 1).

14.0 SYSTEM TAPES

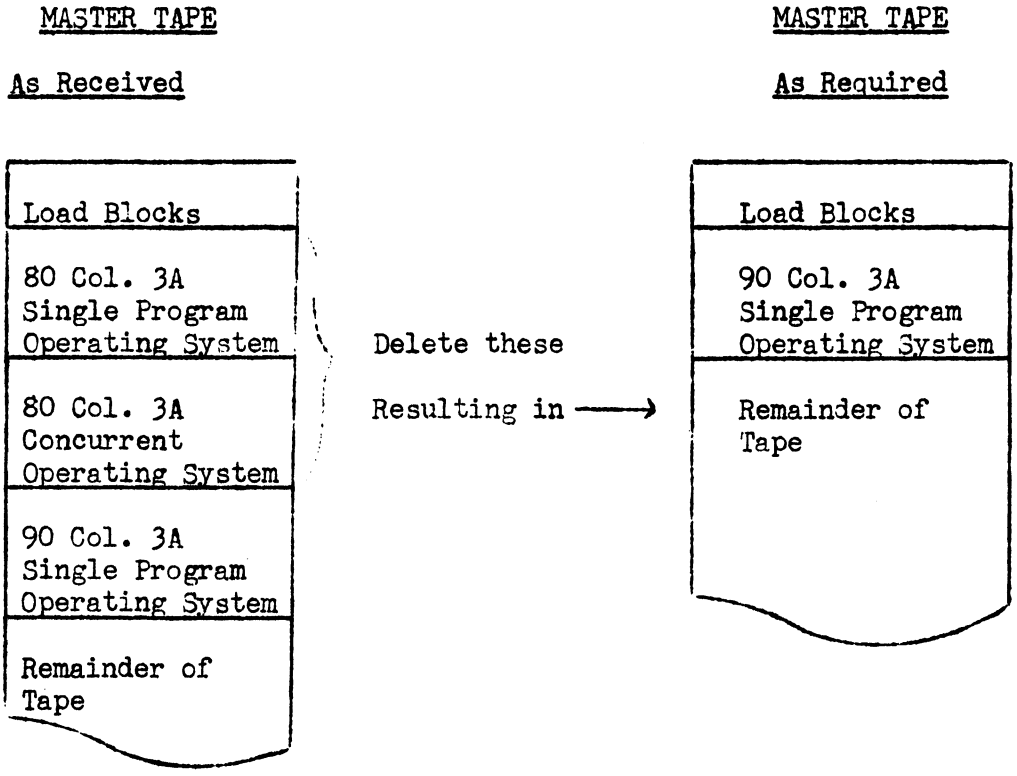
Each system tape contains, immediately following the load blocks, three (3) operating systems. They are:

14.1 a. 80 col., 3A, single program
b. 80 col., 3A, concurrent operation
c. 90 col., 3A, single program

14.2 The operating system desired by the user must immediately follow the load blocks on tape. Therefore, if the 80 col., 3A, single program version is not desired, the operating systems not desired must be deleted from the tape (use AJAX) in order to have the operating system desired immediately follow the loader. Through the use of AJAX a user may create as many different operating system master tapes as required. If none of the above operating systems is desired, a user may delete them all and file one of the various operating systems available through use of the OPS source code PROC. (See UNIVAC 1050 Operating System documents.)

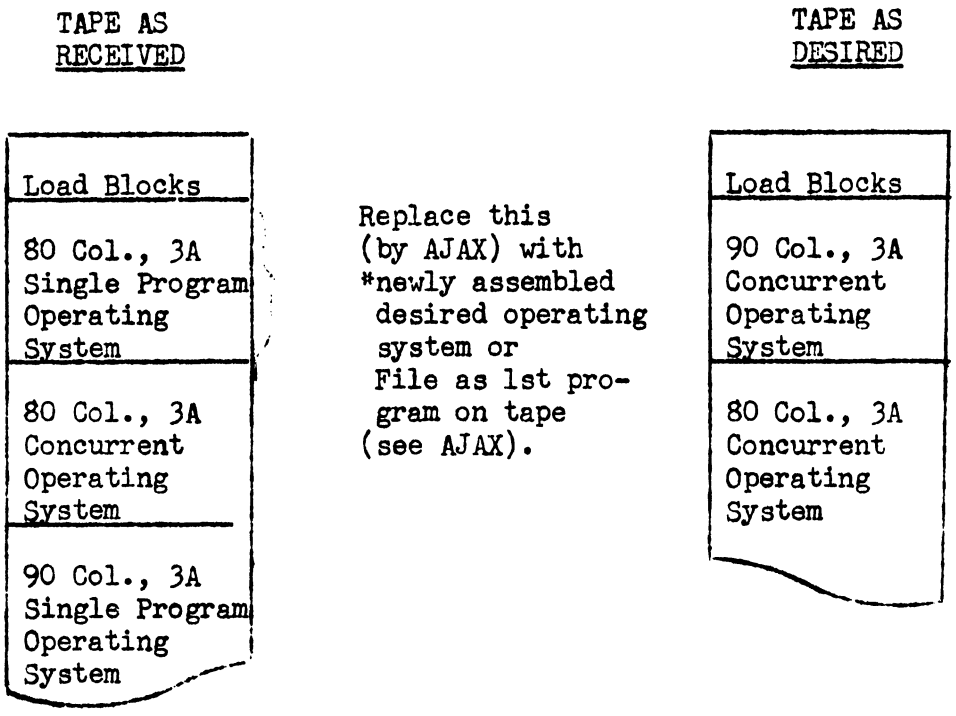
14.3 EXAMPLE 1.

To create a 90 col., 3A, single program operating system master tape.



14.4 EXAMPLE 2.

If none of these operating systems are desired a user may simply assemble the particular operating system desired and replace the 80 col., 3A, single program operating system with his assembled version or by filing it as the first routine on tape.



14.5 *See operating system documents for information on how to assemble the desired operating system.

15.0 LIBRARY TAPES

Library tapes may be created as desired (see Figure 1) in order to maintain source programs on tape. They may be directly assembled from this library. (See PAL Tape Assembler instructions). Library tapes may consist of either or both object and source code and must follow the tape conventions herein described. All programs either source or object must be preceded by a program header card and followed by a program sentinel card when being filed on tape.

16.0 3C SYSTEM TAPES

Each 3C system tape will contain immediately following the tape load blocks two operating systems. They are:

16.1 a. 80 col., 3C, single program
b. 80 col., 3C, concurrent operation

This tape varies from the standard 3A systems tape in this respect (See 14.0).

NOTE: 3C tapes will be in compatible mode at 556 BPI.

AJAX

TAPE MAINTENANCE SYSTEM

FOR THE UNIVAC 1050

AJAX

AJAX is an integrated set of service routines providing the functions necessary to create and maintain Tape Libraries, Systems Libraries and Master Instruction Tapes.

AJAX is essentially a tape to tape utility program with command cards from the Reader directing its services. Programs (object and source) may be filed, replaced, deleted, printed, copied, listed, altered and punched.

AJAX will not accept object cards to be filed or replaced from the Reader nor will it punch object cards from tape. These functions are provided by the Object Pal Utility Service, OPUS (which see).

Tape formats are those described under 1050 System Tape formats.

Card formats are those of the Pal 1050 Assembly System.

There are four types of Command Cards:

Type 1. I/O Command - INOUT describes the major input and output peripherals.

Type 2. Program Correction Commands - DELE, FILE, REPL, COPY & ALTER.

Type 3. Service Commands - PRINT, LIST, PUNCH.

Type 4. Misc. Commands - STOP, SORS, HDR.

INOUT COMMAND.

FORM:

OP	OPERANDS
INOUT	p1, p2, p3

where: p1 = specifies the Input device
C = Reader
0 thru 9 = Tape Unit

p2 = specifies the Output device
C = Punch
P = Printer
0 thru 9 = Tape Unit

p3 = blank or L

If L, AJAX will produce a two block 17000_8 Tape Loader as the first two blocks on the Output Tape. This function performed only if p2 names a Tape Unit.

NOTE:

The following parameter combinations on the INOUT card are not acceptable:

C,C T,T (if both name same unit).

An INOUT card must precede all other Command cards.

PROGRAM CORRECTION COMMANDS

DELE COMMAND.

FORM:

OP	OPERANDS
DELE	p1, p2

where: p1 = program to be deleted (Max. 4 char. I.D.)

p2 = blank or SKIP, if SKIP, all programs from present position of Input Tape to p1 will be deleted, otherwise, they will be copied.

NOTE:

The SKIP parameter is discussed at the end of this document.

COPY COMMAND.

FORM:

OP	OPERANDS
COPY	p1, p2

where: p1 = program to be copied (Max. 4 char. I.D.)

p2 = blank or SKIP, if SKIP, all programs from present position of Input Tape to p1 will be deleted, otherwise, they will be copied.

REPL COMMAND.

FORM:

OP	OPERANDS
REPL or	p1, p2 (source only)
REPL(T)	p1, p2 (source or object from tape)

where: p1 = program to be replaced (Max. 4 char. I.D.)

p2 = blank or SKIP, if SKIP, all programs from present position of input tape to p1 will be deleted, otherwise, they will be copied.

(T) = Names Tape Unit (0-9) containing replacement program.

FILE COMMAND.

FORM:

OP	OPERANDS
FILE or	p1, p2 (source only)
FILE(T)	p1, p2 (source or object from tape)

where: p1 = program after which the filed program is to be placed, unless it is to be the first program on tape or if it is to be filed immediately after a program just operated on, in which case p1 must equal "HERE".

p2 = blank or SKIP as in REPL.

(T) = Names Tape Unit (0-9) containing program to be filed.

ALTER COMMAND.

FORM:

OP	OPERAND
ALTER	p1, p2

where: p1 = program to be altered (Max. 4 char. I.D.)

p2 = blank or SKIP as in REPL.

This card must precede a SORS or HDR correction command (which see) and is used only in conjunction with them.

*SERVICE COMMANDS - (No Output Tape)

PRINT COMMAND.

FORM:

OP	OPERAND
PRINT	p1, p2

where: p1 = program to be printed (Max. r char. I.D.)

p2 = Δ or SKIP, if Δ AJAX will print from present position of Tape to p1 inclusive.

p2 of INOUT card must = p.

PUNCH COMMAND.

FORM:

OP	OPERAND
PUNCH	p1, p2

where: p1 = program to be punched.

p2 = Δ or SKIP, if Δ AJAX will punch from present position of Tape to p1 inclusive.

p2 of INOUT card must = C.

LIST COMMAND.

FORM:

OP	OPERAND
LIST	

This command has no parameters and will LIST each program header w/sentinel that is contained on the Input Tape.

p2 of INOUT card must = p.

NOTE:

The SKIP PARAMETER should not be used on the STOP CARD following service commands.

---

*Not to be used with Program Correction Commands, or intermixed with each other since INOUT Command prohibits use of Printer and Punch simultaneously.

MISC. COMMANDS

STOP COMMAND.

FORM:

OP	OPERAND
STOP	p1, p2

This command is always the last command to AJAX and must always be present.

If p1 is blank, AJAX will copy the remainder of the input to the output as preset by the INOUT card except in the case of PRINT and PUNCH.

If p1 is "SKIP", AJAX will terminate the output at its present location.

p2 may be blank or "END".

If p2 is "END", AJAX will reload the operating system from Tape Unit 0 when its functions are complete after depressing the start button at the successful completion STOP. If p2 is Δ AJAX will re-initialize, upon depressing start.

The following two commands must be preceded by an ALTER card.

SORS COMMAND.

FORM:

OP	OPERAND
SORS	n1, n2, n3

This command is used to make individual source card corrections by the deleting, adding or replacing of individual lines of code.

where: n1 = page-line-insert no. where correction is to start. If n2 = 0, source cards which follow will be inserted after n1.

n2 = page-line-insert no. of last card of deletion. AJAX will delete from n1 to n2 inclusive when n2 ≠ 0.

n3 = number of source card corrections to be added which immediately follow this card. (0 to 9).

NOTE:

1. n1 = n2 if replacing one card.

2. If more than 9 consecutive source cards are being inserted a second SORS command is required for each additional set of 9 insertions, where n1 must equal "HERE", n2 must be Ø and n3 the number of cards.

HDR COMMAND.

FORM:

OP	OPERAND
HDR	

This command is used to replace a source or object header card. It must be immediately followed by the new header card.

(T) = Variable Tape Unit
*RRRR = 4 char. Prog. I.D.

EXAMPLES

SERVICE	FUNCTION	COMMAND NEEDED		
PRINTING	Print Entire Tape	1.	INOUT	(T), P
		2.	STOP	
	Print One Program	1.	INOUT	(T), P
		2.	PRINT	*RRRR, SKIP
		3.	STOP	
	Print Several Programs	1.	INOUT	(T), P
		2.	PRINT	*RRRR, SKIP
		3.	PRINT	*RRRR, SKIP
		4.	STOP	
	List Programs from Tape	1.	INOUT	(T), P
		2.	LIST	
		3.	STOP	
PUNCHING (Source only)	Punch Entire Tape (Will SKIP Obj. Prog.)	1.	INOUT	(T), C
		2.	STOP	
	Punch One Program	1.	INOUT	(T), C
		2.	PUNCH	*RRRR, SKIP
		3.	STOP	
	Punch Several Programs	1.	INOUT	(T), C
		2.	PUNCH	*RRRR, SKIP
		3.	PUNCH	*RRRR, SKIP
		4.	STOP	SKIP
CARD TO TAPE (Source only)	Card to Tape	1.	INOUT	C, (T)
		2.	STOP	
	Card to Tape, Tape LdR to be added.	1.	INOUT	C, (T), L
		2.	STOP	
		3.	Cards follow in reader.	
		{		
		99.	% Tape Sentinel	

SERVICE	FUNCTION	COMMAND NEEDED
COPYING	Copy Tape w/Loader	1. INOUT (T), (T), L 2. STOP
	Copy Tape w/o Loader	1. INOUT (T), (T) 2. STOP
	Copy One Program w/o Loader	1. INOUT (T), (T) 2. COPY *RRRR, SKIP 3. STOP SKIP
	Copy all Programs from Card #2 to Card #3 inclusive with Tape Loader	1. INOUT (T), (T), L 2. COPY *RRRR, SKIP 3. COPY *RRRR 4. STOP SKIP
FILING	File Cne Program from Cards (Source Only) and copy remainder of Input	1. INOUT (T), (T) 2. FILE *RRRR (Program before the one filed) 3. Program to be FILED 99. STOP
	File One Program from Tape Unit 4 (Source or Object) at the present position of the Tape and copy remainder of Input.	1. INOUT (T), (T) 2. FILE4 HERE 3. STOP
	File One Program from Tape Unit 3 and terminate Output at that poin on Tape.	1. INOUT (T), (T) 2. FILE3 *RRRR (Program before the one filed) 3. STOP SKIP

SERVICE	FUNCTION	COMMAND NEEDED		
REPLACING	REPL one Program from Cards (Source only) and copy remainder of Tape.	1.	INOUT	(T), (T)
		2.	REPL	*RRRR
		3.	Program Replacement	
		99.	STOP	
	REPL one Program from Tape Unit 4 (Source or Object) and copy all of Input.	1.	INOUT	(T), (T)
		2.	REPL4	*RRRR
		3.	STOP	
	REPL one Program from Tape Unit 6 and another from Tape Unit 5 and copy all of Input.	1.	INOUT	(T), (T)
		2.	REPL6	*RRRR
		3.	REPL5	*RRRR
		4.	STOP	
	REPL two programs from Tape Unit 7 and copy all of Input. Write Loader on front of Tape.	1.	INOUT	(T), (T), L
		2.	REPL7	*RRRR
		3.	REPL7	*RRRR
		4.	STOP	
DELETING	DELETE one program, LdR to Output Tape.	1.	INOUT	(T), (T), L
		2.	DELE	*RRRR
		3.	STOP	
	DELETE programs CCCC thru TTTT from a Tape consisting of routines AAAA thru ZZZZ w/o Tape Loader.	1.	INOUT	(T), (T)
		2.	DELE	CCCC
		3.	DELE	TTTT, SKIP
		4.	STOP	
SOURCE CORRECTIONS WHILE COPYING	Insert 5 Source Cards in program BBBB, after page-line-insert #00600 and copy remainder of Tape.	1.	INOUT	(T), (T)
		2.	ALTER	BBBB
		3.	SORS	00600, 0, 5
			-5	Source Additions
		9.	STOP	
	Same as above but delete lines 00600 to 00605 while making additions.	1.	INOUT	(T), (T)
		2.	ALTER	BBBB
		3.	SORS	00600, 00605, 5
			-5	Source Additions
		9.	STOP	

SERVICE	FUNCTION	COMMAND NEEDED	
SOURCE CORRECTIONS WHILE COPYING	Delete lines 00600 to 00610 in program BBBB while skipping from beginning of the Input to BBB and copying remainder to Output.	1. INOUT 2. ALTER 3. SORS 4. STOP	(T), (T) BBBB, SKIP 00600,00610,0
HdR CHANGE	Replace header program BBBB Output Tape to have a Tape Loader and copy entire Tape.	1. INOUT 2. ALTER 3. HdR 4. New HdR Card 5, STOP	(T), (T), L BBBB

IMPORTANT NOTES ON AJAX

1. The output tape will never have a tape loader on the front unless the p3 parameter of the INOUT card is "L".

2. OBJECT CARD handling is not performed by AJAX, use OPUS (Object Pal Utility Service.)

3. Service Commands (PRINT, PUNCH, LIST) perform one service at a time and may not be intermixed.

4. On FILE(T) and REPL(T) commands the tape unit is variable (0-9). Programs to be filed or replaced may be stacked on the same tape unit or not, as desired, but must be in positional filing sequence.

5. PRINT, PUNCH and LIST do not produce an output tape, therefore they cannot be intermixed with program commands. Program and misc. commands, however, may be intermixed as desired.

6. Programs are operated on in tape sequence only.

7. The examples, although extensive, are not complete. Use of the p2 (SKIP) parameter can be very useful in automatic deletion, while performing other functions. (See discussion SKIP parameter at end of document).

8. The Prog-ID may be 4 characters or less.

AJAX COMMAND LIST

COMMAND	PARAMETER 1	PARAMETER 2	PARAMETER 3
INOUT	Input Unit	Output Unit	Tape Loader
REPL(T)	Prog. Replaced	SKIP or Δ	N. A.
FILE(T)	Prog. before Filed one.	SKIP or Δ	N. A.
DELE	Prog. Deleted	SKIP or Δ	N. A.
COPY	Prog. Copied	SKIP or Δ	N. A.
ALTER	Prog. Altered	SKIP or Δ	N. A.
SORS	First line of Dele. or if p2=0 line before insertion.	Last line of Deletion or 0 if no deletion.	No. of added cards
HDR	N. A.	N. A.	N. A.
PRINT	Prog. Printed	SKIP or Δ	N. A.
PUNCH	Prog. Punched	SKIP or Δ	N. A.
LIST	N. A.	N. A.	N. A.
STOP	Δ.= Copy SKIP = Terminate	Δ = Do Nothing END = Load OSO when AJAX functions, completed.	N. A.

NOTE:

SKIP always effects the programs from the present position of the tape to p1 (See discussion of SKIP next page).

SKIP

The SKIP parameter may be used with all commands except the following: LIST, SORS and HDR.

Its presence or absence has a large effect on the command which contains it.

Example:

Assume a tape library consisting of programs numbered 0001 to 0400. The following two commands,

```
DELE 0010
DELE 0040
```

would result in the deletion of the two routines only, however,

```
DELE 0010
DELE 0040, SKIP
```

would result in the deletion of all routines from 0010 through 0040 inclusive, and

```
DELE 0010
STOP SKIP
```

would result in the deletion of the entire tape past 0009. The same pattern follows with REPL, FILE, ALTER and COPY.

A. SKIP (p2) when used with program correction commands (DELE, REPL, FILE, COPY and ALTER), performs automatic deletion from the present position of the tape to p1.

B. SKIP (p2) when used with program service commands (PRINT and PUNCH) will delete the function of PRINT or PUNCH from the present position of tape to p1. In other words, its absence allows Punching or Printing from present position of tape to p1 inclusive.

AJAX STOPS

M ADDRESS	REASON AND ACTION
021110	AJAX ready - depress start.
040x44	Block-ct. error x-tape, restart.
020772	Tape unit in FILE or REPLACE already in use from INOUT card. Move tape, reload new command card.
024141	Card to tape command not followed by a stop card, load stop card and depress start.
024142	Tape to print command not followed by stop, list, or print card, load proper command, depress start.
024143	Tape to card command not followed by stop or punch command. Reload proper command, depress start.
022220	Cannot locate p1 (maybe tape positioning) restart.
022222	Object programs not acceptable from reader (use OPUS).
022223	Object punch requested - not provided, restart (use OPUS).
020700	Command not recognized, reload proper command, depress start.
027777	Successful completion, depress start to re-initialize, or load Ops as designated by stop card.
020510	Header missing REPL or FILE, reload HdR card or new tape and hit start.
024222	No stop card after LIST, load stop card and depress start.
020100	Page-line-insert number cannot be located restart.
020300	N3 = 0 on source insertion, reload SORS command, depress start.

M ADDRESS	REASON AND ACTION
020550	Parameter size error - reload proper command card and depress start.
010000	Printer off normal, clear problem, depress start.
012000	Punch off normal, clear problem, depress start.
011000	RdR error, reload 1 card unless the reader motor is off, depress start (serial reader always reload one). (except for off-line).
027272	No INOUT card, Load one and hit start.
140x66	Where x equals tape unit. Tape parity, depress start for recovery attempt.
140x55	Unrecoverable tape error. Restart.

OPUS

OBJECT CODE MAINTENANCE SYSTEM

FOR THE UNIVAC 1050

OPUS

OPUS is a service routine designed to provide the functions necessary to maintain PAL object code on Tape Libraries, Systems Libraries and Master Instruction Tapes. The PUNCH command also makes it possible to obtain a program deck of cards.

OPUS allows object programs to be filed and replaced from cards and by use of the ALTER and SQZE commands makes it possible to correct object code on tape.

OPUS does not provide any source program facilities. These services are provided by AJAX (see AJAX, June 1964).

Tape formats are those described under 1050 System Tape Formats.

Card formats are those of the PAL 1050 Assembly System.

The following commands pertaining to object code are accepted by OPUS:

1. FILE
2. REPL
3. PUNCH
4. ALTER
5. SQZE
6. STOP

Input and Output Libraries.

With the exception of the PUNCH command the input library is assumed to be on tape unit 0 and the output library will be produced on tape unit 1.

The PUNCH command does not produce a tape output library and assumes the input library to be on tape unit 1.

Octal Numbers.

An octal number must be preceded by a decimal zero.

Headers and Sentinels.

All programs being filed or replaced from the card reader must have proper headers and sentinels as described in the 1050 System Tape Formats.

FILE COMMAND

FORM:

OP	OPERANDS
FILE	p1

where: p1 = the 4 character program I.D. after which the program in the card reader is to be placed, unless it is the first program on tape or if it is to be filed immediately after the program just operated on in which case p1 must equal "HERE".

or where: p1 = "ALL" in which case all the object programs in the reader will be written on tape unit 1 until the first 'Z' sentinel in the reader is encountered (Z in column 1).

REPL COMMAND

FORM:

OP	OPERANDS
REPL	p1

where: p1 = the four character program I.D. of the object program to be replaced.

ALTER COMMAND

FORM:

OP	OPERANDS
ALTER	p1, p2

where: p1 = the four character program I.D. of the object program to be altered.

p2 = an octal number indicating the number of characters that have been added to a program by use of the SQZE command.

This increment will be added to the total number of characters and high address fields contained in the program 'R' block.

SQZE COMMAND

FORM:

OP	OPERANDS
SQZE	p1,p2,p3,p4,p5,p6,pn

where: p1 = the name of the program or segment to be corrected.

p2 = the starting address in octal of the characters to be corrected or added to the program.

p3 = the correction in octal. A maximum of 16 characters (8 store locations) may be corrected with one SQZE card.

p4 = "R" if the starting address of the correction requires base address modification at load time. If not, p5 becomes p4, etc.

p5,p6,pn = octal numbers pointing to the least significant characters within the correction that may require base address modification at load time. If no modification is required, no parameters are needed.

PUNCH COMMAND

FORM:

OP	OPERANDS
PUNCH	p1

where: p1 = the four character program I.D. of the object program to be punched.

or p1 = "ALL" in which case all of the object code contained on the input library will be punched.

NOTE:

1. No program headers or sentinels will be punched.
2. The input library for the PUNCH command is assumed to be on tape unit 1.

STOP COMMAND

FORM:

OP	OPERANDS
STOP	p1

This command is always the last command to OPUS and must always be present.

If p1 is blank, OPUS will copy the remainder of the input library on tape unit 0 to the output library on tape unit 1 except in the case of PUNCH and FILE "ALL". OPUS will re-initialize if program start is depressed.

If p1 is "END", OPUS will proceed as above. If program start is depressed, the operating sysytem will be reloaded from tape unit 0.

EXAMPLES

SERVICE	FUNCTION	COMMANDS NEEDED
PUNCHING (Object only)	Punch one program.	1. PUNCH *RRRR 2. STOP
	Punch all object code on tape.	1. PUNCH ALL 2. STOP
FILING (object only)	File one object program from cards.	1. FILE *RRRR (program before the one filed. 2. Program to be filed. 99. STOP
	File one program from the card reader at the present position of the output library on tape.	1. FILE HERE 2. Program to be filed. 99. STOP
	File all the object programs in the reader on tape unit 1.	1. FILE ALL 2. Programs to be filed. 3. STOP (Must be preceded by a 'Z' sentinel card, Z in col. 1.)
REPLACING (Object only)	Replace one program from cards and copy remainder of tape.	1. REPL *RRRR 2. Object program to be replaced. 99. STOP
OBJECT CODE CORRECTIONS	Correct 2 characters in program AAA which is absolute. The 2 characters start in location 04000.	1. SQZE AAA,04000,07777 99.
	Add a jump instruction in relative program B. The starting address and the instruction address need base address modification at load time. The instruction is to be loaded into program relative location 05000.	1. ALTER B, 05 2. SQZE B, 0500, 03000051000, R, 04 (Same line)

*RRRR is program I.D. consisting of from 1 to 4 characters.

IMPORTANT NOTES ON OPUS

1. Source card handling is not performed by OPUS, use AJAX.

2. The PUNCH command may not be used in conjunction with any other command except PUNCH and STOP.

3. Programs are operated on in tape sequence only.

OPUS STOPS

M ADDRESS	REASON AND ACTION
027777	OPUS ready - depress start. If STOP "END" command has been used the operating system will be reloaded.
020700	Improper command - reload proper command and depress start.
020400	Routine header missing while filing or replacing. Reload proper header card and depress program start.
020300	Error SQZE command. Restart.
022220	Cannot locate program on tape. Restart.
110000	Reader error, reload 1 card if present in error stacker, depress start.
040u44	Block count error on tape unit u, restart.
140u55	Tape error on tape unit u, restart.
140u66	Tape error on tape unit u, program has rocked tape 5 times. To continue, depress start. Program will attempt to recover.
120000	1. Punch error, for read check, stacker full, hopper empty and off line, depress ready and program start. 2. In most cases, it is possible to recover from other punch errors. The punch track must be cleared and blank cards manually fed through all stations. After this is done depress ready and start. A read check will occur, proceed as in 1, above.

DISPLAY THE CONTENTS OF MEMORY.

Any storage position can be displayed when the processor has been brought to an orderly stop. The following can be employed:

Display First Character.

1. Depress the CC-Display/Alter Selection.
2. Record the value of CC.
3. Depress the MEM-Display/Alter Selection button.
4. Set up the desired address in the M portion of the Alteration switches.
5. Depress the Display button.

The contents of the desired storage position will be displayed in the C portion of the Display lights. The address+1 set up in the M portion of the Alteration Switches will be displayed in the M display lights. This new address will be available for additional sequential displays.

Display the Second and Subsequent Sequential Characters.

6. Depress SEQ Display/Alter Selection button.
7. Depress Display button.

Repeat the last step for each new character in sequence to be displayed. The storage address is automatically incremented after each storage character has been displayed.

ALTER THE CONTENTS OF MEMORY.

Any storage position can be altered when the processor has been brought to an orderly stop. The following procedure can be employed:

Alter First Character.

1. Depress the CC-Display/Alter Selection button.
2. Record the value of CC.
3. Depress MEM-Display/Alter Selection button.
4. Set up desired Address in the M portion of the Alteration Switch.
5. Set up the bit value of the character to be inserted in C portion of the Alteration Switches.
6. Depress the Alter button.

The contents of the desired storage position will be filled with the character represented in the C Alteration Switches. The address+1 set up in the M portion of the Alteration Switches will be displayed in the M display lights. This new address will be available for additional sequential alterations.

Alter Second and Subsequent Sequential Characters.

7. Depress the SEQ Display/Alter Selection button.
8. Set up the desired character in C portion of the Alteration Switches.
9. Depress the Alter button.

Repeat Steps 8 and 9 for each new character in sequence to be altered. The storage address is automatically incremented after each insertion.

TRACE FOR ZERO OPERATION CODE.

It may be desirable to trace program execution for a zero Op-code, if so, use following procedure.

1. Load program normally to base address display stop.
2. Depress Op (a Trace Mode button).
3. Set Op portion (upper 5 bits) of instruction display register to all zeros.
4. PROGRAM START.
5. If an instruction with a zero Op code is referenced; Trace Stop (a Trace Mode button) will light with a computer STOP.
6. Depress CC to display location +5 of zero Op code.

MANUAL INSTRUCTION EXECUTION.

The Processor Control Panel can be used to perform operator-created instructions for all instructions with the exception of the Jump Loop and the indexing functions.

1. Depress the One Instruction-Mode button.
2. Depress the CC-Display/Alter Selection button.
3. Record the value of CC displayed in the M portion of the Display Lights.
4. Depress the Inst. -Display/Alter Selection button.
5. Record the value of the thirty display lights if required for a later operation.
6. Set up the new instruction in thirty Alteration Switches.
7. Depress the Alter button.
8. Depress the Program Start button.

The new instruction will be performed instead of the instruction previously staticized in the instruction register. The processor, after completing this new instruction, will bring the next instruction stored at the address specified by the control counter at the end of executing the operator generated instruction, and stop.

The reason for the special handling of a Jump Loop instruction is that when this instruction is staticized, the control counter has only been incremented four times rather than the usual five times.

Steps 1 thru 4 and Step 8 above could be used to step through instruction execution of any desired address area.

1050 REPRODUCER - OPERATING INSTRUCTIONS

The Reproducer will reproduce source and object cards. It is a LOAD and GO Routine. The cards to be reproduced are loaded immediately behind it in the Reader. A blank card terminates the cards to be reproduced.

PROCEDURE: Object Cards.

1. Load the Reproducer and the cards to be reproduced into the Reader input hopper.

2. Follow the "Load Card Procedure".

The Reproducer need not be reloaded between reproductions. To continue, depress start.

PROCEDURE: Source Cards.

1. Follow the same procedure as for object cards. If it is desired to reproduce the Program I. D. taken from the label of the 'BEGIN' card, depress sense switch #2, and this I. D. will be produced on each card. If it is desired to re-sequence the card numbers, depress sense switch #1.

TDMP OPERATING INSTRUCTIONS

TDMP is a relocatable program which may be loaded from the systems tape when it is desired to print one or more memory dumps written onto tape by the OPS. (Refer to OPS write-up, section IVC.)

1.0 Mount memory dump tape on logical servo 1.

2.0 Set printer as desired.

3.0 Load TDMP using normal call procedure.

4.0 Stop Ø2ØØØ1: TDMP initialized. Press PROGRAM START to commence printing.

5.0 Stop Ø2ØØØ7: memory dump has been printed.

a. To print another memory dump: set trace-address switches to Ø1, trace mode to PROC, and press PROGRAM START. Program will come to stop Ø2ØØØ1.
b. To release TDMP; set trace-address switches to other than Ø1, trace mode to PROC, and press PROGRAM START. The OPS will come to stop Ø7ØØØ1.

6.0 ERROR STOPS:

Use the normal error-recovery procedure for the following peripheral-error stops.

a. Ø1ØØØØ: printer error
b. Ø14Ø155: memory parity error
c. Ø14Ø166: tape parity error
d. Ø14Ø177: servo off-line or nonready

7.0 FORMAT:

The memory dump is printed in octal format

a. Tetrad Area: Each line displays the contents of 8 tetrads as follows: address of MSC of Tn, (Tn), (Tn+1), address of MSC of Tn+2, &c.
b. Non-Tetrad Memory: Each line displays the contents of 40 character positions as follows: address n, (n through n+4), (n+5 through n+Ø11), address n+Ø12, &c.
c. Duplicate Lines: If the 40 characters of a non-tetrad line are equal to the last 5 characters of the previous line, printing of the line is supressed, and the next 40 characters are examined. This process is continued until an inequality is detected or until the end of the dump is reached. A line of asterisks on the print-out indicates that one or more lines have been suppressed.

8.0 Since the memory dump was written while the OPS had program control, some information will appear in OPS working storages rather than in the memory positions it occupies when the worker program has control. This varies according to whether a single-program or concurrent OPS is being used.

a. Single-Program OPS

(1) 8 LSC of AR2 are destroyed
(2) Channel 5 interrupt entry appears in 8 LSC of AR2
(3) 4 MSC of AR2 are destroyed
(4) Contents of X1 appear in 4 MSC of AR2

b. Concurrent OPS

(1) 8 LSC of AR2 are destroyed
(2) Channel 5 interrupt entry appears in 8 LSC of AR2
(3) 3 LSC of AR1 are destroyed
(4) Contents of tetrads 16, 17, 18, 8, and index registers 1 through 7, for low order program appear in memory positions Ø13Ø2 through Ø1355
(5) The corresponding information for the high order program appears in memory positions Ø1356 through Ø1432

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 The function of sorting on the UNIVAC 1050 is to:

1. Accept a file of data, one record at a time,
2. Rearrange the file,
3. Produce the file, on request, one record at a time, for further processing.

1.2 The file is rearranged according to a transitive relationship between pairs of records so that for any ordered pair (A, B), the relationship is true wherever record A is produced for further processing before (in time) record B.

2.0 DATA

2.1 Size of Items or Records to be Sorted

The record size of any given sort must be fixed, i.e., record size is equal for all records to be sorted.

Minimum: 1 character
Maximum: 1024 characters

2.2 Volume of Data

Minimum: No data
Maximum: The maximum volume of data which can be sorted in a single sort run is determined by the amount of data which can be contained on any single reel of a collation phase tape. This varies according to the length of tape on servos available to the sort and the block size calculated by the sort.

If the data to be sorted exceeds this maximum, merges of the sorted data must be performed to produce an ordered file.

2.3 Key

The ordering relationship between records is define for portions of those records called the key. Each record must have the following properties:

2.3.1 Each key occupies the same character positions in every record where each key consists of one to ten fields.

2.3.2 Each field consists of contiguous character positions, where each field consists of one to sixteen characters.

2.3.3 Each field may be absolute binary or algebraic decimal, where the selected option holds for every record.

2.3.4 Each field within a record may be of a different type than other fields within the same record, and may be of the following types:

2.3.4.1 Ascending Binary.

As a result of a binary comparison, the field which is smaller is considered the chosen field and the record to which it belongs will be selected first.

2.3.4.2 Descending Binary.

As a result of a binary comparison, the field which is larger is considered the chosen field and the record to which it belongs will be selected first.

2.3.4.3 Ascending Decimal.

As a result of a decimal comparison, the field which is smaller is considered the chosen field and its record will be selected first.

2.3.4.4 Descending Decimal.

As a result of a decimal comparison, the field which is larger is considered the chosen field and its record will be selected first.

2.4 Selection of a Record

2.4.1 A comparison is performed upon field #1 of record A vs. field #1 of record B. A record is considered chosen according to the type of field previously defined.

2.4.2 If the first fields within the keys of two records are equal, the ordering relationship between the two records will be determined by the comparison upon field i of record A vs. field i of record B, where field i is the first case of inequality.

2.4.3 If all fields within the keys of two records being compared are equal, either one of the two records may be selected.

3.0 HARDWARE

3.1 Uniservo Tape Units Available to the Sort.

Type: IIIA, IIIC, IVC, VIC

Minimum: 3 servos
Maximum: 6 servos

All servos must be of same type.

3.2 Memory

Minimum: 8K characters

Maximum: 32K characters

3.3 Card Reader, Card Punch, Printer

None of the above mentioned peripheral units are required to run a sort program; however, the own code sections may use these devices as input or output media.

4.0 PROGRAM STRUCTURE

Each sort program is written by the programmer as he writes any other program. The sorting function is provided by a library subroutine. The use of the sort subroutine implies certain minimum characteristics of the calling program. In particular, there must be a section of the program which forms the first pass own code (FPOC) of the sort, and another section which forms the last pass own code (LPOC). Each section is very similar in form to any typical data processing program.

As such, the programmer has almost complete freedom to include data processing other than sorting in either or both sections. If there is an input tape file in the first pass, it would be controlled by a different subroutine called separately. The same would hold true if input data is read from cards, if output data of the last pass is written on tape, etc. The program communicates with the sort through the use of a jump return instruction. In the first pass, an item is sent to be sorted in the same way that it could be sent to be written in an output file. Control is not returned when the sort file is closed. In the last pass, the programmer asks for an item from the sort in the same way that he could ask for an item from an input file. This mechanism is intended to supply a maximum of first pass own code and last pass own code flexibility, with a minimum of extra learning. On the other hand, if the sort program is to do nothing but read an input tape, sort the data, and write the data on an output tape, the calling program is very short and completely formalized.

5.0 LOGICAL DESIGN

The general method used is sorting by merging. [1]

5.1 Dispersion Pass Method.

The Dispersion Pass will employ replacement selection. [1] Each record will be moved from an area in the user's FPOC to the sort tournament[2] area; each record will also be moved from the tournament area to an output area. The tournament size will be determined at object time from the amount of memory available to the dispersion pass after the amount of space used by:

1. Operating System, and
2. Dispersion Pass, and
3. Output areas, and
4. FPOC (include input routine and input areas) has been determined.

The smallest possible tournament size is two.

5.2 Collation (Merge) Phase Method

The collation phase employs a version of the forward read polyphase[3] method. Each time a record is read and written, it is moved within memory only once.

The method used to dispose of dummy strings is one that was derived from methods proposed by Mendoza[4] and Malcolm[5]. The combined method contains the simplicity claimed by Malcolm and the efficiency demonstrated by Mendoza. This method adds theoretical dummy strings to the actual string distribution to advance all string counters to the next highest "ideal level." Hence, it is not necessary to write any data on tape to illustrate the presence of dummy strings; only internal processing is required to eliminate them. The theoretical dummy strings will be collated in the first two cycles of the polyphase merge. Dummy strings will be merged prior to actual strings.

A positive sequence check on the output of every pass except the last is generated by inserting a parameter in the SORT3 call. (See 7.1).

5.3 Rerun

After the distribution of strings is completed onto k tapes, collation is ready to begin. The sort will proceed thru as many polyphase cycles as necessary to begin last pass. After each of these cycles is completed, two tapes are rewound.

1. The tape just exhausted of input.

2. The output tape.

The programmer has the option to establish rerun points by inserting a parameter in the SORT3 call (see Section VII). If the programmer has decided to establish rerun points, the tape exhausted of input will be rewound with interlock after the second cycle and must be replaced with a blank. The same procedure will be followed after the succeeding 3rd, 4th, . . . , (k+1)th cycles. Now, k tapes will have been removed and these tapes will contain all the data and sufficient information for rerun purposes. When the next cycle is completed, the tape that rewinds with interlock and the most recently rewound (k-1) tapes comprise the latest rerun point. The operator may keep the most recent rerun point by removing the tape just rewound with interlock and mounting a blank on that servo or ignore the most up-to-date rerun point, thus saving the time required to mount a blank tape.

5.3 (continued)

After each tape is dismounted it should be labeled by the operator. This label should identify the servo from which the tape was removed and a number which indicates the most recent k tapes.

If the programmer has not exercised the option of establishing rerun points, no tapes will be rewound with interlock. If a malfunction occurs which will not allow the sort to continue, the operator may continue the sort from the latest rerun point.

5.4 References

1. Friend, E. H., Sorting on Electronic Computer Systems, Journal of the Association for Computing Machinery, Vol. 3-July 1956, p. 134-168.

2. Goetz, M. A., Internal and Tape Sorting Using the Replacement-Selection Technique, Communication of the ACM, Vol. 6 - p. 202-205.

3. Gilstad, R. L., Polyphase Merge Sorting - An Advanced Technique, Proceedings of the EJCC, Dec., p. 144-147.

4. Mendoza, A. G., A Dispersion Pass Algorithm for the Polyphase Merge, Communications of the ACM, Vol. 5, October, 1962, p. 502-504.

5. Malcolm, D. M., String Distribution for the Polyphase Sort, Communications of the ACM, Vol. 6 - May 1963, p. 217-220.

6.0 ENVIRONMENT

6.1 Software Support

All sort generated programs will function with an Operating System.

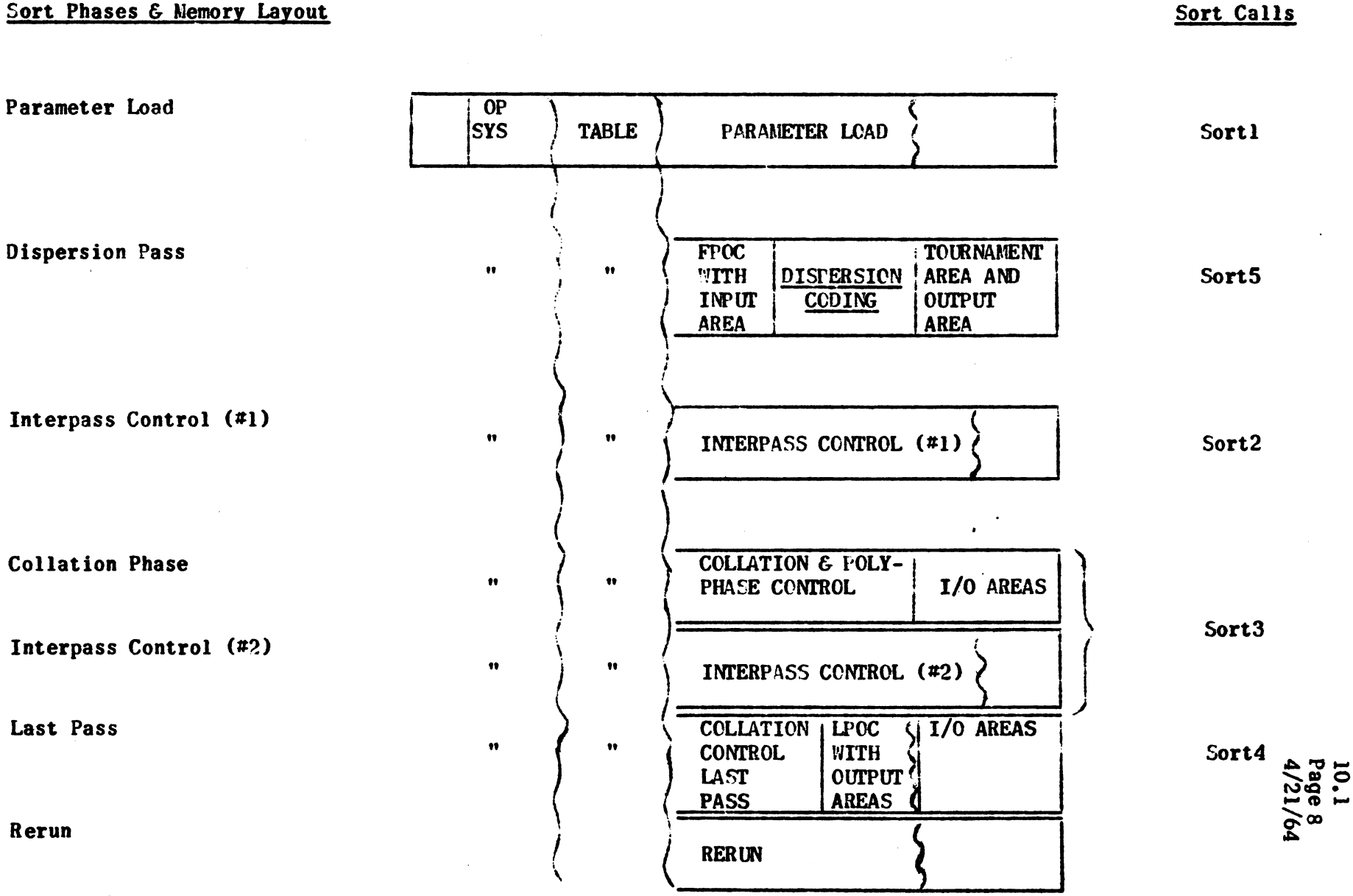
6.2 Operational Characteristics

The sort object program consists of an instruction tape generated by the sort generation run. In order to run the sort program it must be mounted on servo 0. After the program is initially loaded by the operator, the remaining phases are located automatically by the sort program. Seven separate phases exist:

1. Parameter Load
2. Dispersion Pass (including FPOC)
3. Interpass Control (#1)
4. Collation Phase
5. Interpass Control (#2)
6. Last Pass (including LPOC)
7. Rerun

Figure 1 shows the contents of each of the above phases and their relative positions in memory at object time.

FIGURE I


Sort Phases & Memory Layout
Sort Calls
Parameter Load
OP SYS
TABLE
PARAMETER LOAD
Sort1
Dispersion Pass
"
"
FPOC WITH INPUT AREA
DISPERSION CODING
TOURNAMENT AREA AND OUTPUT AREA
Sort5
Interpass Control (#1)
"
"
INTERPASS CONTROL (#1)
Sort2
Collation Phase
"
"
COLLATION & POLY-PHASE CONTROL
I/O AREAS
Sort3
Interpass Control (#2)
"
"
INTERPASS CONTROL (#2)
Last Pass
"
"
COLLATION CONTROL LAST PASS
LPOC WITH OUTPUT AREAS
I/O AREAS
Sort4
Rerun
RERUN

6.3 Description of Sort Phases.

6.3.1 Parameter Load

Some of the functions performed in this phase are:

6.3.1.1 Loads a key comparison subroutine based on the key definition which is used at object time for all key comparisons.

6.3.1.2 Analyzes the statements referring to block size, memory size, servos and item size to build a table of information at object time which is used by subsequent phases of the sort.

6.3.1.3 Calculates a block size for this particular sort at object time.

6.3.1.4 Allows the operator to eliminate a servo from the sort if the servo allocation is such that this can be done (e.g. A servo can not be eliminated from a 3 servo sort). This may become necessary if a servo is inoperable at object time.

6.3.1.5 Copies the other sort phases as required onto collation servos. This allows the instruction tape to be removed and allows servo 0 to be used by the sort.

6.3.2 Dispersion Pass

6.3.2.1 Requires own code section.

6.3.2.2 Accepts records from the own code section one at a time, and disperses them in ordered sequences (strings) to the servos assigned to the sort and not used by first pass own code.

6.3.3 Interpass Control (#1)

There are two conditions which may require a redistribution of strings prior to the collation phase:

6.3.3.1 Certain levels of distribution when one LPOC servo is used. However, if this is the case, copying of strings in this phase will not normally occur.

6.3.3.2 When more than one of the servos assigned to the collation phase is reserved for FPOC use. The Dispersion pass will use only the servos not reserved for FPOC for the initial dispersion of data. If this condition exists, this phase will copy strings to all but one of the servos that were reserved for FPOC and also assigned to the collation phase.

6.3.4 Collation Phase

This phase uses the information left in memory (e.g. string counts, servo allocation, etc.) and the strings on several tapes. This phase then cycles until the strings are reduced to a number required by the last pass.

6.3.5 Interpass Control (#2)

If more than one of the servos assigned to the collation phase is reserved for LPOC use, a copy of string(s) may take place in order to release the servo(s) assigned to LPOC. However, this copy of data will not take place if only one servo is reserved for LPOC.

6.3.6 Last Pass

6.3.6.1 Requires an own code section.

6.3.6.2 Delivers all sorted records to the own code section one at a time.

6.3.7 Rerun

This phase of the sort is used only if a malfunction occurs which prevents the sort from continuing (e.g. unreadable tape, memory parity error, etc.). After rerun is loaded, all tapes will be validated before proceeding to insure that the proper tapes are mounted. After all tapes have been validated, they will be read forward the proper number of strings. Rerun will locate and read in the collation phase and the sort will continue.

7.0 GENERATION PROCEDURE

Specifications for the Sort, FPOC and LPOC are written on the standard PAL coding form. The first card must be a BEGIN card. The second card must be an *STD card to call the standard library definitions. The label field of the BEGIN card must contain a four character label in the form XXYY, where XX are unrestricted and $00 \leq YY \leq 90$.

7.1 The SORT is generated in five specific phases and these are referenced as follows:

Parameter Load	SORT1	p1, p2, p3, ... , pn
Interpass Control	SORT2	
Collation Phase	SORT3	p1, p2, p3
Last Pass	SORT4	
Dispersion Pass	SORT5	

SORT2, SORT4, and SORT5 contain no operands field entries. SORT1 contains operands field entries as follows:

p1 is the number of fields in the key.

p2 is the type and sequence desired for field 1 and may be

B - for ascending binary
D - for ascending decimal
DB - for descending binary
DD - for descending decimal

p3, etc., is the type and sequence desired for fields 2, etc., and may be entries as above.

SORT3 contains operands field entries as follows:

p1 is the number of servos available to the SORT.
p2 = RERUN if it is desired to have RERUN coding generated.
p3 = SEQCK if it is desired to have a positive sequence check during each collation phase.

This phase may be called in one of four ways:

SORT3	p1, p2, p3
SORT3	p1, , p3
SORT3	p1, p2
SORT3	p1

7.2 In addition to the specific references to the various phases of the sort, the following parameter cards must also be supplied immediately after the SORT1 parameter, and must appear in the order illustrated.

7.2.1 Minimum Block Size.

Written in one of the two following ways:

1. | MINBS | p1

where p1 represents the minimum block size (number of items per block) the Sort should use.

2. | MINBS | NONE

A minimum block size of fifty-six characters will be assumed, or if the record size is greater than fifty, the minimum block size will be one record length plus six.

7.2.2 Memory Size.

This parameter is written in one of three ways. It determines the amount of memory to be used by the Sort, which includes FPOC, LPOC, Sort coding and input-output areas, but which does not include memory used by the operating system. The three ways of writing this parameter are:

1. | MEMRY | p1

2. | MEMRY | p1, p2

3. | MEMRY | NONE

where p1 = minimum amount of memory to be reserved at assembly time, expressed decimally.

p2 = maximum amount of memory to be used at object time expressed decimally.

NONE = neither p1 nor p2 is desired to be specified.

The BEGIN card of the Sort program being assembled can be written in one of three logical ways:

1. | BEGIN | 1

   | BEGIN | 3

These are logically equivalent, and cause a relative program to be assembled which will be loaded at the beginning of an even row, i.e., a base address mod 128 will be assigned at load time.

2. | BEGIN | 5

This is essentially the same as above, but will be loaded slightly differently during concurrent running. (See below).

3. | BEGIN | absolute address

This causes an absolute program to be assembled.

The Operating System in use at time of running the Sort may be one of two basic types, i.e., it allows for single or concurrent running of programs. If the single type Operating System is in use, memory will be assigned as follows:

| MEMRY | p1 - p1 characters will be used

| MEMRY | p1,p2 - all of memory will be used, regardless of the value of p2

| MEMRY | NONE - all of memory will be used.

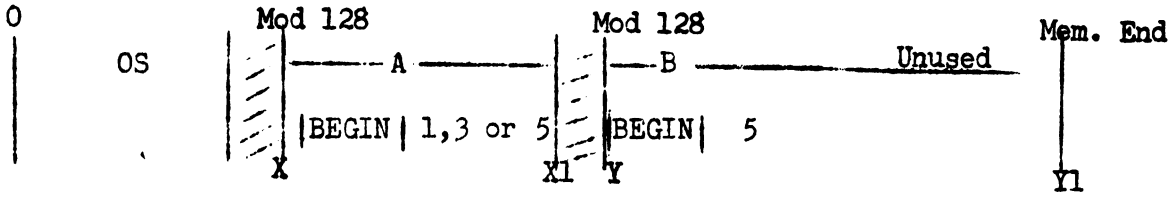
If the concurrent type Operating System is used, two relative programs may be run. However, only one absolute program may be run, i.e., the Operating System will not load a second program of either the absolute or relative type. When two relative programs are run, the first to be loaded will be called A, and the second to be loaded will be called B. Program A will always be loaded following the Operating System. Program B will be loaded following program A, if

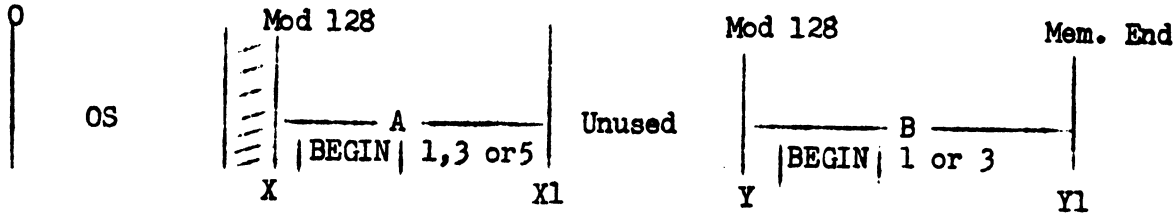
|BEGIN| 5

was used. Program B will be loaded at the high end of memory if

|BEGIN| 1 or BEGIN 3 was used.

Graphically





The values X, X1, Y and Y1 are stored in fixed memory locations within the operating system. After the Sort is loaded, it examines these values and sometimes changes X1 as necessary to reflect the usage of additional memory.

The following table shows the limiting value of the amount of memory *, or the last location number in memory used by the Sort#.

	PROG A BEGIN 1, 3 or 5	PROG B BEGIN 1 or 3	PROG B BEGIN 5	PROG ABSOLUTE
|MEMRY| p1	* p1	* p1	# Y1	* p1
|MEMRY| p1, p2	* p2, but not exceeding Y or Y1	* p1	# Y1	# Y1
|MEMRY| NONE	# Y or Y1	Will not Work	# Y1	# Y1

To obtain maximum utilization of memory, sorts which are to be run non-concurrently should use

|MEMRY| NONE

For Sorts to be run concurrently, they should use

|BEGIN| 5 and |MEMRY| NONE.

7.2.3 Type of Servo,

| TAPES | p1

p1 = 3A for IIIA tapes
3C for IIIC tapes
4C for IVC tapes
6C for VIC tapes

7.2.4 Servo Allocation,

| SERVO | p1, p2, ... , pn

p1, p2, etc., are the logical servo numbers (0-5) of the tape units available for sorting. At least 3 servos must be stated.

7.2.5 Last Pass Servos,

| LPSER | p1, . . . pn

p1, . . . pn are the logical servo numbers of the tape units available for use by LPOC.

7.2.5.1 If it is desired to use one or more servos for LPOC which have been specified in the SERVO statement, p1 must equal p1 of the SERVO statement.

7.2.5.2 The LPSER statement must have p1 = NONE if no servo is needed for LPOC.

7.2.6 First Pass Servos.

| FPSER | p1,...pn

p1,...pn are the logical servo numbers of the tape units available for use by FPOC.

7.2.6.1 If it is desired to use one or more servos for FPOC which have been specified in the SERVO statement, they must be specified in FPSER statement. If servo zero (0) has been specified in the SERVO statement, it must also be specified in the FPSER statement. (See para. 7.2.6.2 for the only exception).

If the SERVO statement contains a reference to servo 0, the Sort will rewind servo 0 with interlock after the load of the Dispersion Pass; otherwise, servo 0 will be left in a forward position.

7.2.6.2 The FPSER statement must have p1 = NONE if no servo is needed for FPOC.

7.2.7 Key Definition.

| KEYn | p1, p2

n - is the field number (1-10) for an n field key, the consecutive numbers 1 through n must be used for definition.

p1 is the length of the field and may be from 1 to 16 characters.

p2 is the position number of the rightmost character of the field in the record. p2 is defined relative to the number one.

7.2.8 Item Size.

| ITEM | p1

p1 is the number of characters in the record to be sorted.

7.3 FPOC including macros preceeds the SORT5 call. LPOC including macros preceeds the SORT4 call. An END card follows SORT5, andthe operands field must contain ZENT.

7.4 Writing Own Code.

7.4.1 General.

FPOC and LPOC should not use labels with a first letter of X, Y, or Z. The sort coding uses labels that begin with Y and Z and the standard I/O calls will produce labels that begin with X. If FPOC or LPOC uses labels that begin with X, Y, or Z, it is possible that an error of duplicate labels may occur at generation time.

7.4.2 First Pass Own Code.

7.4.2.1 BEGIN.

The first instruction to be executed by FPOC must be labelled ZBEGN. Control will be transferred to this location before First Pass is initialized. FPOC executes a JR ZSTRT.

7.4.2.2 Deliver and Item.

FPOC executes a JR ZPUT. Index register five must contain the absolute address (MSD) of the input item to be sorted.

7.4.2.3 No More Items.

FPOC executes a J ZEND.

7.4.2.4 Index Registers.

a. Index registers one and four are used by First Pass and must not be altered.

b. Index register two and three are used by the First Pass but may be altered by FPOC.

c. Index register six and seven are not used by the First Pass and are free for FPOC.

d. Index register five will be used by the First Pass but will not be altered.

7.4.3 Last Pass Own Code.

7.4.3.1 BEGIN

The Last Pass will execute a J YSTRT. This label will appear on the first instruction of LPOC initialize path. Control will be transferred to this location (YSTRT) before any items are delivered to LPOC and after all input tapes to Last Pass have been opened.

7.4.3.2 Obtaining an Item.

LPOC executes a JR YGET. Index register one will contain the absolute address (MSD) of the selected item.

7.4.3.3 No More Items.

Last Pass will execute a J YEND. This label will appear on the first instruction of the LPOC terminate section. Control will be transferred to this location (YEND) when no more items remain.

7.4.3.4 Index Registers.

a. Index registers one, two, and three are used by Last Pass but may be altered by LPOC.

b. Index registers four, five, six, and seven are not used by the Last Pass and are free for LPOC.

7.5 A typical input to Sort generation would appear as follows:

Remington Rand **UNIVAC**

UNIVAC 1050

PAL ASSEMBLER CODING FORM

Program SORT TRANSACTIONS

Programmer J. Globule Date 8/7/64

Page 001

Program-ID SIRØØ

For BEGIN only

Sequence Page	Line	Ins	Label	Operation	Operands	Comments
001	01		SIRØØ1	BEGIN	5 RELATIVE PROGRAM	
	02			*STD	STANDARD TAG DEFINITIONS	
	03		SIØRTI/1	3,B,B,D	3 KEYS,2 BYWARY,/ DECIMAL,ASCENDING SIØRT	
	04		MINBS	2	2 ITEMS PER BLØCK IS MINIMUM	
	05		MEMRY	NØNE	SIØRT IS TØ USE ALL AVAILABLE MEMØRY	
	06		TAPES	3A	3A SERIVØS	
	07		SERIVØ	/,Ø,2,3	4 SERIVØS USED BY THE SIØRT	
	08		LPSERI	/	LAST PASS ØWN CØDE SERIVØ IS ØNE	
	09		FPSERI	Ø	FIRST PASS ØWN CØDE SERIVØ IS ZERØ	
	10		KEY1	3,1/7	FIRST KEY	
	11		KEY2	4,24	SECØND KEY	
	12		KEY3	12,49	THIRD KEY	
	13		ITEM	1ØØ	RECØRD SIZE ØF 1ØØ CHARACTERS	
	14		SØRT2			
	15		SØRT3	4,,SEQCK	4 SERIVØS IN SERIVØ CALL,NØ RERUN,INCLUDE	
	16	•			PØSITIVE SEQUENCE CHECK	
	17	•	LAST	PASS	ØWN CØDE,TAPE ØUTPUT ØN SERIVØ ØNE	
	18		LPØC	AREA	1ØØ8 ØUTPUT AREA 1Ø ITEMS PER BLØCK	
	19	•	CALL	ØUTPUT	NØ LIMITING - TAPE3	
	20		LPØUT	TAPE3	'ØUTPUT',ØØØ4,A,5,,,,LPØC,,AR,,1ØØ,2,1ØØ8	

Form. 275 20

Remington Rand **UNIVAC**

UNIVAC 1050

PAL ASSEMBLER CODING FORM

Program SORT TRANSACTIONS

Programmer J. Globole Date 8/9/64

Page 1 3: 002

Program-ID 75 80: SRT01

For BEGIN only

Sequence Page 1 3	Line 4 5	Ins 6	Label 7 11	Operation 13 18	Operands 19 30 40	Comments 50 60 70
002	01		YSTRT	OPEN3	LPOUT OPEN OUTPUT FILE	
	02		LPC1	SA1	DST,4	STORE ADDRESS OF ITEM OUTPUT AREA IN
	03		*		DESTINATION TEXT	
	04		JR	YGET	GET ONE ITEM - ITEM ADDRESS IN X1	
	05		TFR	0,,X1	TRANSFER ITEM TO OUTPUT	
	06		PUT3	LPOUT	WRITE ITEM ROUTINE	
	07		J	LPC1		
	08		YEND	CLOSE3	LPOUT,LOCK NO MORE ITEMS, CLOSE OUTPUT FILE,	
	09		*		REWIND WITH INTERLOCK	
	10		JR	07316	STOP SEQUENCE OF CORDINATOR	
	11		JD	07777777	FINAL STOP	
	12		JR	07000	PROGRAM RELEASE	
	13		SORT4		CALL LAST PASS	
	14		* FIRST	PASS	OWM CODE, INPUT OM TAPE	
	15		FPOC	AREA	10008	INPUT AREA, 10 ITEMS PER BLOCK
	16		* CALL	INPUT	ROUTINE-TAPE2	
	17		FPIN	TAPE2	'INPUT',000,A,4,0,0,FPOC,AR,END,100,2,1008	
	18		ZBEGN	JR	ZSTRT	INITIALIZE SORT
	19			OPEN2	FPIN	OPEN INPUT FILE
	20		FP1	SA1	X5,4	STORE ADDRESS OF FIRST ITEM

Form. 275 zo

Remington Rand **UNIVAC**

UNIVAC 1050

PAL ASSEMBLER CODING FORM

Program SORT TRANSACTIONS

Page 003

Programmer J. Globule Date 8/4/64

Program-ID SRØØ1

For BEGIN only

Sequence Page	Sequence Line	Ins	Label	Operation	Operands	Comments
1 3	4 5	6	7 11	13 18	19 30 40 50 60 70	
	01			JR	ZPUT	DELIVER ONE ITEM TO THE SORT
	02			GET2	FPIN	GET ONE ITEM FROM INPUT FILE
	03			J	FP1	
	04		END	J	ZEND	NO MORE ITEMS
	05			SORT5		CALL DISPERSION PASS
	06			END	ZENT	
	07					
	08					
	09					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					

Form. 275 20

8.0 Glossary

Block Size	Number of characters per tape block.
Field	A section within a key which is used as a sub-ordering relationship between records.
First Pass	Dispersion Pass.
FPOC	First Pass Own Code.
Item	See record.
LPOC	Last Pass Own Code.
Key	An area within a record or item which is used as the ordering relationship between records.
Operating System	Performs program locating and loading functions, unit coordination, program switching (when desired), and tape instruction issues.
Record	All the information regarding one individual or item pertinent to a given problem or set of problems. Sometimes called an item.
String	An ordered sequence of data.
Tournament Size	Number of items or records contained in the tournament area.

SORT STOPS DISPLAYS

1. Parameter Load.

STOP	REASON	ACTION
031200	Cannot find a servo to delete.	Press run to continue. Will return to 031300 Stop.
031201	Less than 3 servos defined in |SERVO| call.	Reassemble
031202	Too many FPOC servos defined.	Reassemble
031203	Too many LPOC servos defined.	Reassemble
031204	Illegal servo combination.	Reassemble
031205	Less servos defined in |SORT3| call than defined in |SERVO| call.	Reassemble
031300	Initial Sort Stop.	a. Set trance mode to PROC. Set trace switch equal 01 to continue Sort, press run. b. Set trace mode to PROC. Set trace switch equal 02 to delete one servo from Sort run, press run.
031301	Stop enables user to reset trace switches.	Press run to continue
031400	FPOC too large.	Reassemble
031401	Designated |MINBS| is too large.	a. Check MINBS Call and MEMRY Call and reassemble. b. If running concurrently, check BEGIN card, or run as single program.
031402	LPOC too large.	Reassemble

STOP	REASON	ACTION
0315SS	Sort has deleted one servo-servo S.	Press run to continue.
031601	Segment missing from instruction tape.	Reload or reassemble.
031602	Instruction tape I.D. no good.	Reassemble
031700	Invalid MINBS statement.	Reassemble
031702	Sort I.D. from BEGIN card is too large.	Correct I.D. on BEGIN card must be in format AABB where $00 \leq BB \leq 90$. Reassemble.

2. First Copy Pass.

0321SS	EOT window on servo SS.	Restart with longer blanks.
0326SS	Segment missing on servo S.	Try restarting, otherwise reassemble.
0327SS	Block count error.	Restart

3. Collation Phase.

033044	Data sequence error.	Restart (rerun will be available at a later date).
0336SS	End of tape detected on servo 'SS'.	Reduce volume of data and restart or mount longer tapes and restart (rerun will be available at a later date.)
0332SS	Label block missing on servo 'SS'.	Restart (rerun will be available at a later date.)
0337SS	Block count error on servo 'SS'.	Restart (rerun will be available at a later date).

4. Last Pass.

STOP	REASON	ACTION
0332SS	See above.	
0337SS	See above.	

5. Dispersion Pass.

035001	Item count limit reached.	Press run to continue, will only stop once.
0356SS	End of tape window on servo S.	Get longer tape and restart.

6. General.

Upon depressing the run button at any unrecoverable stop, control will be transferred to the Operating System. This will allow a second program to continue if running concurrently, or will allow operator request button to be depressed for purposes of obtaining a memory dump. See Operating System for details.

4K Load Routine (With memory fill)

1.0 Introduction.

The load routine for the column serial reader fills memory with a specified character (□) and loads the program which follows it in the card reader. It performs its functions in the following order:

a. Establish the interrupt entries for the Class I, Class II and card reader interrupts. The other interrupt entries are destroyed when loading the load routine and if used, must be initialized by the program being loaded.

b. Fill memory except the tetrad area with the character □ (077).

c. Fill the entire tetrad area with the character □ (077).

d. Load the load routine itself into consecutive locations starting immediately after the read area (see E. below).

e. Load the program itself, reading the cards into the area starting at 0600 (octal).

The only locations which cannot be loaded using the load routine are:

a. Tetrads 7, 8, 16, 18, 19 and 36.

b. The read interrupt entry.

c. The area occupied by the routine itself (108 characters for 90 column, 103 for 80 column.

d. The area used by the loader to read cards (109 characters for 90 column cards, 160 characters for 80 column cards.)

2.0 Operating Instructions.

2.1 To load a program using this routine

a. Place the six cards of the load routine in the reader followed by the program to be loaded. The first card of the program to be loaded has an R in column 74. (84 for 90 column cards).

b. Press "Clear", "Load Card", "Start", "Continuous", and "Start". The load routine will fill memory, load the program, and transfer control to the program loaded.

c. If an error occurs during the reading of the load routine, the computer will stop, with 30 0XX XXX 60 in the instruction register where XXXXX is an address in the area used for reading by the load routine. Begin the load operation over again at step a.

d. If an error occurs during the loading of object cards, the computer will stop with 30 1100000 60 in the instruction register. This is caused either by an error condition in the reader or by a failure of the check sum. Ready the reader to refeed the last card fed and the one in the read station. Depress Ready (at the reader) and Start.

2.2 The load routine may be operated with neither memory fill nor tetrad clear. To do so, remove the second and third cards from the deck. Operating instructions for the resulting 3-card deck are the same as above.

2.3 The load routine may be operated with tetrad clear but no memory fill. To do so remove the second card from the deck. Operating instructions for the resulting 4-card deck are the same as above.

2.4 The load routine may be operated without the check sum feature. To do so remove the fifth card from the deck. This reduces the program space required for the load routine by 30 characters for 90 column routine, 35 characters for the 80 column routine.

3.0 Options Available Through Reassembly:

At the beginning of the source code deck for the loaders are six 'EQU' cards. These cards define the labels PGM, REA, LMT, CHR. By altering definitions for these labels the routine may be changed as follows:

3.1 To change the locations in which the load routine is stored, define the label PGM to be the address of the first location the load routine is to occupy. Thus, if it were desired to have the load routine occupy locations 4000 to 4063, the card defining PGM would be replaced with the card:

```
PGM    EQU    4000
```

3.2 To change the area into which the load routine reads cards while loading, replace the card defining REA with a new definition of REA. The address supplied in this definition must be a multiple of 128 (or 200 if it is expressed in octal notation).

3.3 To change the character with which memory is filled, replace the definition of CHR. If the character itself is written, it must be surrounded with quotes. Thus the present definition of CHR might be written in any one of the following three ways:

```
CHR    EQU   '□'
CHR    EQU   63
CHR    EQU   077
```

3.4 The tetrad area may be cleared to blanks by changing line 00310 to read

```
PD   7,8
```

4K Input-Output Routines.

1.0 General.

These routines are in the form of source code. Each routine begins with a comment card containing the name of the routine in columns 13 to 18. Since there are variations possible to severla of the input-output routines, it is recommended that a master copy be maintained of each of these routines and a copy reproduced from the master for inclusion with each program.

The routines provided are:

*RD9L	Read 90-column cards with lockout.
*RDTL	Read 80-column cards translated with lockout.
*RD9	Read 90-column cards with overlapped processing.
*RDT	Read 80-column cards, translate, and overlap processing.
*PH9L	Punch 90-column cards with lockout.
*PHTL	Punch 80-column cards, translated, with lockout.
*PH9	Punch 90-column cards with overlapped processing.
*PHT	Punch 80-column cards, translated, with overlapped processing.
*PR	Print a 128-character line (for buffered or unbuffered systems).
*PRX	Print a 132-character line (for buffered printer only).
*PRL	Print a 128-character line without overlapped processing (primarily for unbuffered printer).

Each routine has at least the three entrances for the functions initialize, execute, and close. Each routine addresses an area whose name is standard and entrance to the execute section causes the routine to place in a fixed index register the relative address of the next available area. This address is relative to the beginning of the standard area. Thus *RDT which reads into the area labelled XAR will deliver the card image to the calling routine in the 80 locations starting at XAR + 80.

It will place 80 in index register 1.

The following table summarizes certain characteristics of these routines. For a more detailed discussion of the use of these routines see sections 3.0 to 5.0.

Routine Name	Index Register Used	Name of Entrance to Initialize	Execute	Close	Size	Area Name	Area Size	Total Space
RD9L	1	XIR	XXR	XCR	100	XAR	109	209
RDTL	1	"	"	"	85	"	80	160
RD9	1	"	"	"	148	"	199	347
RDT	1	"	"	"	143	"	160	303
PH9L	2	XIH	XXH	XCH	190	XAH	237	427
PHTL	2	"	"	"	155	"	208	363
PH9	2	"	"	"	215	"	327	542
PHT	2	"	"	"	200	"	288	488
PR	3	XIP	XXP	XCP	160*	XAP	256	416
PRX	3	"	"	"	160*	"	264	424
PRL	3	"	"	"	135*	"	128	263

*Does not include the space required by XOP and XRP.

Programs are to address the image area for each routine using the appropriate index register as shown in the table above. Thus to bring columns 3 to 5 of an image read by *RDT to AR1, the instruction would be written

```
BA1   XAR + 4, 3, 1
```

The input to the assembler for a program using these routines must include

1. The reservation of the area(s) for the routine(s) used.

2. A copy of the source code for the routines used.

3. The source code for the program itself.

The reservation of the area for one of these routines must precede the source code for the routine IN THE INPUT TO THE ASSEMBLY. The address of the first location of the area must be a multiple of 64 for

a. XAP if the printer is not buffered.

b. XAR for use with *RDT and *RDTL

c. XAH for use with *PHT and *PHTL

The address must be a multiple of 128 for

a. XAR used with *RD9 and *RD9L

b. XAH used with *PH9 and *PH9L

One easy way to ensure this is to use an appropriate origin statement:

```
      ORIG   $,128
XAR   AREA   199
```

The standard loader and memory dump for the 4K systems assume that the print area will start in memory location 384 (0600 octal) and be followed by the read or punch area. Each program should have at least 245 locations (199 for 90-col. systems) reserved for input-output areas starting at this location if the standard loader and memory dump are to be used.

In each case when one of the functions performed by these routines is desired, it is obtained by executing a 'JR' instruction addressing the tag associated with the section performing that function. For example, to initialize the card reader routine one would write the instruction

```
JR   XIR
```

2.0 Compatibility.

These routines are designed to be used in a manner analogous to the corresponding routines for UNIVAC 1050 Systems with a larger memory, including those routines designated for use with the Coordinator. They are constructed in such a manner as to allow programs written using them to be reassembled with a minimum of alteration and run in the environment of an expanded system.

To convert a program using these routines to one using routines written for a larger configuration the following steps are required:

a. Replace the area reservation for each routine to be replaced with one that is appropriate to the new routine that is to replace it.

b. Remove from the deck the source code for the routine(s) being replaced.

c. Insert into the deck (input to the assembly)

1) The appropriate call to the PAL library if the tape assembler is to be used,

or 2) The appropriate output of the I/O Specializer if the card assembler is to be used.

In either case, the call for the routine must specify the same index register used by the routine being replaced. Thus, P3 must be 1 for a reader routine, 2 for a punch routine, and 3 for a printer routine.

d. Insert into the deck a standard set of 'EQU' cards equating the tags of the 4K routines to the corresponding tags of the replacement routines. For example, in moving to a larger system and replacing the reader routine, the following cards would be included in the input to the assembly:

```
XIR    EQU    XINRD

XXR    EQU    XCTRD

XCR    EQU    XCLRD
```

3.0 Use of Card Reader Routine.

3.1 Initialize (XIR)

The initialize section must be entered before there is any attempt to get a card image. Base address tetrad (tetrad 36) and the channel interrupt entry are set to their appropriate values. A feed card order is issued except in the case of the routines with lockout (*RD9L and *RDTL).

3.2 Execute (XXR)

This section is entered when the worker program wants a new card image. A feed card instruction will be issued and the base address of the area available to the worker program is placed in index register 1. The card image in the case of 90-column reader will appear in 90 consecutive locations in untranslated form.

3.3 Close Out (XCR)

For compatibility with the routines in large systems a close out section is provided.

3.4 Interrupt.

The interrupt section is automatically entered by the hardware in the case of an error condition. The computer will be brought to a stop with the following in the instruction register:

30 110000 60

If there is a card in the error stacker, it must be replaced in the input hopper, followed by the card in the read station and the remaining cards in the input hopper. Then to resume processing from the point of the error, depress Ready and Start.

4.0 Use of the Punch Routine.

4.1 Initialize (XIH)

XIH must be entered before there is any attempt to edit data to be punched. The area available to the worker program is cleared to spaces. The base address (relative to the address of XAH) of the area available to the worker program is placed in index register 2. All counters and variable connectors are set to their initial conditions. The channel entry is not affected at this time, having been established correctly in loading the program.

4.2 Execute (XXH)

XXH must be entered when the worker program has finished the editing of data and wants it to be punched. The punch instruction is issued and the base address of the area available to the worker program is placed in index register 2. The arithmetic registers and tetrads 16 to 19 are destroyed.

4.3 Close Out (XCH)

XCH is entered after the last output data card has been delivered to the punch through entry to XXH. A feed card is issued to send the last valid card into the output stacker.

If the last valid card punched before closing causes a read check error, it will be properly repunched. However, in this case, the card remaining in the punch unit will not be a blank card as is customary. Unless this card is cleared from the punch unit it may cause a punch error when the punch is next used.

4.4 Interrupt.

The interrupt section is automatically entered upon completion of a punch instruction. It returns control to the point of interruption after its work has been completed. In the case of an error the computer will be brought to a stop with the following in the instruction register:

30 120000 60

Depress Ready and Start buttons. The computer will repunch the error card(s) and continue with the normal processing.

5.0 Use of the Printer Routine.

5.1 Initialize (XIP)

XIP must be entered before there is any attempt to edit data to be printed. All areas are cleared to spaces. Counters and variable connectors are set to their initial value. The base address of the first working area is placed in index register 3. The print interrupt entry is established while loading the program and is not affected by XIP.

5.2 Execute (XXP)

XXP is entered when the worker program has finished the editing of data and the data is to be printed. XVP, a single character, must be supplied with the number of lines to be advanced by the worker program before entering XXP. A print instruction will be issued and the base address of the next area available to the worker program is placed in index register 3.

5.3 Close Out (XCP)

XCR must be entered when the worker program wants to be sure that the last print line given to XXP has been printed.

5.4 Interrupt

The interrupt section is entered automatically by the hardware in case of an error. The computer will stop with

30 100000 60

in the instruction register. When the difficulty is corrected, depress the Ready and then the Start button to reprint the line causing the difficulty and resume normal processing.

5.5 Call (XQP)

XQP is entered when the worker program wants a "remote area" to be printed. A "remote area" means a print area which is not included in the print area, XAP. The worker program can place any number of such areas anywhere he wants as far as memory capacity permits, subject to the limitation that, if the printer is not buffered, each such area must begin in an address which is a multiple of 64. XQP can be used to print such things as heading lines, page numbers, and so on from these "remote areas" saving the program the trouble of transferring constants to print areas. Prior to entering XQP, the worker program must supply the location XRP (3 characters) with the base address of the remote area and XVP with the number of lines to be advanced. The size of a remote area must be the same as that of a print area in XAP, that is, 128 characters for use with *PR and *PRL and 132 characters for use with *PRX. XQP transfers control to XCP before issuing the print instruction.

5.6 Advance Paper (XUP)

XUP is entered when the worker program wants the paper to be advanced without printing. The number of lines of advance must be placed in the single location at XVP before entering XUP. XUP transfers control to XCP before issuing the advance instruction.

5.7 Special Notes

a. XQP and XUP need not be included unless their functions are desired. They require 35 and 45 locations respectively. XQP constitutes the cards for page 3 (i.e., card sequence numbers in the range 00300 to 00399) and XUP the cards for page 4. Either may be included in or excluded from the routine independently of the other.

b. The number of tag definitions required in replacing the print routine is greater than either of the others. The following definitions are needed:

```
XXP     EQU     XCTPR
XVP     EQU     XADVC
XIP     EQU     XINPR
XCP     EQU     XCLPR
XQP     EQU     XCTQL
XRP     EQU     XRMAR
XUP     EQU     XCTAD
```

c. Areas available within the defined areas.

Use of these areas is not recommended since it complicates the problem of upward compatibility. 'ORIG' cards included in the assembly for a 4K system to define constants or storage at these positions would have to be removed, allowing the locations so defined to follow in the sequence of locations assigned to the worker program.

The locations not used are as follows:

Routine	Totl. Avail.	#Avail. in one seq.	Address of First Loc.
*RD9	19	19	XAR + 45
*PH9L	38	19 19	XAH + 109 XAH + 173
*PHTL	48	48	XAH + 80
*PH9	57	19 19 19	XAH + 45 XAH + 109 XAH + 173
PHT	48	48	XAH + 80

4K SMALL Dump Routine (SDUMP)

1.0 General.

SDUMP is a routine to print the contents of memory in octal. It operates in a minimum amount of space, and at the same time restores everything it uses except the print area. The routine will be distributed in the form of source code which can be assembled to produce a routine which occupies memory locations 3698 to 4095 and uses the area starting at 384 (octal 0600) from which to print. Since it is assumed that this area is being used by the worker program to contain a print image, the contents of the area are printed before the routine begins printing the rest of memory. Each line of the printout contains (at the left) the 5 digit octal address of the lowest order memory location printed on that line, followed by the octal representation of 32 memory locations in 8 groups of 4 locations each. (Each group of 4 locations being represented by 8 octal digits).

Since various options exist which can be exercised by altering the source code, it is recommended that a master copy of the source code be maintained at each installation. This can then be reproduced to provide a working copy for anyone who desires to produce a modified routine. Any such routine should be clearly labelled to indicate the deviations from the standard.

2.0 Use of SDUMP

2.1 Manual Entrance to SDUMP

To enter SDUMP, manually, execute a transfer of control to location 07167 (octal). (Start of the routine's program area plus 5). The routine will print the contents of memory, and stop with the instruction 30 007174 60 in the instruction register. The M portion, however, will be different if the print routine has already been entered by program as described in section 2.2. Manually execute a jump to the desired point in the program to continue running.

2.2 Programmed Entrance to SDUMP

To enter SDUMP automatically in the program, one writes the instruction JR 3698 (the m portion of this instruction addresses the start of the program area for SDUMP). The contents of memory will be printed and the computer will stop with the instruction 30 0mmmmm 60 in the instruction register, where mmmmm is the address of the instruction to which control is to be returned after the SDUMP is finished. To continue with the program from the point at which it was interrupted, depress M and Start.

2.3 Error Recovery

If an abnormal condition occurs in the printer the program will stall in a loop. To continue with the dump, depress Stop, correct the condition in the printer, and depress Ready on the printer and Start.

2.4 Alphanumeric Printing

Depression of Sense Switch 1 prior to entering SDUMP will cause it to print in alphanumeric rather than ocatl format.

3.0 Reassembly Options

3.1 The upper limit of memory printed is defined by the tag LMT:

```
LMT    EQU    07777
```

The lower limit is always zero. To change the upper limit, replace the above definition (card 00110). The location used as an upper limit must always be one less than a multiple of 32.

3.2 The area from which printing is done is defined by the tag XDA:

```
XDA    EQU    0600
```

This area may be made a separate area if there is room in memory. In this way the print image will not be destroyed in the process of obtaining a printout of memory. To do so the above definition must be replaced (card 00115).

3.3 The area occupied by the print routine may be altered. This is done by changing the definition of XDP:

```
XDP    EQU    3698
```

This definition is found on card 00120.

3.4 Space Reduction

The program for the memory print can occupy fewer locations if some of the tetrad and interrupt entry area can be destroyed at the time that SDUMP operates. The following table summarizes the various limitations that can be imposed, the space reduction that results, and the cards to remove from the source deck to achieve each such reduction:

Restriction	Positions Saved	Card to Remove
AR2 will be destroyed	31	-,-,-,-
AR2 not printed properly 5 loc.		-
Some extraneous characters appear at right of listing	5	-
No alphanumeric dumping	10	-,-
Tetrads 32, 33 will be destroyed	10	-,-
Print interrupt entry is destroyed	10	-,-
Tetrad 19 is destroyed	10	-,-
Total	76	

U1050 DATA TAPE CONVENTIONS.

1.0 Label Blocks.

1.1 The First block of each file will be a Label Block.

1.2 A file must start at the beginning of a tape.

1.3 If a file exceeds one tape, each succeeding tape will contain a Label Block as the first block. Contained within the Label Block will be a number of the tape within the file.

2.0 End-of-File Blocks.

2.1 The last data block of a file will be followed by two end-of-file blocks.

2.2 Where rerun is allowed for, rerun information, bracketed by bypass blocks, may appear between the last data block and the end-of-file blocks.

3.0 End-of-Tape Blocks.

3.1 When a file exceeds one tape, each tape except the last will have the last data block followed by two end-of-tape blocks.

3.2 Where rerun is allowed for, rerun information, bracketed by bypass blocks, may appear between the last data block and the end-of-tape blocks.

4.0 Bypass Blocks.

4.1 When a file includes information required for rerun, the information (block or blocks) will be preceded by a bypass block and followed by a bypass block.

4.2 Two bypass blocks and the rerun information contained between them cannot overlap tapes of a multi-tape file.

5.0 Data Blocks.

5.1 All blocks, excluding those described in 4.0, which appear after a Label Block and before an end-of-reel or en-of-file block are termed data blocks.

6.0 Blocks Modes.

6.1 UNISERVO IIIA: All blocks will be written and read in the four character mode.

6.2 UNISERVO IIIC, IVC, and VIC: All blocks will be written and read in the binary mode.

7.0 Block Sizes.

7.1 All blocks in a file, excluding those contained between bypass blocks will be fixed in size.

7.2 The maximum block size will be 4092 characters.

7.3 The minimum block size for any output file contained in a run when rerun is to be allowed for will be 400 characters.

7.4 The minimum block size will otherwise be limited only by the size of the label block (20 character minimum). Refer to 8.3)

7.5 UNISERVO IIIA: The block size for any file must be a multiple of 4 characters.

8.0 Block Formats.

8.1 All blocks will contain an indication of the block type in the first character.

8.2 All blocks will be counted. All blocks, with the exception of those contained between bypass blocks, will contain a binary block count in the second through fourth characters.

8.3 Label Blocks.

Character

0	Octal 3
1-3	Block Number
4-16	Tape Label
17-19	Reel Number
20-n	Unused

Character 0 will contain an octal 3 which identifies this as a label block.

Characters 1 through 3 will contain the binary block number of this block.

Characters 4 through 16 will contain the tape label.

Characters 17 through 19 will contain the reel number expressed decimally in three characters. Each reel of a multi-reel file must contain the number of the reel within the file.

Characters 20 through n are unused.

8.4 Data Blocks.

Character

0	Octal 4
1-3	Block Number
4-5	True Data Character Count
6-n	Data (and Fill, if necessary)

Character 0 will contain an octal 4 which identifies this as a data block.

Characters 1 through 3 will contain the binary block number of this block relative to the first block on the tape.

Characters 4 through 5 will contain a binary count of the number of data characters in the block.

Characters 6 through n will contain data and fill. For UNISERVO IIIA tapes, one to three characters of fill may be required to make the block size a multiple of 4. More fill may be present in the occasional blocks of data when it is desired to write a block which is not yet full.

8.5 Bypass Blocks.

Character

0	Octal 5
1-3	Block Number
4-n	Rerun Information

Character 0 will contain an octal 5 which identifies this as a bypass block.

Characters 1 through 3 will contain the binary block number of this block relative to the first block on the tape.

Characters 4 through n will contain rerun information.

8.6 End-of-Tape Blocks.

Character

0	Octal 6
1-3	Block Number
4-n	Unused

8.6 (continued)

Character 0 will contain an octal 6 which identifies this block as an end-of-tape block.

Characters 1 through 3 will contain the binary block number of this block relative to the first block on the tape.

Characters 4 through n are unused.

8.7 End-of-File Blocks.

Character	
0	Octal 7
1-3	Block Number
4-n	Unused

Character 0 will contain an octal 7 which identifies this block as an end-of-file block.

Characters 1 through 3 will contain the binary block number of this block relative to the first block on the tape.

Characters 4 through n are unused.

PROGRAMMED MULTIPLY AND DIVIDE

1.0 General

The routines are known as MPN, MPC, and DV and will be provided in PAL JR source code. The routines are entered by means of a JR to N, L, where N represents the name of the routine (MPN, MPC, or DV) and L represents the number of characters of the multiplier or quotient.

With the exception of the following notes (2.0 and 3.0) the programmed multiply and divide routines parallel the hardware with respect to entrance requirements and exit conditions.

It is strongly recommended that a master copy of this source code be maintained at each installation and reproduced to provide a copy for inclusion in each program using multiply or divide.

2.0 Multiply-Considerations

a. MPN and MPC are inseparable.

b. The space requirement for MPN and MPC is 256 characters.

c. Multiplication involving blanks, alphabetics or special characters behaves differently from the hardware.

d. MPN and MPC affect indicators in the same manner as hardware for all legitimate multiplications except in the case of MPC, the KZR indicator (37) will reflect the condition of the combined accumulation in AR1.

3.0 Divide-Considerations

a. The space requirement for DV is 301 characters.

b. Blanks, alphabetics, and special characters cause results different from those produced by hardware.

c. DV may produce settings different from the hardware of KZR indicator (37) and KM indicator (38) if decimal overflow occurs during the operation of the divide subroutine.

d. The DV assumes the presence of the MPN and MPC coding at assembly time in order to share 22 characters of constants. If DV is to be used without MPN and MPC, the space requirement for DV is 323 characters.

4.0 Using the Routines

Each routine is entered using a JR of the form: JR N,L where N is MPN, MPC, or DV and L is the number of characters of the multiplier or quotient; each of the routines is entered using a JR instruction of the form; JR N,L.

4.1 Multiply

4.1.1 Entrance Requirements - MPN and MPC

a. Multiplicand in AR2 with sentinel immediately to the left of the MSD.

b. Multiplier in tetrads 20 and 21 with length specified by the L character of the JR.

c. MPC will use the contents of AR1 as a value to be increased by the product of the multiplication.

4.1.2 Exit Conditions - MPN and MPC

a. The product will occupy all characters of AR1, with zeroes to the left of significant characters. In the case of MPC, the product will be increased by the contents of all AR1 at entrance to the routine.

b. The contents of tetrads 20 and 21 are destroyed for only the L characters of the multiplier.

c. The contents of AR2 are left unchanged.

d. Indicators

(1) Indicators HI (33), LO (36), EQ (34), and NBOF (39) are unchanged from their entrance conditions.

(2) The KZR indicator reflects the condition of AR1. The indicator will be set only if the product or, for MPC, the product plus the initial content of AR1 is zero. This represents a variation from the hardware multiplication result in which KZR is unchanged by MPC.

(3) The KM indicator (38) reflects the condition of AR1. The indicator will be set only if the product is negative.

(4) The KDF indicator (40) will be set if the product, or for MPC, the product plus the initial content of AR1 exceeds 16 characters. The resulting decimal overflow will be inhibited until control is returned to the user program.

4.1.3 To assemble the multiply routine without the divide routine use the cards containing page numbers 1 and 2.

4.2 Divide

4.2.1 Entrance Requirements - DV

a. Divisor in AR2 with sentinel immediately to the left of the MSD.

b. Dividend in AR1.

c. Quotient length specified in L character of JR.

4.2.2 Exit Conditions - DV

a. The quotient will be developed in the L least significant characters of tetrads 20 and 21. Other characters of those tetrads are destroyed.

b. The least significant characters of AR1 will contain the remainder. The number of characters allowed for this purpose is the sum of the number of characters of the divisor and the quotient. Other characters of AR1 are unchanged. The sign of the remainder is the sign of the dividend.

c. The contents of AR2 are unchanged.

d. Indicators

(1) Indicators HI (33), LO (36), EQ (34) and NBOF (39) are unchanged from their entrance conditions.

(2) The KM indicator (38) reflects the condition of the quotient. The indicator will be set if the quotient is negative.

(3) The KZR indicator (37) reflects the condition of the remainder. The indicator will be set if the remainder is zero.

(4) The KDF indicator (40) will be set if the dividend is more than nine (9) times the value of the divisor for any quotient position.

The resulting decimal overflow will be inhibited until control is returned to the user program.

4.2.3 To assemble the divide routine without the multiply routine use the cards containing page numbers 2 and 3.

4.3 Warning

a. None of the operands may include blanks or other special characters.

b. AR2 must contain a sentinel for division else the divide subroutine, like the hardware, will loop.

4.4 Reserved Tags

These routines use the labels MPC, MPN, DV, and MDS. None of these labels may be defined in a program assembled with these subroutines.

MAGNETIC TAPE CONTROL ROUTINES: UNIVAC 1050 CONVENTIONS.

1.0 INTRODUCTION

There are 3 tape-control routines for tapes conforming to the UNIVAC 1050 data-tape conventions: TAPE1, TAPE2, and TAPE3. TAPE1 is a general-purpose tape input-output routine and can be used to control any number of input and output files. TAPE2 can be used to control a single input file and TAPE3 can be used to control a single output file. TAPE2 and TAPE3 offer certain economies in store requirements and execution times for programs, or program segments, which process a single tape file (for example, a card-to-tape, tape-to-print, or the first and last passes of a sort program).

In general, the 3 routines perform the following functions: check or produce label blocks, advance items, determine when blocks should be read or written, maintain block counts, servo swap, submit XF orders to the tape-handler portion of the Operating System, and relinquish control to the Operating System when necessary.

All three routines may be present in the same program if desired.

2.0 TAPE1 CONTROL ROUTINE

2.1 Introduction

There are three program components involved in the use of the TAPE1 routine: a TAPE1 call, a number of FILE1 calls, and the macro-instructions.

The TAPE1 call directs the assembler to incorporate tape input-output coding into the program. The configuration of the incorporated coding depends upon the input-output requirements of the worker program, which are specified by the values of the parameters in the TAPE1 calling statement. In general, the coding generated will perform the following functions: check or produce label blocks, advance items, determine when blocks should be read or written, maintain block counts, servo swap, submit the necessary XF orders to the tape-handler portion of the Operating System, and relinquish control to the Operating System when necessary.

The FILE1 calls describe the files to be controlled, directing the assembler to generate a number of constants and working storages which will be used by the TAPE1 coding. A FILE1 calling statement contains such information as label, input-output method to be applied, servos, block size, record size, and file area(s).

The macro-instructions furnish linkages with the subroutines of the TAPE1 coding. They are used by the worker program to initiate and terminate the processing of files, to establish rerun points, and to obtain and release individual data items.

2.2 TAPE1 Call

The TAPE1 call specifies a number of parameters which describe the tape input-output requirements of the worker program. These parameters are used to generate the input-output subroutines which will perform the required file- and item-handling functions. The TAPE1 calling statement has the following format:

LABEL	OP'N	OPERANDS
	TAPE1	p1,...,pn

The label field must be blank and the operation field as shown above.

p1 Number of input files to be controlled by TAPE1. This parameter is Ø or blank if there are no input files.

p2 Input method to be applied: DMND, STDBY, or BOTH. In the standby method, a file has 2 input areas; therefore, the next block can be read from tape while the current block, in the other area, is being processed. In the demand method, a file has only 1 input area; therefore, a block can be read only when the previous block has been completely processed. Parameter 9 of the FILE1 call determines which method is to be applied to a particular file.

This parameter is blank if there are no input files.

p3 Form of the item-advance macro-instructions that will be used: AR, TRF, or BOTH. (Refer to Section 2.4.2 for a description of these forms.) Parameter 12 of the FILE1 call, and the macro-instructions actually used, determine which form applies to a particular file.

This parameter is blank if there are no input files.

p4 Number of output files to be controlled by TAPE1. This parameter is Ø or blank if there are no output files.

p5 Output method to be applied: DMND, STDBY, or BOTH. In the standby method, a file has 2 output areas; therefore, a block can be written from one area while the next block is being processed in the other area. In the demand method, a file has only 1 output area; therefore, a block cannot be processed until the write order for the previous block has been completed. Parameter 9 of the FILE1 call determines which method is to be applied to a particular file.

This parameter is blank if there are no output files.

p6 Form of the item-advance macro-instructions that will be used: AR, TRF, or BOTH. (Refer to Section 2.4.2 for a description of these forms.) Parameter 12 of the FILE1 call, and the macro-instructions actually used, determine which form applies to a particular file.

This parameter is blank if there are no output files.

p7 Primary control index register: 1 through 7. The index register designated will be used for communication between the TAPE1coding and the macro-instructions. It may be used by the worker program for other purposes, but its contents will be altered whenever a macro-instruction is executed.

p8 Secondary control index register: 1 through 7. The index register designated will be used by the TAPE1 coding. It may be used by the worker program for other purposes, but its contents will be altered whenever a macro-instruction is executed.

The allocation of this index register to the TAPE1 coding will result in a saving of storage locations and execution time, but may be omitted if the user wishes. In this case, parameter 8 is Ø or blank.

p9 If rerun is to be allowed for, this parameter designates the output file on which rerun points are to be established; otherwise, it is blank. Rerun points are established by use of the RERN1 macro-instruction.

p10 Sentinel option: OPSEN if sentinel option is desired, blank otherwise. If the sentinel option is chosen, the program will stop whenever an end-of-file or end-of-reel sentinel is read. By means of a trace-switch setting, the operator will direct the TAPE1 coding to perform either end-of-file or end-of-reel processing for the file being read.

p11...To be assigned.

2.3 FILE1 Call

Immediately preceding the TAPE1 call, there is a FILE1 call for each file to be controlled by TAPE1. It specifies 15 parameters which describe the file and which are used to generate constants and working storages required by the TAPE1 coding. A FILE1 calling statement has the following format:

LABEL	OP'N	OPERANDS
file ID	FILE1	p1,...,p15

The label field contains a unique 1- to 5-character label which will be used to identify the file in the macro-instructions.

The operation field must be as shown above.

p1 File type: IN for an input, or OUT for an output.

p2 Tape label: 13 or less characters bounded by apostrophes.

p3 Reel number base: 3 decimal digits bounded by apostrophes. This value plus decimal 1 will be the reel number of the first reel.

p4 Tape type and recording density[1]: A, for UNISERVO IIIA tapes; B, for compatible tapes at 200 BPI; C, for compatible tapes at 556 BPI; or D, for compatible tapes at 800 BPI.

p5 Channel: normally 4 for an input file, or 5 for an output file.

p6 First servo number.

p7 Second servo number, if servo swap for alternate reels is desired; otherwise, this parameter is equal to p6.

p8 Label of an area large enough to contain one block of the file. (Refer to p15.) An AREA directive for this area must appear in the worker program.

p9 If the standby method is to be applied to the file, this parameter specifies a second area large enough to contain one block of the file; otherwise, this parameter is blank.

---

1. Translation mode is not specified for compatible tapes because they are always read or written in the binary mode. Refer to UNIVAC 1050 SYSTEM DATA TAPE CONVENTIONS, section 6.2.

p10 Label of a closed subroutine which is to be executed in addition to the standard label processing. For an input file, the subroutine is executed after the label block is read, but before it is checked. For an output file, the subroutine is executed after the label block is assembled in an output area, but before it is written onto tape. For both input and output files, the 4 LSC of AR1 contain the absolute address of the label block when the subroutine is entered.

If there is no label subroutine to be executed for the file, this parameter is blank.

p11 For an input file, this parameter specifies a label in the worker program to which control will be transferred when an end-of-file block is read (subject to the sentinel option described under p10 of the TAPE1 call).

For an output file, this parameter determines what the TAPE1 coding will do if an end-of-tape condition is detected while a PUT1 macro-instruction is being executed.

If this parameter is blank, the TAPE1 coding will close the current reel and open the next, returning control from the PUT1 in the normal fashion. (Refer to Exit Conditions in Close Output Reel, which constitutes a detailed description of the end-of-tape actions performed by the TAPE1 coding.)

If this parameter is not blank, the TAPE1 coding will transfer control to the specified label*. The worker program may then perform any desired end-of-reel processing, such as putting out summary items or hash totals, or establishing a rerun point. This processing must be followed by a close reel macro-instruction, after which normal processing may be resumed.

If there is more than one PUT1 macro-instruction which addresses the file, control may be returned to the proper point by a jump to the exit line of the appropriate PUT1 subroutine. If the AR form of the macro-instructions is used, the exit line is labelled XTP6Ø; if the TRF form is used, the exit line is labelled XTP62. It should be noted, however, that the execution of one or more PUT1 macro-instructions in the end-of-reel processing will alter the exit line. In this case, the worker program must save XTP6Ø+1 through XTP6Ø+3, or XTP62+1 through XTP62+3, before the macro-instructions are executed.

* This transfer of control will take place only once per reel.

pl2 Form of the macro-instructions: AR, if the arithmetic-register form will be used; TRF, if the transfer form will be used.

pl3 Item size. If pl2 is TRF, this parameter cannot be greater than 1024.

pl4 Number of fill characters (refer to pl5).

pl5 Physical block size, which equals: (pl3) times (number of items per block) plus (pl4) plus (6).

For UNISERVO IIIA tapes, this parameter must be a multiple of 4.

2.4 MACRO INSTRUCTIONS

2.4.1 General

The worker program communicates with the TAPE1 coding by means of the macro-instructions described below. Format, entrance requirements, exit conditions, and memory requirements are given for each macro-instruction.

In addition to the specific exit conditions given for each macro-instruction, it should be noted that all macro instructions alter the contents of arithmetic registers 1 and 2, the primary control index register, and the secondary control index register, if any.

An entry in the LABEL field of a macro-instruction applies to the first instruction generated.

2.4.2 Item Handling Macro-Instructions

There are two forms of the item handling macro-instructions: an arithmetic register (AR) form, and a transfer (TRF) form. The AR form supplies the worker program with the absolute address of the first character of the current item (or item area, if the file is an output). The worker program places this value in an index register. All subsequent references to the item use the index register and are item-relative (that is, the first character is addressed as Ø, the second as 1, and so on). Thus an instruction to bring the tenth through the fifteenth characters of the item to AR1 could be written BA1 14, 6, x; where x is the index register which will contain the item address.

If the programmer wants to assign labels to the item and its constituent fields, he may do so by defining the structure of the item with a dummy AREA statement. The AREA statement and its associated field definitions are written in the normal fashion (refer to the Reference Manual, Section 4-C, Pages 103 through 105). They are bracketed by statements which manipulate the location counter (refer to the Reference Manual, Section 4-c, Page 15), preventing the allocation of memory to the item. The following example shows how an item might be defined.

LINE	LABEL	OP'N	OPERANDS
1	label-a	EQU	$
2		ORIG	Ø
3	label-b	AREA	item size, type,,index register
4	FLD1	-	6, 15
•	•	•	•
•	•	•	•
•	•	•	•
j	field-n	-	length, address
j+1		ORIG	label-a

Line 1 establishes a reference point to which the location counter may be reset after the item has been defined. Line 2 sets the location counter to Ø so that the item and field addresses will be Ø-relative. Lines 3 through j define the item and its constituent fields. (Note that the AREA statement cannot specify a fill character.) Line j+1 resets the location counter to the value it contained prior to the item definition.

Using such an item definition, the tenth through the fifteenth characters of the item could be brought to AR1 by the instruction BAl FLD1, which would be equivalent to the instruction BAl 14, 6, x, where x is the index register which will contain the item address.

The transfer (TRF) form transfers items between the file area and some other area in the program. If this other area is a working storage, it and its constituent fields are mapped by a normal AREA statement. The other area may also be one which is associated with a file using the AR form of the macro-instructions. In this case, the macro-instructions address the current item area of the second file by referencing the label assigned to the item in the dummy AREA statement. For example, an item common to both input file A and output file B, and using index register 3 (X3) might be labelled ITMAB. It could then be processed as outlined below.

```
OPEN1 A          Checks A label.  Places 1st item address in  AR1.
BT    AR1,X3     Places 1st item address in  X3.
OPEN1 B          Writes B label

Process item.    Item and component fields addressed through X3

PUT1 ITMAB,B     Transfer item from A area to B area
GET1 A           Places next item address in  AR1
BT   AR1,X3      Places next item address in  X3
```

Return to item-processing. At end of file A, close file B

```
CLOS1 B, RWD     Write out any remaining items.  Write 2 EOF
                 blocks.  Rewind current reel of B.
```

As part of this processing, items could also be transferred from working storage to file B. This would be accomplished by a line of the form:

```
PUT1 ws,B
```

where ws is a label assigned to the first character of a working storage area.

It should be noted that the AREA statements described in the preceding paragraphs pertain only to items, and should be distinguished from the AREA statement or statements which allocate the input or output area(s) required by a file. (Refer to parameters 8 and 9 of the FILE1 call.)

2.4.3 File-Handling Macro-Instructions

The file-handling macro-instructions, with the exception of RERN1, reduce to 3 basic macro-instructions: open file, close file, and close reel. There is no open reel macro-instruction because the functions of such a macro-instruction are implied in, and performed by, the close reel macro-instruction.

The open file macro-instruction initiates the processing of a file. It sets the constants and working storages pertaining to the file to their initial conditions. It also checks or writes a label block and, except in the case of an output file using the TRF form of the item advance, presents the first data item, or its address, to the worker program. The open file macro-instruction may be executed when: (1) no previous macro-instructions have been executed for the file, or (2) the file has been previously opened and was automatically closed at end of file, or (3) the file has been previously opened and was closed by means of a close file macro-instruction.

The close file macro-instruction terminates the processing of a file. In the case of an output file, it writes any remaining items onto tape, together with 2 end-of-file blocks. It also rewinds the tape as indicated in the macro-instruction. In the case of an input file, the macro-instruction simply rewinds the tape as indicated. In both cases, control returns to the worker program when all orders for the file have been successfully completed. A close file macro-instruction should not be executed for an input file if the file has been closed automatically, since the tape has already been rewound.

The close reel macro-instruction initiates, at the worker program's request, the actions which are usually performed automatically at the end of a reel. It is used when the worker program wishes to close a reel prior to the detection of an end-of-tape condition (output)[1] or an end-of-reel sentinel (input). The macro-instruction terminates the processing of the current reel and initiates the processing of the next reel. In the case of an output file, it writes any remaining items onto tape, together with 2 end-of-reel blocks, rewinds the current reel as indicated in the macro-instruction, and writes a label block on the next reel. In the case of an input file, the macro-instruction simply rewinds the current reel and checks the label block of the next reel. In all cases, except that of an output file using the TRF form of the item advance, the macro-instruction presents the first data item, or its address, to the worker program.

---

1. Refer to parameter 11 of the FILE1 call for an exception to this.

2.4.4 Open Input File

LABEL	OP'N	OPERANDS
label	OPEN1	file ID
label	OPEN1	file ID, destination

Format

1. Destination is specified if TRF macro-instructions were called for in the FILE1 line. It is either the label of a working storage area, or the label of an output item.

Entrance Requirements

1. This is the first macro-instruction executed for the file, or the file has been closed.

2. If destination is the label of an item, the output file concerned is open and the index register assigned to destination contains the address of the first character of the current item area.

Exit Conditions

1. The label block has been read from the servo specified by parameter 6 of the FILE1 call and has been checked.

2. If specified in parameter 10 of the FILE1 call, a special label subroutine has been executed.

3. If the AR form is used, the 4 LSC of AR1 contain the absolute address of the first item.

4. If the transfer form is used, the first item has been transferred to the area specified by the label destination.

2.4.5 Input Item Advance (AR)

LABEL	OP'N	OPERANDS
label	GET1	file ID

Entrance Requirements

1. The file is open.

Exit Conditions

1. The absolute address of the first character of the next item is in the 4 LSC of arithmetic register 1.

2. If an end -of-reel was detected, the current reel has been rewound with interlock and the next reel opened. The label has been checked and the worker program's special label processing, if any, has been performed. The absolute address of the first character of the next item is in the 4 LSC of arithmetic register 1.

3. If an end-of-file block is detected, the current reel has been rewound with interlock and control transferred to the label specified in parameter 11 of the FILE1 line. The address of the end of file block is not supplied, and no further macro-instructions may be executed for the file until it is re-opened.

2.4.6 Input Item Advance (Transfer)

LABEL	OP'N	OPERANDS
label	GET1	file ID, destination

Format

1. Destination is either the label of a working storage area, or the label of an output item.

Entrance Requirements

1. The file is open.

2. If destination is the label of an item, the output file with which it is associated is open and its index register contains the address of the first character of the current item area.

Exit Conditions

1. The next item has been transferred to the area specified by destination.

2. If an end-of-reel block was detected, the current reel has been rewound with interlock and the next reel opened. The label has been checked and the worker program's special label processing, if any, has been performed. The next item is in the area specified by destination.

3. If an end-of-file block was detected, the current reel has been rewound with interlock and control transferred to the label specified in parameter 11 of the FILE1 line. The end-of-file block was not transferred, and no further macro-instructions may be executed for the file until it is re-opened.

2.4.7 Close Input File

LABEL	OP'N	OPERANDS
label	CLOS1	file ID, rewind option

Format

1. Rewind option is RWD, for a rewind without interlock, LOCK, for a rewind with interlock, or NORWD, if the current reel is not to be rewound.

Entrance Requirements

1. The file is open.

Exit Conditions

1. The current reel has been rewound as specified.
2. Control is not transferred to the worker program's end-of-file section (parameter 11 of the FILE1 line), but passes to the worker program at a point immediately following the macro-instruction.
3. No further macro-instructions may be executed for the file until it has been re-opened.

2.4.8 Close Input Reel

LABEL	OP'N	OPERANDS
label	CLOS1	file ID, rewind option, REEL
label	CLOS1	file ID, destination, rewind option,REEL

Format

1. Destination is specified if TRF macro-instructions were called for in the FILE1 line. It is the label of either a working storage area, or theitem of an output file.

2. Rewind option is RWD, for a rewind without interlock, LOCK, for a rewind with interlock, or NORWD, if the current reel is not to be rewound.

Entrance Requirements

1. The file is open.

2. If destination is the label of an item, the output file with which it is associated is open and its index register contains the address of the first character of the current item area.

Exit Conditions

1. The current reel has been rewound as specified and the next' reel opened. The label has been checked and the worker program's special label subroutine, if any, has been executed.

2. If destination was specified, the first item of the new reel is in the area specified by destination; otherwise, the absolute address of the first item is in the 4 LSC of AR1.

2.4.9 Open Output File

LABEL	OP'N	OPERANDS
label	OPEN1	file ID

Entrance Requirements

1. This must be the first macro-instruction executed for the file, or the file has been closed.

Exit Conditions

1. The label block has been written on the servo specified by parameter 6 of the FILE1 call.

2. If specified in parameter 10 of the FILE call, a special label subroutine has been executed.

3. If the AR form of the PUT1 macro-instruction is used, the absolute address of the first character of the first item area will be in the 4 LSC of arithmetic register 1.

2.4.10 Output Item Advance(AR)

LABEL	OP'N	OPERANDS
label	PUT1	file ID

Entrance Requirements

1. The file is open.

Exit Conditions

1. The absolute address of the first character of the next item area will be in the 4 LSC of arithmetic register 1. The previous item is not available to the worker program.

2. If an end-of-tape condition was detected, and parameter 11 of the FILE1 call was blank, 2 end-of-reel blocks were written on the current reel. The current reel has been rewound with interlock and a label block was written on the next reel. If specified in parameter 10 of the FILE1 call, a special label subroutine has been executed. The absolute address of the first character of the next item area is in the 4 LSC of arithmetic register 1.

3. If an end-of-tape condition was detected, and parameter 11 of the FILE1 call was not blank, control has been transferred to the specified label. The absolute address of the first character of the next item area is in the 4 LSC of AR1.

2.4.11 Output Item Advance (Transfer)

LABEL	OP'N	OPERANDS
label	PUT1	origin, file ID

Format

1. Origin is either the label of a working storage area, or the label of an input item.

Entrance Requirements

1. The file is open.

2. If origin is an input item, the input file with which it is associated is open and its index register contains the address of the first character of the current item area.

Exit Conditions

1. The item has been transferred to the output area.

2. If an end-of-tape condition was detected, and parameter 11 of the FILE1 call was blank, 2 end-of-reel blocks were written onto the current reel, which has been rewound with interlock. A label block was written on the next reel. If specified in parameter 10 of the FILE1 call, a special label subroutine has been executed.

3. If an end-of-tape condition was detected, and parameter 11 of the FILE1 call was not blank, control has been transferred to the specified label.

2.4.12 Close Output File

LABEL	OP'N	OPERANDS
label	CLOS1	file ID, rewind option

Format

1. Rewind option is RWD, for a rewind without interlock, LOCK, for a rewind with interlock, or NORWD, if the current reel is not to be rewound.

Entrance Requirements

1. The file is open.

Exit Conditions.

1. All items committed to output have been written onto tape. Two end-of-file blocks have also been written.

2. The current reel has been rewound as specified.

3. No further macro-instructions may be executed for the file until it has been re-opened.

2.4.13 Close Output Reel

LABEL	OPEN	OPERANDS
label	CLOS1	file ID, rewind option, REEL

Format

1. Rewind option is RWD, for a rewind without interlock, LOCK, for a rewind with interlock, or NORWD, if the current reel is not to be rewound.

Entrance Requirements

1. The file is open.

Exit Conditions

1. All items committed to output have been written onto tape. Two end-of-reel blocks have also been written.

2. The current reel has been rewound as specified, and the label block has been written on the next reel.

3. If specified in parameter 10 of the FILE1 call, a special label subroutine has been executed.

4. If the AR form of the PUT1 macro-instruction is used, the absolute address of the first character of the first item area will be in the 4 LSC of AR1.

2.4.14 Establish Rerun Point[1].

LABEL	OP'N	OPERANDS
label	RERN1	return

Format

1. Return is the label to which control will be returned when the program is rerun from this point.

Entrance Requirements

1. The file on which the rerun dump is to be written is open.

Exit Conditions

1. All items committed to this file have been written onto tape.
2. A rerun memory dump, bracketed by bypass blocks, has been written.
3. If the AR form of the PUT1 macro-instruction is used, the absolute address of the first character of the next item is in the 4 LSC of arithmetic register 1.

1. This macro-instruction has not been implemented at the present time.

2.5 Estimated Store Requirements

The following paragraphs give estimated store requirements for the TAPE1 coding, the FILE1 calls, and the macro-instructions.

2.5.1 TAPE1 Coding

1. The 3 major sections of TAPE1 coding have the following store requirements:

   Common subroutines, always present 260
   Input subroutines, present if there are input files 350
   Output subroutines, present if there are output files...275

2. If there are input files (p1> Ø), the appropriate values from the following table should be added to the total estimated from paragraph 1, above.

Condition	Description	# of Positions
p8 < 1	Only 1 control index register allocated	54
p10 = OPSEN	Sentinel option selected	35
p2 = STDBY	Standby method used for all inputs	15
p2 = BOTH	Both standby & demand methods used for inputs	25
p4 > Ø	Output files present	30
p3 = AR	All input macro's are AR form	45
p3 = TRF	All input macro's are TRF form	60
p3 = BOTH	Input macro's are both forms	105

3. If there are output files (p4> Ø), the appropriate values from the following table should be added to the total estimated.

Condition	Description	# of Positions
p8 < 1	Only 1 control index register allocated	40
p6 = AR	All output macro's are AR form	50
p6 = TRF	All output macro's are TRF form	55
p6 = BOTH	Output macro's are both forms	105

4. If the standby method is to be applied to any file, add 30 to the total estimated.

2.5.2 FILE1 Call

The number of store positions required for each FILE1 call is 78.

2.5.3 Macro-Instructions

The following table shows the OPERATION and OPERANDS fields for each macro-instruction and the number of store positions required:

MACRO-INSTRUCTION	OP'N	OPERANDS	No. POS.
Open input file: AR	OPEN1	file ID	10
TRF	OPEN1	file ID, destination	20
Input item-advance: AR	GET1	file ID	10
TRF	GET1	file ID, destination	15
Close input file	CLOS1	file ID, rewind option	20
Close input reel: AR	CLOS1	file ID, rewind option, REEL	20
TRF	CLOS1	file ID, destination,rewind option, REEL	30
Open output file	OPEN1	file ID	10
Output item-advance: AR	PUT1	file ID	10
TRF	PUT1	origin, file ID	15
Close output file	CLOS1	file ID, rewind option	20
Close output reel	CLOS1	file ID, rewind option, REEL	20
Establish rerun point[1]	RERN1	return	—

1. This macro-instruction has not been implemented at the present time.

2.6 Estimated Execution Times - Item-Advance Macro-Instructions

Note that these times assume that an end-of-block condition is not encountered during the execution of the macro-instruction.

GET1, AR form: 607.5 usec

TRF form: 1203.0 usec + 9 (item size) usec [+ 13.5 usec, if destination is indexed]

PUT1, AR form: 607.5 usec

TRF form: 1066.5 usec + 9 (item size) usec [+13.5 usec, if origin is indexed]

2.7 Program Stops & Operating Instructions

1. All stops described below are in the format 30 lc uu ss 60, where: c = channel, uu = unit, and ss = stop code.

2. Stop codes 55, 66, and 77 pertain to hardware malfunctions, and occur in the tape-handler portion of the OPS. Refer to the OPS write-up, section IVE, for the appropriate recovery procedures.

STOP CODE	MEANING	OPERATOR ACTION
01	Label error. (AR1) = expected label & reel number (AR2) = actual	a. To try a new tape: 1. Manually rewind erroneous tape. 2. Mount new tape on same servo. 3. Set trace-address switches to other than 01 and trace mode to PROC. 4. Press PROGRAM START. b. To accept erroneous tape: 1. Set trace-address switches to 01 and trace mode to PROC. 2. Press PROGRAM START.
03	Block count error (TØ) = expected flag & block count (T1) = actual	Unrecoverable error. 1. Press PROGRAM START. 2. Program will loop through OPS. 3. Execute prescribed manual jettison procedure.
06* 07*	End-of-reel sentinel End-of-file sentinel	a. To treat sentinel as end-of-reel: 1. Set trace-address switches to 06 and trace mode to PROC. 2. Press PROGRAM START. b. To treat sentinel as end-of-file: 1. Set trace-address switches to other than 06 and trace mode to PROC. 2. Press PROGRAM START.
10	Unidentifiable block. See stop code 03.	Unrecoverable error. See stop code 03.
55	Memory parity error.	Refer to OPS write-up, section IVE.
66	Tape parity error.	
77	Servo off-line or non-ready.	

*This stop will occur only if the sentinel option was chosen in the TAPE1 call.

3.0 TAPE2 CONTROL ROUTINE

The TAPE2 routine controls a single input file, performing all necessary label-checking, block handling, item-handling, and related functions. It is essentially a subset of TAPE1, furnishing certain economies in store requirements and execution time to a program or program segment which has only one tape input file. Two program components are involved in its use: a TAPE2 call, and a set of macro-instructions.

3.1 TAPE2 Call

The TAPE2 call specifies 15 parameters which describe the input file and how it is to be handled. These parameters are used to generate the input subroutines, constants, and working storages necessary to perform the required input functions. The TAPE2 calling statement has the following format:

LABEL	OP'N	OPERANDS
label	TAPE2	p1,...,p15

The label field contains a unique 1- to 5-character label which will identify the file in the macro-instructions.

The operation field must be as shown above.

p1 Tape label: 13, or less, characters bounded by apostrophes.

p2 Reel number base: 3 decimal digits bounded by apostrophes. This value plus decimal 1 will be checked against the reel number of the first reel.

p3 Tape type and recording density[1]: A, for UNISERVO IIIA tapes; B, for compatible tapes at 200 BPI; C, for compatible tapes at 556 BPI; or D, for compatible tapes at 800 BPI.

p4 Channel: 4.

p5 First servo number.

p6 Second servo number, if servo swap for alternate reels is desired; otherwise, this parameter is blank.

1. Translation mode is not specified for compatible tapes because they are always read or written in the binary mode. Refer to UNIVAC 1050 SYSTEM DATA TAPE CONVENTIONS, Section 6.2.

p7 Label of an area large enough to contain one block of the file. (Refer to p14.) An AREA directive for this area must appear in the worker program.

p8 If the standby method is to be applied to the file, this parameter specifies a second area large enough to contain one block of the file; otherwise, it is blank. If an area is specified, an AREA directive for it must appear in the worker program.

p9 Form of the macro-instructions: AR, if the arithmetic-register form is to be used, or TRF, if the transfer form is to be used.

p10 Label of a closed subroutine which is to be performed in addition to the standard label processing, or blank. The subroutine is executed after the label block is read, but before it is checked. The 4 LSC of AR1 contain the absolute address of the label block when the subroutine is entered.

p11 Label in the worker program to which control will be transferred when an end-of-file block is read. (Subject to p15).

p12 Item size. If p9 is TRF, this parameter cannot be greater than 1024.

p13 Number of fill characters. (Refer to p14.)

p14 Physical block size, which equals: (p12) times (number of items per block) plus (p13) plus (6).

For UNISERVO IIIA tapes, this parameter must be a multiple of 4.

p15 Sentinel option: OPSEN, if sentinel option is desired, blank otherwise. If the sentinel option is chosen, the program will stop whenever an end-of-file or end-of-reel sentinel is read. By means of a trace-switch setting, the operator will direct the TAPE2 coding to perform either end-of-file or end-of-reel processing for the file.

3.2 TAPE2 Macro-Instructions

The worker program communicates with the TAPE2 coding by means of the macro-instructions described below. Format, entrance requirements, exit conditions, and store requirements are given for each macro-instruction.

In addition to the specified exit conditions given for each macro-instruction, it should be noted that all macro-instructions alter the contents of arithmetic registers 1 and 2.

An entry in the label field of a macro-instruction applies to the first instruction generated.

3.2.1 Open File

LABEL	OP'N	OPERANDS
label	OPEN2	file ID
label	OPEN2	file ID, destination

Format

1. Destination is specified if TRF macro-instructions were called for in TAPE2. It is the label of an area large enough to contain one item of the file.

Entrance Requirements

1. This must be the first macro-instruction executed for the file.

Exit Conditions

1. The label block has been read from the servo specified by parameter 5 of the TAPE2 call and has been checked.

2. If specified in parameter 10 of the TAPE2 call, a special label subroutine has been performed.

3. If the AR form is used, the 4 LSC of AR1 contain the absolute address of the first data item.

4. If the TRF form is used, the first data item has been transferred to destination.

Store Requirements

5 character positions for the AR form; 15 for the TRF form.

3.2.2 Item Advance

LABEL	OP'N	OPERANDS
label	GET2	file ID
label	GET2	file ID, destination

Format

1. Destination is specified if the TRF form was called for in TAPE2. It is the label of an area large enough to contain one item of the file.

Entrance Requirements

1. The file is open.

Exit Conditions

1. If the AR form is used, the absolute address of the first character of the next item is in the 4 LSC of AR1.

2. If the TRF form is used, the next item has been transferred to destination.

3. If an end-of-reel block was detected, the current reel has been rewound with interlock and the next reel has been opened. The label block has been checked and the worker program's special label processing, if any, has been performed.

4. If an end-of-file block was detected, the current reel has been rewound with interlock and control transferred to the label specified in parameter 11 of the TAPE2 call. The end-of-file block is not available to the worker program, and no further macro-instructions may be executed for the file.

Store Requirements

5 character positions for the AR form; 10 for the TRF form.

3.2.3 Close File

LABEL	OP'N	OPERANDS
label	CLOS2	file ID, rewind option

Format

1. Rewind option is LOCK, for a rewind with interlock, or RWD, for a rewind without interlock.

Entrance Requirements

1. The file is open.

Exit Conditions

1. The current reel has been rewound as specified.
2. Control is not transferred to the worker program's end-of-file section (parameter 11 of the TAPE2 call), but passes to the worker program at a point immediately following the macro-instruction.
3. No further macro-instructions may be executed for the file.

Store Requirements: 10 character positions.

3.3 Estimated Store Requirements - TAPE2 Coding

The following paragraphs give estimated store requirements for the coding generated by the TAPE2 calling statement.

1. The minimum amount of coding that can be turned out will occupy 493 character positions.

2. If the standby method is to be applied (p8 not blank), add 131 to the total estimated.

3. If servo swap is to be performed for alternate reels of the file (p6 not blank), add 80 to the total estimated.

4. If the sentinel option is chosen (p15 = OPSEN), add 35 to the total estimated.

5. If the transfer form of the macro-instructions is to be used (p9 = TRF), add 15 to the total estimated.

6. If special label processing is to be performed (p10 not blank), add 10 to the total estimated.

3.4 Execution Time - GET2 Macro-Instruction

The times shown are calculated on the assumption that an end-of-block condition is not detected during the execution of the macro-instruction.

AR form: 486.0 usec

TRF form: 918.0 usec + 9 (item size) usec + 13.5 usec, if destination is indexed

3.5 Program Stops & Operating Instructions

1. All stops described on the following page are in the format 30 1c uu ss 60, where: c = channel, uu = unit, and ss = stop code.

2. Stop codes 55, 66, and 77 pertain to hardware malfunctions. Refer to the OPS write-up, section IVE, for the appropriate recovery procedures.

STOP CODE	MEANING	OPERATOR ACTION
01	Label error. (AR1) = expected label & reel # (AR2) = actual	a. To try a new tape: 1. Manually rewind erroneous tape. 2. Mount new tape on same servo. 3. Set trace-address switches to other than 01 and trace mode to PROC. 4. Press PROGRAM START. b. To accept erroneous tape: 1. Set trace-address switches to 01 and trace mode to PROC. 2. Press PROGRAM START.
03	Block count error (TØ) = expected flag & block count (T1) = actual	Unrecoverable error. 1. Press PROGRAM START. 2. Program will loop through OPS. 3. Execute prescribed manual jettison procedure.
06*	End-of-reel sentinel.	a. To treat sentinel as end of reel: 1. Set trace-address switches to 06 and trace mode to PROC. 2. Press PROGRAM START. b. To treat sentinel as end of file: 1. Set trace-address switches to other than 06 and trace mode to PROC. 2. Press PROGRAM START.
07*	End-of-file sentinel.	See stop code 06.
10	Unidentifiable block. See stop code 03.	Unrecoverable error. See stop code 03.
55	Memory parity error.	Refer to OPS write-up, section IVE.
66	Tape parity error.	
77	Servo off-line or non-ready.	

* This stop will occur only if the sentinel option was chosen in the TAPE2 call.

4.0 TAPE3 CONTROL ROUTINE

The TAPE3 routine controls a single output file, performing all necessary block handling, item handling, and related functions. It is essentially a subset of TAPE1, furnishing certain economies in store requirements and execution time to a program or program segment which has only one output file. Two program components are involved in its use: a TAPE3 call, and a set of macro-instructions.

4.1 TAPE3 Call

The TAPE3 call specifies 14 parameters which describe the output file and how it is to be handled. These parameters are used to generate the output subroutines, constants, and working storages necessary to perform the required output functions. The TAPE3 calling statement has the following format:

LABEL	OP;N	OPERANDS
label	TAPE3	p1,.....,p14

The label field contains a unique 1- to 5-character label which will identify the file in the macro-instructions.

The operation field must be as shown above.

p1 Tape label: 13 or less characters bounded by apostrophes.

p2 Reel number base: 3 decimal digits bounded by apostrophes. This value plus decimal 1 will be the reel number of the first reel.

p3 Tape type and recording density[1]: A, for UNISERVO IIIA tapes; B, for compatible tapes at 200 BPI; C, for compatible tapes at 556 BPI; or D, for compatible tapes at 800 BPI.

p4 Channel: 5.

p5 First servo number.

p6 Second servo number, if servo swap for alternate reels is desired; otherwise, this parameter is blank.

p7 Label of an area large enough to contain one block of the file. An AREA directive for this area must appear in the worker program.

p8 If the standby method is to be applied to the file, this parameter specifies a second area large enough to contain one block of the file; otherwise, it is blank. If an area is specified, an AREA directive for it must appear in the worker program.

---

1. Translation mode is not specified for compatible tapes because they are always read or written in the binary mode. Refer to UNIVAC 1050 System DATA TAPE CONVENTIONS, Section 6.2.

p9 Form of the macro-instructions: AR, if the arithmetic-register form is to be used, or TRF, if the transfer form is to be used.

p10 Label of a closed subroutine which is to be executed in addition to the standard label processing, or blank. The subroutine is executed after the label block is assembled in an output area, but before it is written onto tape. The 4 LSC of AR1 contain the absolute address of the label block when the subroutine is entered.

p11 This parameter determines what the TAPE3 coding will do when an end-of-tape condition is detected while a PUT3 macro-instruction is being executed.

If this parameter is blank, the TAPE3 coding will close the current reel and open the next, returning control from the PUT3 in the normal fashion. (Refer to Exit Conditions in Close Reel, which constitutes a detailed description of the end-of-tape actions performed by the TAPE3 coding.)

If this parameter is not blank, the TAPE 3 coding will transfer control to the specified label[1]. The worker program may then perform any desired end-of-reel processing, such as putting out summary items or hash totals. This processing must be followed by a close reel macro-instruction, after which normal processing may be resumed.

If there is more than one PUT3 macro-instruction, control may be returned to the proper point by a jump to the exit line of the PUT3 subroutine. If the AR form of the macro-instructions is used, the exit line is labelled XT06Ø; if the TRF form is used, the exit line is labelled XT062. It should be noted, however, that the execution of one or more PUT3 macro-instructions in the end-of-reel processing will alter the exit line. In this case, the worker program must save XT06Ø/1 through XT06Ø/3, or XT062/1 through XT062/3, before the macro-instructions are executed.

p12 Item size. If p9 is TRF, this parameter cannot be greater than 1024.

p13 Number of fill characters. (Refer to p14.)

p14 Physical block size, which equals: (p12) times (number of items per block) plus (p13) plus (6).

For UNISERVO IIIA tapes, this parameter must be a multiple of 4.

1. This transfer of control will take place only once per reel.

4.2 TAPE3 Macro-Instructions

The worker program communicates with the TAPE3 coding by means of the macro-instructions described below. Format, entrance requirements, exit conditions, and store requirements are given for each macro-instruction.

In addition to the specified exit conditions given for each macro-instruction, it should be noted that all macro-instructions alter the contents of arithmetic registers 1 and 2.

An entry in the label field of a macro-instruction refers to the first instruction generated.

4.2.1 Open File

LABEL	OP'N	OPERANDS
label	OPEN3	file ID

Entrance Requirements

1. This must be the first macro-instruction executed for the file.

Exit Conditions

1. The label block has been written on the servo specified by parameter 5 of the TAPE3 call.

2. If specified in parameter 10 of the TAPE3 call, a special label subroutine has been executed.

3. If the AR form of the macro-instructions is used, the 4 LSC of AR1 contain the absolute address of the first character of the first item area.

Store Requirements

5 character positions.

4.2.2 Item Advance

LABEL	OP'N	OPERANDS
label	PUT3	file ID
label	PUT3	origin, file ID

Format

1. Origin is specified if the TRF form was called for in TAPE3. It is the label of an area large enough to contain one item of the file.

Entrance Requirements

1. The file is open.

Exit Conditions

1. If the AR form is used, the absolute address of the next item area is in the 4 LSC of AR1.[1]

2. If the TRF form is used, the item has been transferred from origin to an output area.

3. If an end-of-tape condition was detected, and parameter 11 of the TAPE3 call was blank, 2 end-of-reel blocks were written on the current reel, which has been rewound with interlock. A label block was written on the next reel. If specified in parameter 10 of the TAPE3 call, a special label subroutine has been executed.

4. If an end-of-tape condition was detected, and parameter 11 of the TAPE3 call was not blank, control has been transferred to the specified label.

Store Requirements

5 character positions for the AR form; 10 for the TRF form.

1. This also applies in cases 3 and 4, above.

4.2.3 Close File

LABEL	OP'N	OPERANDS
label	CLOS3	file ID, rewind option

Format

1. Rewind option is RWD, for a rewind without interlock, LOCK, for a rewind with interlock, or NORWD, if the current reel is not to be rewound.

Entrance Requirements

1. The file is open.

Exit Conditions

1. All items committed to output have been written onto tape, together with 2 end-of-file blocks.

2. The current reel has been rewound as specified.

3. No further macro-instructions may be executed for the file.

Store Requirements

15 character positions.

4.2.4 Close Reel

LABEL	OP'N	OPERANDS
label	CLOS3	file ID, rewind option, REEL

Format

1. Rewind option is RWD, for a rewind without interlock, LOCK, for a rewind with interlock, or NORWD, if the current reel is not to be rewound.

Entrance Requirements

1. The file is open.

Exit Conditions

1. All items committed to output have been written onto tape, together with 2 end-of-reel blocks.

2. The current reel has been rewound as specified, and a label block has been written on the next reel.

3. If specified in parameter 10 of the TAPE3 call, a special label subroutine has been executed.

4. If the AR form of the macro-instructions is used, the 4 LSC of AR1 contain the absolute address of the first character of the first item area.

Store Requirements

15 character positions.

4.3 Estimated Store Requirements - TAPE3 Coding

The following paragraphs give estimated store requirements for the TAPE3 coding.

1. The minimum amount of TAPE3 coding that is turned out will occupy 398 character positions.

2. If the standby method is to be applied (p8 not blank), add 141 to the total estimated.

3. If servo swap is to be performed for alternate reels of the file (p6 not blank), add 25 to the total estimated.

4. If the transfer form of the macro-instructions is to be used (p9=TRF), add 10 to the total estimated.

5. If special label processing is to be performed (p10 not blank), add 10 to the total estimated.

6. If normal end-of-tape processing is to be performed (p11 blank), add 30 to the total estimated.

7. If special end-of-tape processing is to be performed (p11 not blank), and the AR form of the macro-instructions is to be used (p9=AR), add 5 to the total estimated.

4.4 Execution Time - PUT3 Macro-Instruction

The times shown are calculated on the assumption that an end-of-block condition is not detected during the execution of the macro-instruction.

AR form: 486.0 usec

TRF form: 949.5 usec + 9 (item size) usec [+13.5 usec, if origin is indexed]

4.5 Program Stops & Operating Instructions

1. There are no program stops in the TAPE3 coding.

2. The following are error stops in the tape-handler portion of the OPS. Refer to the OPS write-up, Section IVE, for recovery procedures.

   a. 30 lc uu 55 60 - memory parity
   b. 30 lc uu 66 60 - tape parity
   c. 30 lc uu 77 60 - servo off-line or non-ready

   In these stops, c = channel, and uu = unit.