PROGRAMMER'S MANUAL FOR THE

SHARE OPERATING SYSTEM

Preliminary Version
October 13, 1959

C. E. Homan, IBM
G. F. Swindle, SDC

TABLE OF CONTENTS

01.01 INTRODUCTION TO THE SYSTEM

The SHARE Operating System, familiarly known as SOS, is a highly flexible complex of languages, procedures and machine codes. The three-fold purpose of the System is to provide assistance to the 709 programmer in the coding and check-out phases of program preparation, to assume from the 709 machine operator those burdens that may be sensibly automated, and to provide the computer installation with an efficient operation and complete and accurate records on machine usage. The System provides for standard formulation of jobs with respect to machine operation, making it necessary for the operator to become familiar only with SOS and not the peculiarities associated with particular jobs. Loss of time between jobs is reduced by the sequential processing feature of the Supervisor. The resultant increase in efficiency offsets the relatively few restrictions imposed upon users of the System.

If the programmer is to make confident and efficient use of SOS, it is essential that he have, in addition to knowledge of the rules for coding and debugging, a broad over-all view of the System and the role which each component plays in the execution of his job.

The initial section of this manual contains a brief discussion of the major subroutines which comprise the SHARE Operating System, i.e., the Com-

piler, Modify and Load, Debugging and the Supervisor. Succeeding sections deal with specific parts of the Supervisory routines, the Buffering routines, the Transmission Macros and the Input and Output Editors. It is presumed that the numerous documents dealing with the computer itself, the rules for SCAT coding and the use of the Debugging Macros are available to the programmer.

01.02 THE COMPILER

Basic to the System is the SCAT language which is employed by the programmer to communicate with the machine. It is a symbolic, machine-oriented language in which the ability to instruct the machine in terms of standard 709 instructions is supplemented by the explicit introduction of commands for generating open subroutines (macro instructions), and special language elements for directing the program assembly process (pseudo-instructions).

For mnemonic quality, programmer convenience and ease of key-punching, the representation of the SCAT language is highly redundant. The function of the Compiler is to reduce the original symbolism to a new form of condensed encoding by computer processing. This intermediate language, called "squoze" encoding, differs from the programmer's encoding in three respects. First, it is fundamentally binary. Second, the information concerning a particular instruction in the original coding appears in a single string; in the ensquozed program particular types of information about the program (e.g., location symbols) are grouped for ease of processing. Third, squoze encoding is far more conservative of bits than the pseudo-English used by the programmer.

The result of the processing of symbolic information by the Compiler is a punched-card representation of the squoze encoding, called the squoze

deck. It is this squoze deck that is used exclusively in subsequent operations with the program. Reference Appendix C for a more detailed description of the squoze deck format.

01.03 MODIFY AND LOAD

The primary purpose of the several files comprising Modify and Load is to interpret and operate upon squoze decks; the functions of the individual files are described in this section. It should be remembered that each file listed below constitutes a separate physical file on the System tape. When a particular file is needed, it is called into core from the System tape, by either the Monitor or another file of Modify and Load and given control. When the file has performed its function, it may pass control to another file of Modify and Load or to the Monitor.

A. M1 is the "bootstrap" of Modify and Load. In addition to reading the preface card and footnotes of the squoze deck, it performs the necessary presetting of indicators and communication region. Having done this, M1 checks for the presence or absence of a modification package in the squoze deck. If there is a "mod package," MO is called in and given control; if not, M4 is the next file to enter and gain control.

B. MO performs essentially the same function as Pass 1 of the Compiler, in that it ensquozes the symbolic modification package and creates the necessary footnote and dictionary entries associated with the modifications. MO calls M3 into core and turns over control to it.

C. M3 converts the ensquozed modification package into a form which is acceptable to M4 and M5 for integration with the original squoze deck. Communication between M3 and subsequent files is accomplished through the communication region established by M1. M3, when done, calls in M4.

D. M4 interprets the squoze dictionary and assigns absolute core address values to the symbols defined therein. If M4 finds an undefined System Symbol in the dictionary (and certainly all System Symbols appearing in the object program will be undefined up to this point), it performs a search of the System Symbol Table in an attempt to assign the current absolute value of that symbol. (See Section 03.01 of this manual for a more complete discussion of the System Symbol Table and its use.) The remaining undefined symbols are stacked in a communication region for later printing by M6. M4 gives way to M5.

E. M5 decodes the squoze text and converts it to absolute binary before writing it on SYSMOT for later loading and execution. If an absolute deck has been requested, it is punched by M5; if a new squoze deck has been called for , text with commentary is punched by this file.

F. M6 prints out error messages describing errors which occurred during the processing of the squoze deck and/or modification package.

G. M7 contains the routines required for punching the preface, introduction, dictionaries, footnotes and text without commentary of a new squoze deck.

H. M8 is the Lister program which interprets the squoze deck and prints a complete listing of the object program (absolute octal, symbolic code, comments, alter numbers, etc.)

I. M9 prints out "catastrophic" error messages (e.g., when Modify and Load has lost control or a possible machine error has been detected.) The only return from M9 is to the Monitor for bypassing the code being processed.

01.04 DEBUGGING

The sequential processing of several jobs through three phases by the System makes it impractical for the operator or programmer to interrupt the execution of a particular program in order to obtain code-checking information. The Debugging macros are included within the System to provide the programmer with a means of procuring the desired data with the original symbolic notation retained whenever possible. The Debugging macros are expanded as calling sequences to System subroutines which take "snapshots" of specified information to be found in memory, on tape or on the drums.

The three types of Debugging Macros which allow the programmer to obtain this debugging output selectively are (1) information, (2) conditional and (3) modal.

1. The information macros specify the location and quantity of information requested by the programmer. These macros are CORE, PANEL, DSC, TAPE, and DRUM

2. The conditional macros allow the programmer to specify the program conditions which must be met before subsequent information macros are executed. These macros are AND, OR, WHEN, EVERY and UNLESS.

3. The modal macros provide the programmer with a reasonable degree of flexibility in specifying debugging output format. These macros are POINT, FORMAT, USE and NUCASE.

01.05 SUPERVISOR

The philosophy of the SHARE Operating System for the 709 is implemented by the Monitor and the method in which it supervises the interaction between various System components. Flexibility is a dominant characteristic of the SHARE 709 System and is exemplified by the various ways in which jobs may use the System. In general, it may be used for input, execution and output.

A. Use of the System for execution implies that the System for a particular job performs several of the following functions:

1. The System analyzes the control cards associated with a job and calls the appropriate System routines to perform the desired operations:

   a. The Compiler to translate the symbolic programs to squoze encoding during the Input phase.

   b. Modify and Load to translate squoze encoding into absolute machine language for execution, to punch an absolute deck, to provide a listing of the program or to punch a new squoze deck during the Input phase.

   c. Debugging

      (1.) SNAP to obtain debugging information during execution of the program.

      (2.) SNAPTRAN during the Output phase to translate the debugging information.

d. The Input Editor for conversion of input during the Input phase

e. INTRAN for use by an object program in the translation of its input data at execution time;

f. The Output Supervisor for transcription of output data at execution time and the Output Editor for conversion and formatting during the Output phase.

g. OUTRAN for use by an object program in the conversion of its output data at execution time.

2. The lower storage trap cells and associated Communication Region entries to standard System routines are available at all times. These may be used in standard fashion or preset to allow entry to a program's special routines. (Section 03.02)

3. The Buffering Routines and the Transmission Macros are permanently in core and may be used by object programs at execution time. Certain of the Execution Coordination Routines are also available for use.

B. Use of the System for input means that during the Input phase a job's input will be translated under system control by the Input Editor and made available to the job at execution time.

C. Use of the System for output means that the information produced by a job and given to the Output Supervisor at execution time will be translated and formatted for off-line presentation by the Output Editor.

The flexibility of the System permits processing of four types of jobs:

1. Type I-E-O jobs in which the translation of input and output data is performed in the appropriate phase by the System. The functions associated with execution are also effected.
2. Type I-E jobs in which the System controls translation of input data and performs the functions associated with execution. The translation of output data is performed by the object program at execution time with or without the use of System routines.
3. Type E-O jobs in which the System performs the normal functions necessary to execution of a job and supervises the translation of output data. The translation of input data is performed by the object program at execution time with or without the use of System routines.
4. Type E jobs in which the System performs the functions associated with execution. Conversion of input and output data is effected by the object program at execution time with or without the use of System routines.

The three basic phases of System operation are designated according to

the function which the System expects to perform in each. In the first (Input) phase, the System effects translation of symbolic programs into squoze encoding, squoze encoding into machine language, and input data into a form acceptable to the object program. The dictionary and code are also written on the mediary tape for use and execution in this or subsequent phases.

In the second (Execution) phase, the System normally performs those functions necessary to loading and executing an object program. This includes presetting the Communication Region; calling System programs requested by the object program such as the Debugging Supervisor, the Input Editor, or the Output Supervisor; and leaving in core the Buffering Routines, Transmission Macros and Execution Coordination Routines.

The third (Output) phase is used by the System for conversion of output data and debugging information.

The phases clearly are named according to the type of operation which the System normally performs in each, but this does not preclude the execution of jobs with attendant input-output translation in any phase.

02.01 GENERAL DESCRIPTION

Control cards provide a compact and flexible method of communication between the programmer, the operator and the System. Two types of control cards enjoy System recognition.

1. "Program" control cards are those whose functions are specific to a particular job. Program control cards are normally furnished by the programmer and enter the System with his job during the Input phase.

2. "System" control cards give information to the System governing the operation of a group of jobs. A System control card normally remains in effect throughout at least one cycle of the System. (A cycle consists of the processing by the System of a group of jobs through Input, Execution and Output phases.) System control cards normally are entered through the on-line card reader by the operator as part of the System initiation procedure.

All control cards have the following common characteristics:

1. Column 1 contains a 7-8-9 combination punch which identifies it as a control card.

2. The operation code of the control card is punched in the normal SCAT format, i.e., beginning in column 8.

3. The variable field must be separated from the operation code by at least one blank column, and must begin no later than column 16, i.e., in SCAT format.

4. If a normal case is defined for a variant within the variable field of a program control card and this normal case reflects the programmer's needs, it is redundant to punch this parameter in the control card. The programmer need specify parameters only in situations which require deviation from the normal case. If the variable field is entirely blank on a program control card of the type that has a normal case defined for all variants, this effectively requests that the System employ the normal case for all variants.

The readers of this document will be concerned primarily with program control cards, but knowledge of System control cards may prove useful in augmenting the programmer's understanding of the System. The following sections describe both classes of control cards.

02.02 SYSTEM CONTROL CARDS

A. The ASSIGN control card.

An ASSIGN control card is employed to advise the System of a change in the status or use of an I/O unit. The System analyzes the variable field of the ASSIGN card, makes the necessary changes in the Communication Region's I/O Unit Control Word Tables, and is governed in its subsequent operation by the indicated status change.

The ASSIGN card may be considered either as a System control card or as a program control card, since its variable field may affect the System's use of an I/O unit for one or more entire cycles, or may apply only during the execution of a single object program.

1. Use of the ASSIGN card as a System Control Record.

ASSIGN XN = Z

where

X is an alphabetic channel designator (A through F)

N is a tape drive number (not used for card equipment)

Z is the assignment desired for the specified I/O unit.

a) Z = OFF to disconnect the unit and make it unavailable for use.

b) Z = ON to place the I/O unit in an "unassigned and available" status

c) Z is of the form SYSXXX referring to the Communication Region control word for the symbolic unit named.

Examples:

a) ASSIGN C = SYSCRD

instructs the System to use the card reader on channel C for on-line input.

b) ASSIGN A5 = OFF

will force the System to avoid the use of tape drive 5 on channel A for any purpose and to print an error message to the operator if an attempt is made to assign A5 for System or object program use.

c) ASSIGN A5 = ON

will return the specified physical unit to available-for-use status.

d) ASSIGN B4 = SYSPOT

will cause tape drive B4 to be used as the Peripheral Output Tape during all subsequent I/O phases until reassignment or System reinitiation.

(Note: Reassignment of System Tapes is permitted only at specific intervals, due to the inter-phase continuity implicit in SOS philosophy. SYSTAP may be reassigned only during initiation, SYSMIT and SYSMOT between cycles, i.e., preceding execution of the input phase, and all other System Tapes between phases. Reassignment of on-line card equipment may be effected at any time between jobs.)

2. Use of the ASSIGN card as a program control card

ASSIGN XN = SYSXZN

where

X = A through F, specifying a symbolic channel

N = 1 through 8, specifying a symbolic tape drive

Z = R for Reserved Tape assignment

Z = U for Utility Tape assignment

The assignment of utility and/or reserved tapes is accomplished by placing the appropriate ASSIGN cards after the JOB card and before the LOAD or SCAT card of the job for which the assignment is to be effected.

Examples:

a) ASSIGN B5 = SYSBR1

assigns unit 5, channel B as symbolic tape SYSBR1 for the job with which this ASSIGN card is associated.

b) ASSIGN C2 = SYSAU1

will assign physical unit 2, channel C as the drive to be utilized in all references to symbolic utility tape A1 during the execution of the object program.

At the conclusion of execution of the object program using the above ASSIGN cards, tape drives B5 and C2 would be rewound by the System, and the operator would be instructed as follows:

REMOVE B5=SYSBR1

REMOVE C2=SYSAU1

B. The DATE control card

A DATE control card results in the presetting of SYSDAT in the Communication Region. This information is available for use by the accounting routines, by the Compiler which places it in 4's left row of the preface card in SQUOZE decks, and by the Lister which places it on each page of listings.

DATE M/D/Y

where M consists of 2 BCI digits specifying the month

D consists of 2 BCI digits specifying the day

Y consists of 2 BCI digits specifying the year

C. The CLOCK control card.

A CLOCK control card results in the setting of the Communication Region cells SYSCLK and SYSCLK+1 with the time and date of the last resetting of the on-line System clock.

CLOCK H.M.S.M/D/Y

where H consists of 2 BCI digits specifying the hour

M consists of 2 BCI digits specifying minutes

S consists of 2 BCI digits specifying seconds

M consists of 2 BCI digits specifying the month

D consists of 2 BCI digits specifying the day

Y consists of 2 BCI digits specifying the year

D. The WST control card.

The WST control card causes the Monitor to turn control over to the System Tape Writer, which copies the old System Tape employing the control cards

and absolute decks from the card reader to delete, insert or replace the specified files. A detailed exposition of the System Tape Writer appears in Appendix C.

E. The GO control card.

The GO control card informs the Supervisor that all System control cards have been read, and causes the System to commence execution of the Input Phase.

GO

No parameters are necessary.

F. The END control card.

The END control card advises the Monitor that it has received all the jobs to be executed in this cycle of operation and may proceed to the Execution Phase.

END

No parameters are necessary.

02.03 "PROGRAM" CONTROL CARDS

A. The JOB control card.

JOB JN,RN,MN,ET,EMO,EPO

where JN = Job Number

RN = Run Number

MN = Man Number

ET = Estimated Running Time (Unit is 0.01 hrs.)

EMO = Estimated Mediary Output (Words)

EPO = Estimated Peripheral Output (Records)

The JOB card must be the first card of any deck to be run through the System since it serves to notify the System that a new job is about to commence. The information contained in the variable field of the JOB card serves these purposes:

1. It causes the System to enter the Accounting Initiation routine.

2. The identifying information (job, run and man-numbers) is passed along from phase to phase to identify debugging and other peripheral output.

3. The estimated running time may be used by an Interval trap routine or the operator to determine whether the object program is using too much time in the execution phase.

4. The estimates of mediary and peripheral output will enable the System to discover and forestall such errors as excessive debugging output, etc.

Each installation may, with a minimum amount of modification to the Monitor, specify its own parameters for the JOB card.

B. The LOAD control card.

The LOAD card follows the JOB card in the input deck and indicates that a squoze deck (with or without a modification package) is to follow. It furnishes all the required information as to which parts of the System will be required, when they will be required and how they are to operate in conjunction with the associated squoze deck.

LOAD A, B, C, D, E, F, G, H, J, K, L, M, N, P, Q

where A = + for input with only commentary text

A = - for input with only non-commentary text

A = B for input with both squoze texts

B = R for row binary input

B = C for column binary input

C = GO to execute if no definite errors

C = GOIF to execute if no errors

C = GOGOGO to execute in any event short of catastrophe

C = NOGO to suppress execution

D = Phase number for execution

E = SQZ to punch new squoze deck

E = NOSQZ to suppress punching squoze deck

F = + to output only commentary text

F = - to output only non-commentary text

F = B to output both texts

G = ABS to punch absolute deck

G = NOABS to suppress punching absolute deck

H = R for row binary output

H = C for column binary output

J = LIST to obtain listing

J = NOLIST to suppress listing

K= DICT to output dictionary output

K= NODICT to suppress dictionary output

L= DEBUG to execute with debugging

L= NOBUG to execute without debugging.

M= SS to use System symbols

M= NS to suppress System symbols

N= IN to execute with INTRAN

N= NOIN to execute without INTRAN

P= OUT to execute with OUTRAN

P= NOOUT to execute without OUTRAN

Q= NOMAC to execute without transmission macros

Q= TRMAC to execute with transmission macros

The normal case is:

LOAD B, C, GO, 2, NOSQZ, B, NOABS, C, NOLIST, DICT, DEBUG, SS, NOIN, NOOUT, NOMAC

C. The SCAT control card.

SCAT C,D,E,F,G,H,J,K,L,M,N,P,Q

(The parameters C through Q are as described in the writeup of the LOAD control card.)

The normal case is:

SCAT NOGO, 2, SQZ, B, NOABS, C, LIST, DICT, DEBUG,SS, NOIN, NOOUT, NOMAC

The SCAT card informs the System that a symbolic deck to be compiled follows immediately, and causes the presetting necessary to fulfilling the demands of the various parameters.

Single Text.

The options for single-text-output on both the SCAT card the LOAD card, and the options for single-text input on the LOAD card, may be exercised. The admissible parameter forms are as follows:

B signifies that both texts are present or required.
+ signifies that text with commentary only is present or required.
- signifies that text without commentary only is present or required.

No difficulty should be experienced with single-text output. Use of this option will result in a SQZ deck, numbered consecutively throughout, with only one text section and with a Preface card bearing the correct text word counts.

Single-text input may consist of either an originally double-text deck, with one text section manually removed; or an integral single-text deck, produced by a previous single-text output run. In the former case, the Preface card, which must not be changed, denotes a double-text SQZ deck the input parameter on the LOAD card must correspond to the physical state of the deck; in the latter case, the Preface card describes the deck completely and the input parameter on the LOAD card is ignored.

The following points should be noted:

1. Single-text output may be obtained from double-text input and vice versa. If, however, single-text input uses text without commentary then the two text sections of double-text output will, in the absence of modifications, be identical.

2. The normal case for both input and output is double text. Thus, though an integral single-text deck will load without difficulty using the control card LOAD NOGO, SQZ, LIST, a double-text output will be produced.

3. The appropriate text section of a single-text output deck, of either kind, will be punched by the M7 section of M & L. This

If the programmer wishes the data to be converted and edited, the parameters of the DATA card cause appropriate initialization to be made prior to entry into the Input Editor (Section 07.01) for conversion and transcription. If the programmer requires only that the data be transcribed on the specified tape unit, the Input Editor is not required. BCD records are written as 12 word logical records and binary records are written as 24 word logical records. A subsequent DATA control record causes and End of Logical Group flag to be written, and any other control record causes and End of Logical File flag to be written on SYSMOT or an End of Logical Tape flag to be written on SYSXRN. In the latter case, SYSXRN will be re-wound and disassigned.

The format of the DATA control card is:

DATA A,B,C

where

A is EDIT if the Input Editor is to be used for conversion and transcription.

A is NOEDIT if simple transcription is required.

B is GO to continue processing if bad data is encountered by the Input Editor.

B is GOIF to discard the job if bad data is encountered.

C is SYSMOT if data is to be placed on the System Mediary Output Tape.

C is SYSXRN if data is to be placed on reserved tape "N" on channel "X".

The normal case is:

DATA EDIT, GOIF, SYSMOT

02.04 INPUT DECK MAKE-UP

Owing to the ordered sequence in which the various components of the System perform their functions on a given job, there exist strict rules for the make-up of the input deck. The first of these rules is that the JOB card must be the first card in a deck. The JOB card may be followed by an IDENT card, and/or ASSIGN cards. The next card must be either a LOAD, SCAT, or WST control card, depending upon the type of operation to be executed. The composition of the remainder of the deck for each of these cases is described below:

A. LOAD control card
Blank card if desired
Squoze deck (no blank card at end of squoze deck)
When desired, a MOD package preceeded by a MOD card and ended by an ENDMOD is placed before the blank imbedded near the front of the squoze deck.
Data package(s) (optional, see below)
See 99.04.01 for squoze deck make-up.

B. SCAT control card
Blank card if desired
Symbolic deck (terminated by an END card)
Blank card required
Data package(s)(optional if 'GO' requested, see below)

Data packages may be for Input Editor translation (EDIT) or for mere transcription (NOEDIT). Their make-ups are:

A. DATA EDIT,...control card
Blank card if desired
Data cards
ENDATA control card
Blank card

B. DATA NOEDIT,...control card
Data cards

Multiple data packages may be used at the positions of the input decks as noted above, and in such a set, some can be EDIT packages while others are NOEDIT packages.

Note: The above are all of the allowable sequences of cards in an input deck for one job. Successive jobs are merely repetitions of these sequences.
See 99.05.01 for a more detailed explanation of input deck make-up.

To omit the blank card following the END card of a symbolic deck is catastrophic; blanks following System Control cards are now optional with the exception of ENDATA and WST.

03.01 SYSTEM SYMBOLS

System Symbols are used by the programmer for communication with the Monitor. Each symbol consists of six characters, the first three of which are SYS. The latter three characters mnemonically define specific locations to which the programmer may wish to make reference. These locations are of four types:

1. Control words for input-output units.
2. System transfer points (see Section 03.03).
3. System Data Cells for storage of key information (see Section 03.03).
4. Entries to System routines available for programmer use (see Section 03.04).

The first three types appear within the Communication Region of the Monitor; System Symbols of the fourth type refer to locations in the Execution Coordination routines, Buffering routines and Transmission Macros.

A System Symbol appearing in a programmer's code will be left by the Compiler as an undefined symbol since the locations assigned to them are variable. Modify and Load assigns absolute values to these symbols at load time with the aid of the System Symbol Table provided in the Monitor. Programs will normally have at least one of these undefined symbols since all codes must be terminated by a TSX to SYSTEM or SYSERR.

Entries in the System Symbol Table contain in bits 0-17 the base 50 representation of the last three characters of the symbol. The current absolute value assigned this symbol appears in the address part of the entry. The System

Symbol "SYSERR" (the entry to the System error subroutine) would appear as:

Bits 0-17: ERR in Base 50 (E*2500 + R*50 + R)

Bits 21-35: Absolute address of SYSERR.

Appendix A contains a complete list and definition of all System Symbols.

03.02 LOWER STORAGE TRAP CELLS

Decimal cells 0-31 of core memory are referred to as the Lower Storage Trap Cells. The absolute locations of these cells are dictated by hardware considerations and are not subject to reassembly. Entry is caused by:

1. The occurrence of a Trap in the Standard Trapping mode.
2. Floating Point Trapping mode which traps on floating point overflow or underflow.
3. Data Synchronizer Trapping mode which may trap at the occurrence of a channel signal:
   a) A redundancy tape check.
   b) An end of file.
   c) An IOCT, IORT, or IOST for which no load channel instruction is waiting in the main program upon completion of the command.
4. The interval trap and real time package.
5. A STR instruction.

When trapping occurs, control is transferred to the appropriate System or programmer subroutine via the trap cells and the Communication Region transfer points. Entry to a programmer's special routines through these transfer points is discussed in Section 03.03.

The Lower Storage Trap cells are assembled in the following fashion:

ORG	0		
PZE		Cell 0:	Trap Data Storage
TTR	TRP	Cell 1:	Trapping Mode Entry
TTR	SYSSTR	Cell 2:	STR Entry
PZE	**	Cell 3:	Interrupt Address Storage
TTR	SYSXTR	Cell 4:	Interrupt Entry
	****	Cell 5:	Not used
	****	Cell 6:	Not used
	****	Cell 7:	Not used
TTR	SYSPIL	Cell 8:	Floating Point Trap Entry
	****	Cell 9:	Not used
	****	Cell 10:	DSC Trap Data Storage (Channel A)
TTR	ATRP	Cell 11:	DSC Trap Entry (Channel A)
	****	Cell 12:	DSC Trap Data Storage (Channel B)
TTR	BTRP	Cell 13:	DSC Trap Entry (Channel B)
	****	Cell 14:	DSC Trap Data Storage (Channel C)
TTR	CTRP	Cell 15:	DSC Trap Entry (Channel C)
	****	Cell 16:	DSC Trap Data Storage (Channel D)

TTR	DTRP	Cell 17:	DSC Trap Entry (Channel D)
	****	Cell 18:	DSC Trap Data Storage (Channel E)
TTR	ETRP	Cell 19:	DSC Trap Entry (Channel E)
	****	Cell 20:	DSC Trap Data Storage (Channel F)
TTR	FTRP	Cell 21:	DSC Trap Entry (Channel F)

(Remaining cells through cell 31 are not used.)

03.03 COMMUNICATION REGION TRANSFER POINTS AND ASSOCIATED STANDARD ROUTINES

Transfer points are designed to allow the programmer three options in dealing with errors discovered in the object program. Each transfer point consists of two instructions within the Communication Region, the first of which is labeled with a System Symbol to which the programmer may refer. A transfer point is of the form

```
SYSXYZ   TXH   **,,**
         TXL   XYZ,,0
```

where SYSXYZ is the reference System Symbol

XYZ is the System routine normally entered.

The option desired by the programmer is exercised through modification of the instruction at SYSXYZ. Under no circumstances may the instruction at SYSXYZ+1 be altered or replaced by the object program.

The options afforded the programmer are:

1. Execute the standard routine provided by the System with the return determined by the System routine entered.

   Method

   If the first instruction is not preset by the programmer, the TXH at SYSXYZ will fail. Entry will be made to the specified System routine via the TXL at SYSXYZ + 1.

2. Execute the standard routine provided by the System and return to the address specified by the programmer.

   Method

   The decrement of SYSXYZ is preset by the object program with the special

return. After the standard System routine has been entered and executed, control is transferred to the address specified in the decrement.

3. Execute a special routine provided in the object program.

Method

The instruction at SYSXYZ is replaced by a transfer to the object program's special routine.

4. Execute a special programmer routine followed by the system routine.

Method

The instructor at SYSXYZ is replaced by a transfer to the object program's special routine. This routine then transfers to SYSXYZ+1 after performing its own functions.

5. Execute a special programmer routine followed by the system routine but making a return specified by the programmer.

Method

The instruction at SYSXYZ is replaced by an instruction such as TXL OWN 1,, OWN 2. Transfer to SYSXYZ causes entry to OWN 1. After OWN 1 is completed, it transfers to SYSXYZ+1 to carry out the system routine. The system routine then makes the special return to OWN 2.

There are nine transfer points and associated standard routines which the programmer may wish to use:

1. SYSERR--The standard Unexpected Error routine is entered by a TSX SYSERR, 4 and provides a message on SYSDOT saying'THIS JOB HAS CAUSED A RETURN TO SYSERR FROM XXXX." If no special return is given in the decrement of SYSERR, the system will load SNAP into core and give a console scoop before returning to SYSTEM. The console scoop is suppressed for special returns to avoid the possibility of SNAP covering the object code.
   The calling sequence to SYSERR is
   TSX SYSERR, 4
   No return normally

2. SYSBAD
   TSX SYSBAD, 4
   X L(U), T, Y

   X= PZE for read
   =MZE for write

   L(U)=location of tape unit address

T=0 for BCD mose
=7 for binary mode

Y=location of the beginning of the IO table

The bad spot routine will reposition the tape and read or write the records indicated in the object program's IO list. If the routine is not able to accomplish this without encountering a condition listed below, the tape will be positioned as found and the program transfers to SYSTRC. That routine prints the number of the unit which cannot be used successfully and transfers back to the monitor to process the next job. The conditions which may cause this are:

1. record or end-of-file cannot be written, 5 attempts have been to do so. Blank tape has been written prior to each attempt.

2. records in the IO list cannot be read successfully. The tape was correctly repositioned, but 10 attempts to re-read have resulted in redundancy.

3. in repositioning the tape from an ambiguous command list (see below), the routine is not able to read the first 5 words without redundancy. 15 attempts have been made.

4. beginning-of-tape has been encountered while the routine is trying an extra backspace and comparison of the 5 words (see below), in order to find the correct position for re-reading.

5. the tape cannot be correctly repositioned although more than 25 extra records have been backspaced and the first 5 words compared.

The IO list is specifically restricted in that it may not conceal more than 25 records with count type commands.

In writing a tape, the command list is unequivical about the number of records which must be backspaced before another attempt can be made to write. Read commands, unless they are exclusively IORP or IORT, force the bad spot routine to do a special search.

It backspaces the minimum number of records, reads the first 5 words referred to in the command list and compares them to the first 5 words which must have been read into memory. A match, or a match except for 1 bit only, is recognized as a correct position for an attempt at re-reading. Therefore, an IORP or IOSP at the beginning of the list must specify a word count which is less than or equal to a record for the first 5 words referred to.

The routine will not make the 5 word comparison unless the tape can be read at least that far without redundancy, and it makes 15 attempts to do this.

In the comparison shows that the tape is not yet in position, an extra backspace is issued and the comparison made again. A beginning-of-tape encountered at this stage will cause the routine to abandon the job after spacing the tape forward in order to leave it as found.

3. SYSTRC--The standard Tape Redundancy Check routine is entered by a TSX SYSTRC, 4 with the symbolic name of the tape unit in index register 2. An indication of the error is made in the problem status indicators and the routine exits to SYSERR.

4. SYSIOC--The standard Input Output Check routine is entered by a TSX SYSIOC, 4. An indication of the error is made in the problem status indicators and the routine exits to SYSERR.

5. SYSTDC--The Standard Divide Check routine is entered by a TSX SYSTDC, 4. Normally this routine goes to SYSERR after leaving an indication of the error.

6. SYSTUF--The Standard Floating Point Underflow routine is entered when a floating point underflow is trapped. The offended register is set to zero and an error indication made. Return is normally to the object program at the instruction following the one causing underflow.

7. SYSTOF--The standard Floating Point Overflow routine is entered when a floating point overflow is trapped. Normally this routine exits to SYSERR after leaving an indication of the error.

8. SYSTRP--The standard System Trap routine is entered when a transfer is trapped after the object program has executed ETM. The system routine calculates the proper return even if the programmer is using his own trap routines. If a transfer to the programmers trap routine is in SYSTRP, this routine will be entered with the trapping mode off after calculating the proper return and placing it as a TTR instruction with tag and indirect bits if any in SYSTRX. The programmer may, if he wishes, go to the system trap routine after doing his own work by transferring to SYSTRP 1. This will cause each transfer instruction and its location to appear on the debugging out-put tape SYSDOT as part of the output for that job. The trapping mode is then entered and return is made either to the calculated return or to the return specified in the decrement of SYSTRP. If SYSTRP is not changed by the programmer, each transfer trap will create the SYSDOT information and the calculated return will be made.

Modification of the system trap routine.

a. The proper return address is computed whether a programmer or

system trap routine is used and is made available in SYSTRX.

b. The programmer may now enter his own trap routine by
   1. Changing the instruction of SYSTRP to

      TXL A,,B

      where A is the location of the programmer trap routine
            B is a return address (see below)

   2. Include the system trap routine as part of his routine by

      TRA SYSTRP+1

   3. Return from this system trap routine will be made to the location B specified in the decrement of SYSTRP.

9. SYSSTR--A store location and trap instruction normally causes entry to the unexpected Error Routine SYSERR.

03.04 EXECUTION COORDINATION UTILITY ROUTINES

Numerous System routines are available to the programmer at execution time for the purposes of performing standard operations and conserving space and effort on the programmer's part. The Buffering Routines, Transmission Macros, Debugging Macros, INTRAN and OUTRAN Macros and System error routines are discussed in this manual and other documents. There are two additional routines which the programmer may wish to use. These are SYSCAP, the Comment Attached Printer Routine, and SYSMTL, the Mediary Tape Loader.

SYSCAP may be used to print up to 12 words of information in a single line on the attached printer. The routine should be used only for those messages to the operator which are vital to the operation of a job. SYSCAP is entered through the calling sequence...

```
TSX     SYSCAP, 4
 X      L,,N
return
```

where

X is PZE to force a skip to channel 2 of the carriage control tape

X is MZE to suppress skipping

L is the location of the first BCI word to be printed

N is the number of BCI words to be printed

The N BCI words are converted to line image, the computer is delayed until the printer channel is free, and the comment is transmitted to the printer.

SYSMTL may be used by the programmer to load code following a TCD card into core storage from SYSMIT, the mediary input tape.

The calling sequence is -

```
TSX     SYSMTL, 4
 X      L(U),,R
```

where

X is PZE to preset core to TSX SYSERR, 4

X is MZE to suppress presetting

L(U) is the location of the I/O unit control word, normally SYSMIT

R is the return address

R is 0 to use return on tape, i.e., the address specified on the END or TCD card

Upon return, the accumulator is positive if loading was terminated by a TCD card and negative if terminated by an END card. In either case, the address of the END or TCD card may be found in SYSTRA.

(Note: SYSMTL assumes that SYSMIT is in correct logical position. To alternate program files or load them in other than sequential order, the programmer must provide the correct logical position by use of buffering routines SYSBKS, SYSWTK, and/or SYSRTK.)

04.01 AVAILABILITY OF MACHINE COMPONENTS

During the Execution Phase the following machine components are available for use by the programmer:

A. Core storage above the System origin, which is the location of the first cell following those routines which must stay in core at all times and those special System routines requested by the programmer for use in the execution of his job. This location is found in the address field of SYSORG

1. If no System routines are requested SYSORG will be approximately 10,500 octal.
2. If in addition to the normal routines Debugging is requested, SYSORG will be approximately 12,600 octal.
3. If the transmission macros are requested SYSORG will be approximately 13,700 octal.
4. If INTRAN is requested SYSORG will be approximately 42,200 octal
5. If OUTRAN is requested SYSORG will be approximately 26,500 octal.

B. All drum units.

C. The on-line printer - only to give the operator special instructions via the System subroutine SYSCAP (Section 03.04)

D. All tapes except those designated as System tapes may be assigned as programmer reserved or utility tapes. The System tapes may be used in any of the following manners:

1. SYSMIT may be read through the buffering routines to:
   a. Bring in sections of code following a TCD card with the aid of the System mediary tape loader SYSMTL.

Bring in data converted by the System during phase 1 with the aid of the System buffering routines SYSRTK (the routine which reads a logical record) and SYSWTK (the routine which reads a single word).

2. SYSMOT may be written at execution time to transmit data for conversion to the System through use of the routines which write logical records, flags and information in buffered format (SYSNPT, SYSINF, SYSBLK and SYSWHT).

3. The System peripheral printer tape and peripheral punch tape, SYSPOT and SYSPPT, are normally reserved for use by the System, but some installations may choose to place these tapes at the disposal of the programmer at execution time.

During the Execution Phase the following machine components are not available to the programmer:

A. Core storage from decimal location 00000 through System origin.

B. The three System tapes except as noted under D in the preceding section.

C. Sense switches 1 through 6.

D. The M-Q entry keys.

04.02 TAPE STATUS CONTROL

The System maintains a dynamic record within the Communication Region of the status of all tapes in the 709 System. Each physical tape drive on the computer will always be in one of the following five states:

A. In use as a System tape, e.g., SYSTAP, SYSPOT, SYSMIT, etc.

B. In use as a programmer's reserved tape, e.g., SYSBR1

C. In use as a programmer's utility tape, e.g., SYSAU2.

D. Free and available for use.

E. Disconnected and unavailable, indicating that the specified tape drive is not presently attached to the computer.

Assignment of programmer reserved and utility tapes is normally accomplished by the operator or set-up man. For each symbolic tape which the programmer indicates that he expects to use, the operator inserts an ASSIGN card in the input deck associating a physical tape drive with the symbolic name. The information from the ASSIGN card is encoded and transcribed onto SYSMOT preceding the object code.

At execution time, the tape assignment information is analyzed and specified assignments are effected, prior to loading the object code. Good coding practice recommends that a check be made to insure that all necessary assignments have been made. This can be accomplished by the proper calling sequence to SYSTAS (see Section 04.04) or by testing the I/O Unit Control Word for non-zero status.

The tape assignment procedure is based on the following basic assumption:

1. The referencing of system, reserved and utility tapes is made symbolically through I/O control words. There is an I/O control word for each possible symbolic unit, i.e., one for each system tape, and eight for the reserved tapes and eight for the utility tapes possible on each channel present on a given machine. Reserved and utility control words normally are zero if the tape is not assigned, or contain the physical address of the unit in the address portion of the word if the tape is assigned. System tape control words contain in the decrement field the reference address, i.e., the physical unit which will be assigned when assignment of the symbolic unit is necessary. The address portion of this word is zero if the tape is unassigned, or contains the correct physical tape address if the tape is assigned. The mediary tapes, SYSMIT and SYSMOT are exceptions in that their decrement is used in the normal manner for buffered tapes, i.e., the decrement contains the location of the first buffer associated with the symbolic unit, or zero if no buffer is attached.

2. The tape status list is used for two purposes and is constructed in the following manner. There are eight entries (one for each possible physical unit) per 709 channel and these are arranged in descending order, so that the A1 entry is always the last in the table. The prefix reflects the status of each physical unit:

   PZE - Assigned as a systems tape
   PON - Assigned as a programmer reserved tape
   PTN - Assigned as a programmer utility tape
   PTH - Logically disconnected, i.e., unavailable for use - "OFF"
   MZE - Disassigned and available for use - "ON"

   The distinction between these last two states is important. A physical unit which is logically disconnected is one which is either in the hands of the CE's or by virtue of parameters used in assembling the monitor has been shown to be outside the range of the machine configuration. For example, if the parameters indicate that only 5 physical units are to be available

on channel B, units B6, B7 and B8 automatically will be logically disconnected. A unit which is unassigned is simply not in use. As a safeguard against operator error, it is necessary before assigning a logically disconnected state first to change its status to disassigned. The address and decrement fields are used as a reference to the I/O control words for programmer utility and reserved tapes.

3. The current tape assignment package differs from original specifications in its delegation of responsibility. Whereas the original intent was to have the monitor make tape assignments on the basis of its records of unit availability and communicate its choices to the operator, the current routines assume that it is operationally more desirable to have the operator communicate his desires to the monitor. It was felt that the "set-up man" could upon receiving a stack of jobs analyze the tape requirements of each, order the jobs and designate tape assignment for successive jobs. In such a way as to reduce between job tape--handling delay to a minimum. The operator then could use the set-up man's chart of tape usage as a guide to his action and use the printer messages as a check. Otherwise there can be no reasonable guarantee that the system's choice of utility tapes for one job will not conflict with the set-up man's arrangement of reserved tapes in the subsequent job. If prior to a cycle a set-up man is to specify optimum tape assignment he must have complete control over the tapes.

4. In a system of this sort, various interlocks are necessary to insure that necessary between phase and between job tape handling has been accomplished. For this reason, halts have been inserted as interlocks. These halts are among the most widely misunderstood portions of the tape assignment procedure. Their primary purpose is not to allow for changes, but to insure that the necessary changes have already been made.

5. All tapes are rewound prior to disassignment and all system tapes are rewound when assigned (except during recovery).

04.03 SYMBOLIC NAMES OF SYSTEM COMPONENTS

All input-output components within the Mock Donald system are referred to by six character symbols which specify the locations within the Communication Region of the input-output unit control word. The address part of the control word contains the BCD mode address of the unit, and the prefix reflects the status of the unit.

A. System Tapes

1. SYSTAP -- location of the System Tape, normally unit 1 on channel B.
2. SYSPIT -- location of the peripheral input tape, normally unit 2 on channel B.
3. SYSPOT -- location of the peripheral output tape, normally unit 1 on channel A.
4. SYSDOT -- location of the peripheral debugging output tape, normally unit 4 on channel A.
5. SYSPPT -- location of the peripheral punch tape, normally unit 2 on channel A.
6. SYSMIT -- location of the mediary input tape, normally unit 3 on channel A, depending upon the phase being executed.
7. SYSMOT -- location of the mediary output tape, normally unit 3 on channel A or unit 3 on channel B, depending upon the phase being executed.
8. SYSLBR -- location of the library tape. This is normally a part of the System tape.

B. On-line System card equipment

1. SYSCRD -- location of the on-line card reader, normally on channel A.

2. SYSPRT -- location of the on-line printer, normally on channel A.

3. SYSPCH -- location of the on-line punch, normally on channel A.

C. Utility and reserved tapes are referred to symbolically through cells in the communication region of the form SYSxUn or SYSxRn where

x is one of the characters A through F referring to a channel which is on the computer. This should be the programmer's recommendation for the physical channel assignment; however, the operator is in no way prohibited from assigning a different physical channel.
U specifies a utility tape (one which is scratch tape for this job.)

R specifies a reserved tape (one which was produced by a prior job or will be used by a subsequent job).

n is the programmer's symbolic tape number (1 through 8).

D. Examples of usage of symbolic tape names

1. Rewind reserved tape 2 on channel B using buffering rewind subroutine.

```
TSX     SYSRWD,4
PZE     SYSBR2
Return
```

where SYSRWD is the entry to the system rewind routine.

2. Read a block of information in buffered format from utility tape 3 on channel A.

```
TSX     SYSRTK,4
PZE     SYSAU3
Returns
```

where SYSRTK is the entry to the system routine to read a buffered record.

04.04 TAPE ASSIGNMENT ROUTINE - SYSTAS

Contained within Execution Coordination (and therefore available to programmers at execution time) is a skeletal tape assignment routine. The three principal uses of this routine are:

A. To insure that tape assignments intended to be accomplished by the operator with ASSIGN cards have actually been effected.

B. To perform reassignment when a program's reference to a symbolic tape unit, for one reason or another, involves multiple physical reels.

C. To reassign a particular physical tape unit when a program requires more symbolic tapes than there are physical tapes on the computer.

The calling sequence to SYSTAS is

```
TSX     SYSTAS, 4
(X)     L(U)
Error Return    (illegal (X) or L(U))
Normal Return
```

where

X = PON to assign (or check for assignment of) a reserved tape.

X = PTW to assign (or check for assignment of) a utility tape.

X = MZE to disassign a symbolic tape, and return the physical unit which was assigned to it to an available status.

Examples:

A.
```
TSX     SYSTAS, 4
PON     SYSBR1
TSX     SYSERR, 4       Error Return
```

will insure that a physical drive has been assigned as SYSBR1, if not assigned, the operator will be instructed to enter an assignment through the MQ keys.

B.
```
TSX     SYSTAS, 4
PTW     SYSAU1
TSX     SYSERR, 4       Error Return
```

will insure that SYSAU1 has a tape assigned it; if not, the operator may assign a physical tape to SYSAU1 through the MQ keys.

C.
```
TSX     SYSTAS, 4
MZE     SYSBR1
TSX     SYSERR, 4
TSX     SYSTAS, 4
PON     SYSBR1
TSX     SYSERR, 4
```

will disassign the physical tape associated with SYSBR1 and request that the operator reassign through the MQ keys.

(Note: The fact that tape assignment is accomplished through the use of SYSTAS only in those exceptional cases where ASSIGN cards are inadequate or misused suggests a concurrent use of SYSCAP to advise the operator regarding the nature of the operation.)

05.01 INTRODUCTION

The buffering routines are provided to the System and the programmer as a means of reading or writing mediary tapes. These routines do not read or write tapes directly, but instead make use of buffer units. The write routines place information and flag in buffer units to be written on tape at an appropriate time. The read routines take information from buffer units which have been filled appropriately. The actual reading and writing will be discussed in the section on dispatchers.

A. Buffer units normally consist of 256 words. The first word of a buffer unit is the buffer unit code word, and the remaining 255 words are available for information and flags. The system provides 2 buffers, but the programmer normally should provide additional core storage to the buffering routines for division into buffers.

The amount of core storage to be furnished is dependent upon the number of tapes to be read and/or written in buffered format and the number of buffer units to be attached to each such tape. (Section 05.02)

B. Physical tape records have a maximum length equal to System buffer size exclusive of the unit control word, i.e., 255 words. These records may consist of several logical records or parts of logical records.

C. Logical records may be of any size. They consist of information which the programmer plans to process in the future and

flags which identify and control the information.

D. Although all of the buffering routines described in Section 05.02 are available for use by the programmer, the direct use of only three (BUFAD, RETAK and NPUT) will accomplish all but the most complex operations.

E. Flags are written as DSU commands.

1. Block flags describe the location of blocks of information and the magnitude of the blocks.

IOCP Y,,N

where Y is the location in core of the block

N is the number of words in the block

The block flag is written on tape immediately preceding the information in the block.

2. Nominal origin flags may appear in write sequences but they are never written on tape.

IOCPN Z

where Z is the new nominal origin reference for the block flags in the following logical block.

This type of flag causes the buffering routines to process block flags in a non-standard fashion. The origins specified in the following block flags are used only to desig-

nate the initial locations for writing the data on tape. The flags themselves are altered prior to being written on tape. The first block flag origin is replaced by the address specified in the nominal origin flag. The origins of subsequent block flags are replaced by the address specified in the nominal origin flag plus the counts of the preceding block flags. Any non-block flag terminates this mode of processing. The information may later be read in if desired under control of the now revised block flags. The use of the nominal origin flag furnishes one means of writing from one area in core onto tape for later reading into another area of core, all under control of block flags.

3. End flags define logical control separations of information on tape.

a. Logical end of record flag

```
TCH  SYSLER  or
TCHN SYSLER
```

(The latter indicates the end of a record containing bad data. When reading such a record with SYSRTK, the error (bad data) return will result.

b. Logical end flags

```
TCH SYSPEF,,K
```

where K 0 for end of logical group
K 1 for end of logical file
K 2 for end of logical tape

c. Symbol flags may appear at the beginning of logical records for identification purposes.

```
IOSPN   Y,,N
```

where Y is the location of N synonymous symbols for this record. The symbol flag is written on tape immediately ahead of the N symbols.

d. Sequence flags

IOSP N,,O

N is the sequence number of this record.

F. Physical end-of-file marks are written by NPUT and WHYTE when end-of-logical file (TCH SYSPEF,,1) and end-of-logical tape (TCH SYSPEF,,2) flags are specified. When RETAK and WOTAK encounter one of these physical end conditions the tape is positioned to read the associated flag; any subsequent attempt to read this tape will force the tape to be repositioned at the same flag. It is not possible for RETAK or WOTAK to space over one of these flags. Programmers may not write these two flags on SYSMOT, but should write one of them on job-specific mediary tapes when all other output is completed. This will avoid reading over into non-buffered redundant records at a later time.

05.02 USE OF THE BUFFERING ROUTINES

A. Buffer Addition Routine - BUFAD

Purpose. -

To make a specified area of core storage available for use as input-output buffers.

Calling Sequence. -

```
TSX     SYSBFD,4
PZE     A,,N
error return
return
```

where SYSBFD is the entry address for BUFAD

A is the first cell of the block available

N is the number of cells in the block.

An error return will be made if the area specified is inadequate in size to provide at least one buffer of standard length (256 words).

B. Buffered Tape Write Subroutines - NPUT

Purpose. -

To output "n" logical flags and associated information.

Calling Sequence. -

```
TSX     SYSNPT,4
PZE     L(U),,N
.........
N flags
.........
return
```

where SYSNPT is the entry address for NPUT

L(U) is the location of the I/O unit control word for the tape specified

N is the number of flags which follow.

Example. -

```
        TSX     SYSNPT,4
1.      PZE     SYSBU 3,,9
2.      IOCPN   Z
3.      IOCP    A,,N1
4.      IOCP    B,,N2
5.      TCH     SYSLER
6.      IOCP    C,,N3
7.      TCH     SYSLER
8.      TCH     SYSPEF,,0
9.      TCH     SYSPEF,,1
10.     TCH     SYSPEF,,2
```

1. Nine flags follow which define information to be written on the tape assigned as utility unit 3 on channel B.

2. The nominal origin to appear on subsequent block flags in the block is to be relative to Z.

3., 4., 5.

Write a logical record in the following format:

```
IOCP    Z,,N1
```

N1 words from location A

```
IOCP    Z + N1,,N2
```

N2 words from location B

```
TCH     SYSLER
```

6. Write a logical record in the following format:

```
IOCP    C,,N3
```

N3 words from location C

```
TCH     SYSLER
```

7. Write a flag signaling the end of a logical group

```
TCH   SYSPEF,,0
```

8. Write a flag signaling the end of a logical file

```
TCH   SYSPEF,,1
```

9. Write a flag signaling the end of a logical tape

```
TCH   SYSPEF,,2
```

C. Output routine for a block flag - BLOCK

This routine and the two which follow it (INFO and WHYTE) are used to output flags and information one word at a time.

Purpose. -

To output a single flag and initialize the routines to output the information which will follow it on tape.

Calling Sequence

```
TSX     SYSBLK, 4
OP      L(U), TAG
```

where SYSBLK is the entry address for BLOCK

OP specifies the register in which information to be output subsequently may be obtained

L(U) is the location of the I/O unit control word for the specified tape

TAG is the index register available for use

The block flag to be output is in the logical accumulator on entry to BLOCK

D. Output Routine for a word of information - INFO

Purpose

To output a word of information as part of the BLOCK specified by the previous BLOCK flag.

Calling Sequence

```
TSX     SYSINF, 4
```

where SYSINF is the entry address for INFO

Usage

A word of information is in a register according to the OP specified in the previous BLOCK linkage (STO, SLW, STQ, or STI).

E. Output routine for a non-block flag - WHYTE

Purpose

To output a non-block flag

Calling Sequence

```
TSX     SYSWHT, 4
PZE     L(U)
```

where SYSWHT is the entry address for WHYTE

L(U) is the location of the I/O unit control word for the specified tape.

The non-block flag to be output is in the logical accumulator on entry to WHYTE.

Example of the use of BLOCK, INFO and WHYTE

The following sequence of code outputs:

1. A block flag specifying 500 words of information
2. 500 words of information a word at a time as it is computed
3. A logical end of record flag

```
        CAL     BLKFL       pick up block flag
        TSX     SYSBLK,4    go to output block flag and initial-
                            ize INFO
        STO     SYSAR2, 2   parameters specifying register,
                            output unit, and index register
LOOP    CLA     A+500, 2    pickup data word
        ADD     B+500, 2    form sum
        TSX     SYSINF, 4   go to output sum
        TIX     LOOP, 2, 1  test for all sums formed
        CAL     ENDFL       pickup end flag
        TSX     SYSWHT, 4   go to out put end flag
        PZE     SYSAR2      parameter specifying output unit
BLKFL   IOCP    A,,500      block flag
ENDFL   TCH     SYSLER      logical end of record flag
A       BSS     500         read-in storage
B       BSS     500         read-in storage
```

F. Logical Record Input Subroutine - RETAK

Purpose

This routine causes one logical record to be taken into working storage.

Calling Sequence

```
TSX          SYSRTK, 4
OP           L(U), , Y
error return (TCHN SYSLER - bad data flag -terminated log. record just read)
end of logical file return
end of logical group return
normal return
```

where SYSRTK is the entry address for RETAK

OP = PZE to place the record in storage starting at Y

OP = MZE to place the record in storage at Y + the nominal origin of the record (See Section 5.01)

L(U) is the location of the I/O unit control word for the specified tape.

Example of use of RETAK

Through use of the DATA control card, information may be placed on tape in buffered format. A DATA deck might be composed of:

```
DATA        NOEDIT,SYSMOT
    6 BCI  cards
DATA        NOEDIT,SYSMOT
    6 BCI  cards
DATA        NOEDIT,SYSMOT
    6 BCI  cards
(end of input deck)
```

This input deck will be placed on SYSMOT as 3 logical groups, each consisting of 6 logical records of 12 words, each, the third group being followed by a logical end of file.

The coding required to read this data at execution time could be:

```
P1      TSX       SYSRTK, 4          go to read logical record
P2      PZE       SYSMIT,,A          parameters specifying unit,
                                     storage mode and address
        TRA       ERROR              error return
        TRA       EOF                logical end of file return
        TRA       EOG                logical end of group return
        LXD       P2,2               step storage address for next
                                     logical record
        TXI       *+1,2,12
        SXD       P2,2
        TRA       P1                 go to read next logical record
EOF     all data cards read - exit
EOG     computation on data in group
        AXT       A,2                reset storage address for new
                                     logical group
        SXA       P2,2
        TRA       P1                 go to read next logical group
```

G. Word Input Routine - WOTAK

Purpose

To input a single flag or word of information

Use

WOTAK may be used instead of RETAK when the programmer wishes to input only parts of logical records or is interested in interpreting the flags associated with input data.

Calling Sequence

```
TSX     SYSWTK, 4
OP      L(U), TAG
PZE     A,,B
        information return
```

where SYSWTK is the entry address for WOTAK

OP is CLA, CAL, LDQ or LDI specifying the register into which the word of information is to be placed

L(U) is the location of the I/O unit control word for the specified tape.

TAG is the index register available for use by the buffering routines.

A is the return to be made when a block flag is encountered. A must either be the location of the PZE parameter (i.e.*) or the location of a routine which goes to the location of the PZE parameter as shown in the example below.

B is the return to be made upon encountering any flag other than a block flag.

The calling sequence specified by the programmer is temporarily replaced by the buffering routines while WOTAK is in operation. The three-word calling sequence is replaced by:

```
TXI     *+1, TAG, -1
TXL     SI, TAG, -Y-N
OP      T-Y, TAG
```

where SI is the address in WOTAK where restoration of the original calling sequence is accomplished.

T is the location of the buffer into which the information has been read from tape.

Y is the nominal origin of the first word in the block.

N is the number of words in the block.

OP is the instruction which moves a word of information from the buffer to the desired register.

TAG is the index register specified in the original calling sequence.

Example of the Use of WOTAK

In the example given for the use of RETAK, complete logical records were brought into working storage and the associated flags were of interest

only to the buffering routines. If only the first three words of each logical record were of interest to the programmer, the following sequence of code might be used to read the same tape:

```
Q1      AXT     A, 1            preset for group storage
        SXA     Q5, 1
Q2      AXT     4, 1            preset test for three words
Q3      TSX     SYSWTK, 4           TXI   *+1, 2, -1
        CAL     SYSMIT, 2           TXL   SI, 2, -Y-N
Q4      PZE     BF,,NBF             CAL   T-Y, 2
        TNX     Q3, 1, 1        go to step over unwanted words
Q5      SLW     **, 1           store information
        TRA     Q3              go to get next word
BF      LXA     Q5, 4           step storage for next logical record
        TXI     *+1, 4, 3
        SXA     Q5, 4
        TRA     Q4              go to read next logical record
NBF     CAS     LER             test for end of logical record
        TRA     *+Z             no
        TRA     Q2              yes - go to find next logical record
        CAS     LEG             no - test for logical end of group
        TRA     *+2             no
        TRA     Q7              yes - go to perform computation on
                                  group
        CAS     LEF             no - test for logical end of file
        TRA     *+2             no
        go to end of data action
        error - unexpected flag
Q7      perform computation on logical group
        TRA     Q1              read next logical group
LER     TCH     SYSLER          go to read next logical group
LEG     TCH     SYSPEF,,0
LEF     TCH     SYSPEF,,1
```

In this example, it is assumed that the information is on tape as shown. If there were sufficient data to necessitate the use of more than one buffer, ad-

ditional coding would be necessary to handle the possibility that the first part of a block of data might be in one buffer and the second part in another buffer. The programmer would then have to test the count appearing in the block flags.

H. Logical Backspace Routine - BKSPC

Purpose

To backspace a logical record.

Calling Sequence

```
TSX    SYSBKS, 4
 X     L(U)
error return
return
```

where SYSBKS is the entry address for BKSPC.

X is PZE for an input tape.

X is MZE for an output tape.

L(U) is the location of the I/O unit control word for the specified tape.

An error return is made when the routine is unable, for any reason, to perform the requested operation.

I. Tape Rewind Routine - REWIND

Purpose

To rewind a tape read or written in buffered format.

Calling Sequence

```
TSX     SYSRWD, 4
PZE     L(U)
return
```

where SYSRWD is the entry address for REWIND

L(U) is the location of the I/O unit control word for the specified tape.

This routine empties all buffers attached to an output tape and writes an end of file on the tape prior to rewinding, etc. Input tapes are simply re-wound. The buffers attached to the unit are released for attachment to other units.

J. Drain Routine - DRAIN

Purpose

To empty output buffers onto tape.

Calling Sequence

```
TSX         SYSDRA, 4
PZE         L(U)
ERROR       RETURN
NORMAL      RETURN
(ERROR RETURN IS NOT USED)
```

Where L(U) is the I/O unit control word for the specified output tape. This routine empties all buffers attached to the specified output tape and releases the buffers. The tape is otherwise unaffected.

05.03 BUFFERING DISPATCHER

Efficient use of the System buffering routines requires that input and output be accomplished on a preferential basis. A standard dispatching routine is included within the System and is available to the programmer. The facility is also provided for the programmer to substitute his own special dispatcher.

The System dispatcher is composed of three principal subroutines:

A. Initialization

Purpose

To provide the dispatcher with a list of priority preferences.

Calling Sequence

```
TSX  SYSDPI,4
PFX  L,,N
return
```

where PFX is PZE to add a dispatching list
PFX is MZE to delete a dispatching list
L is the location of a list of I/O unit control word addresses to be dispatched
N is the number of entries in the list

The format of the list is

```
L      PFX   L(U),,N1
          .  .  .  .

L+N-1  PFX   L(U),,NN
       PZE
```

where PFX is PZE for an input unit
PFX is MZE for an output unit
L(U) is the location of the I/O unit control word for the tape for which dispatching is to be performed
N is the number of buffers to keep ahead

A terminal word of zero is required.

B. Normal Dispatching

Purpose

To give control to the Dispatcher and allow it to perform its function on the provided list.

Calling Sequence

```
TSX  SYSDIS,4
Return
```

C. Dispatcher Suppression

Purpose

To prevent all subsequent dispatching on a particular I/O unit.

Calling Sequence

```
TSX  SYSDPS,4
PZE  L(U)
Return
```

Where L(U) is the location of the I/O unit control word for the tape for which dispatching is no longer required or wanted.

06.01 TRANSMISSION MACROS

The Transmission Macros are a set of calling sequences to an input-output dispatching program. While relieving the programmer of nearly all I/O timing considerations, these macros provide for efficient use of the parallel input-output facility of the 709.

Data is moved directly between working storage and the specified I/O unit using the DSC commands provided by the programmer. Buffers are not employed; therefore, no arbitrary limits are placed on the length of records being read or written. The programmer may interrogate the status of any transmission at any time, and he may assign priority to specific operations.

If an object program, through the use of a Transmission Macro, attempts to initiate transmission on a channel that is in operation, an entry is made for that macro in the "stack table" of the channel involved and control is returned to the object program. Subsequently, when any of the transmission macros gives control to the dispatching program, the status of each channel is checked and waiting operations (i.e., those previously entered into the stack table) are initiated.

Since the Transmission Macros are not designed to read or produce information in the MockDonald buffered format, it is not possible to operate

on the same tape with both the transmission macros and the MockDonald buffering routines.

In the following description of each of the transmission macros, each element in the designated variable field is expressible as a SCAT symbolic expression. Each refers to a core address except "T", which is the index register associated with the address "Y", and "N", which is a spacing count for the non-transmitting tape-moving macros. The location counter symbol (*) may not be used, and a blank or zero field may have special meaning.

Each of the macros except DISP specifies an input-output operation referring to a table located at the effective address Y - C(T), or indirectly thereto if the macro is indirect. The first word of an I/O table specifies the symbolic tape unit; in the case of a READ or WRITE macro, the remainder of the table consists of the DSC commands which are to control the actual transmission. The I/O table is required by the dispatching program throughout the execution of the transmission and must not be altered until the operation is complete.

The following example is offered to illustrate the format of an I/O table:

```
X       PZE     SYSAR1
        IOCP    A,,3
        IOST    B,,1000
```

This table, when referred to by a READ macro, specifies input from the tape assigned as Reserved Tape #1 on Channel A in the binary mode (MZE in the prefix of X indicates BCD mode).

The first three words of a record will be transmitted to locations A through A + 2, and the remainder of the record (up to a maximum of 1,000 words) will be transmitted to location B and cells following.

Each of the transmission macros except DISP has an ERROR return as one of its variable field parameters. With the exception of the IN and OUT macros, the dispatching program will transfer control to the location specified by the ERROR parameter if either of two conditions occurs:

1. The stack table for the channel involved is full, or
2. The tape specified in the first word of the I/O table has not been assigned.

Upon return to ERROR, the sign of the MQ register will be plus in the first instance and minus in the second. If ERROR is zero or blank, the return will be to SYSERR with index register 4 containing the complement of the macro location.

A. READ Macro

READ Y,T, ERROR

The execution of this macro enters a Read operation in the

stack table of the specified channel. When the dispatching program executes the READ operation, the I/O table at location Y - C(T) is used to specify the unit and control the reading. The status of a READ operation is tested by an IN macro referring to the same I/O table.

B. STEPR Macro

STEPR Y,T,N, ERROR

This macro enters a STEP RECORDS operation in the proper stack table. The I/O table at Y - C(T) need consist only of the symbolic tape address. That tape is spaced forward N records or until End-of-File or End-of-Tape is encountered. Such an end indication is not counted as a record. The status of a STEPR operation is tested by an IN macro.

C. STEPF Macro

STEPF Y,T,N, ERROR

This macro enters a STEP FILES operation in the stack table which causes the specified symbolic tape to be spaced forward N files.

The status of a STEPF operation is tested by an IN macro.

D. WRITE Macro

WRITE Y,T, ERROR

This macro enters a WRITE operation in the proper stack table, exactly as described for READ. The status of a WRITE operation is tested by an OUT macro.

E. WEOF Macro

WEOF Y,T, ERROR

This macro enters a Write-End-of-File operation in the stack table of the dispatching program, with the one-word I/O table at location Y = C(T) symbolically specifying the tape on which an end-of-file mark is to be written. The WEOF operation status is checked by an OUT macro.

F. BACKR Macro

BACKR Y,T,N, ERROR

This macro causes the tape specified in the I/O table to be backspaced N records, or until Beginning-of-Tape is encountered. This operation, because of its inherent inefficiency, should be used

sparingly, if at all. Its status is checked by an OUT macro.

G. BACKF Macro

BACKF Y,T,N, ERROR

This macro enters in the stack table a Backspace Files operation which, when executed, backspaces the specified tape N programmed files. (The tape is backspaced N+1 files. If load point is reached in less than N+1 backspaces, an error is noted for subsequent reporting on an OUT macro. If loadpoint is reached on the N+1st backspace, the operation is complete. If not at loadpoint, the tape is moved forward over the end of file mark.) If a "programmed file" is defined to start at load point or the gap following an end of file mark and to end with an end of file mark, a tape is always positioned in a programmed file. Then a BACKF N will move the tape back to the beginning of the Nth programmed file preceeding the one in which the tape is currently positioned.

The status of a BACKF operation is tested by an OUT macro.

H. BACKT Macro

BACKT Y,T, ERROR

This macro causes the logical tape specified symbolically at location Y - C(T) to be rewound.

The status of the operation is tested by an OUT macro.

I. IN Macro

IN Y, T, NI, ERROR, EOF

The execution of this macro interrogates the status of the earliest READ, STEPR or STEPF operation relating to the I/O table at location Y - C(T). If the operation has been successfully completed, a return is made to the next program step; the accumulator contains the DSC registers for the channel at the completion of the operation in the case of a READ.

If the operation has not been completed, a "Not In" return is made to location NI unless NI is zero or blank, in which case the dispatcher retains control until the operation is complete. On the "Not In" return, the accumulator is set to indicate the state of the transmission:

1. A READ Operation
   a. If the operation has not yet been started, the accumulator is set to zero.
   b. If the operation is in progress, the accumulator contains the status of the DSC registers involved in the transmission.
2. A STEPR or STEPF operation
   a. If operation has not started, the decrement of the accumulator is set to zero.
   b. If operation is in progress, the decrement of the accumulator contains the number of files or records already stepped, and the address contains the number remaining to be stepped.

If the operation was a READ or STEPR and was successful but terminated on an End-of-File, a return is made to location EOF with the accumulator containing the DSC registers or stepping counts as they were when the End-of-File was encountered. In the case of these two macros, if EOF is zero or blank, return is made to ERROR. If the operation was a STEPF and the specified number of files were successfully spaced over, the return is to EOF or, if EOF is zero or blank, to the next program step.

If the operation produced an error, a return is made to location ERROR with the accumulator containing the DSC registers or stepping counts as they were at completion of transmission and with the MQ register containing error bits in the following pattern:

Bit	Nature of Error
S	No READ, STEPR or STEPF macro has been given for the specified I/O table.
1	Operation completed but unsuccessful.

Bit	Nature of Error
2	Not used.
3	End-of-File was encountered during operation

The IN macro removes its corresponding entry from the stack table on all returns except to NI. The object program must execute IN macros in such a way that each READ, STEPR or STEPF entry in the stack tables is removed in order to preclude saturation of the stack tables and resultant "Too Full" error returns when subsequent macros are executed.

J. OUT Macro

OUT Y,T,NO, ERROR

The execution of the OUT macro interrogates the status of the earliest WRITE, WEOF, BACKR, BACKF or BACKT operation relating to the I/O table at the specified location and, on all except the NO return, removes the corresponding entry from the stack tables. The program must execute OUT macros in such a way as to insure that each entry will be removed from the stack tables.

If the operation has been successfully completed, a return is made to the next program step with the accumulator containing the DSC registers or backspacing counts as they were at the end of the transmission.

If the operation has not been completed, a "Not Out" return is made to location NO with the accumulator indicating the state of transmission in the same format as that described for the IN macro. If NO is zero or blank, the dispatching program retains control and delays until the operation is complete.

If the operation was completed but produced an error, the return is to ERROR with the MQ containing the appropriate error bits as follows:

Bit	Nature of Error
S	No WRITE, WEOF, BACKR, BACKF or BACKT macro has been executed relating to the specified I/O table.
2	A BACK operation attempted to backspace beyond the beginning of tape. (The decrement of the accumulator contains the number of records or files actually spaced.)
4	End-of-Tape encountered by WRITE or WEOF.

K. RUSH Macro

```
RUSH    Y,T ERROR
```

The execution of this macro causes a channel priority to be given to the unit specified in the contents of location Y - C(T), i.e., all previous operations which relate to this unit are advanced in the stack table of the designated channel so that these operations will be executed as soon as possible. RUSH does not interrupt a transmission already in progress, nor does it apply to operations on that channel which are initiated by macros following the RUSH macro. If more than one RUSH macro is executed, the order of priority is the order of execution of the RUSH macros.

L. DISP Macro

DISP

The execution of this macro causes the dispatching program to update the operation of all channels. The updating function is automatically performed each time any of the transmission macros is executed, but it may be necessary for certain types of programs to execute DISP occasionally to avoid undue idle time on channels.

The installation of the Data Synchronizer Trap device on the computer will obviate any need for the DISP macro, since the DSC trap automatically transfers control to the System's dispatching routine.

06.02 EXPANSIONS OF TRANSMISSION MACROS

The various transmission macros are automatically expanded by the Compiler as follows:

A.	READ	Y,T,ERROR	CAL * TXL SYSTMA,,ERROR PZE Y,T Normal Return
B.	STEPR	Y,T,N, ERROR	CAL * TXL SYSTMA+1,,ERROR PZE Y,T,N Normal Return
C.	STEPF	Y,T,N, ERROR	CAL * TXL SYSTMA+2,,ERROR PZE Y,T,N Normal Return
D.	WRITE	Y,T,ERROR	CAL * TXL SYSTMA+3,,ERROR PZE Y,T Normal Return
E.	WEOF	Y,T, ERROR	CAL * TXL SYSTMA+4,,ERROR PZE Y,T Normal Return
F.	BACKR	Y,T,N, ERROR	CAL * TXL SYSTMA+5,,ERROR PZE Y,T,N Normal Return

G.	BACKF	Y,T,N,ERROR	CAL * TXL SYSTMA+6,,ERROR PZE Y,T,N Normal Return
H.	BACKT	Y,T,ERROR	CAL * TXL SYSTMA+7,,ERROR PZE Y,T Normal Return
I.	IN	Y,T,NI,ERROR,EOF	CAL * TXL SYSTMA+8,,ERROR PZE Y,T,NI PZE EOF Normal Return
J.	OUT	Y,T,NO,ERROR	CAL * TXL SYSTMA+9,,ERROR PZE Y,T, NO Normal Return
K.	RUSH	Y,T,ERROR	CAL * TXL SYSTMA+10,,ERROR PZE Y,T No rmal Return
L.	DISP		CAL * TXL SYSTMA+11,0 Normal Return

07.01 THE INPUT EDITOR

The Input Editor is available for the conversion of data during the Input Phase. The binary results of the conversion are written on a mediary output tape, in a form suitable for later reading by means of the buffered routines RETAK and WOTAK.

Cards containing data to be converted by the Input Editor will generally have a class code punched in column one. This code specifies the conversion program to be used. Class codes 0, 1, . . . , 9 are reserved for installation conversion routines; +0, A, B, . . . , I are available for use by the individual programmer.

The conversion corresponding to a programmer class is described by a FORMAT statement. This statement must be inserted at some point prior to the data to be converted.

A special programmer class, $, permits the use of data cards which cannot be punched with a class code in column one. The $ FORMAT statement, which describes the conversion required for these cards, must immediately precede them.

The Input Editor requires that INTRAN be on the system tape.

07.02 THE DATA PACKAGE

The data to be processed is arranged as follows:

1.	DATA	A, B, C	(MD control card)
2.	Blank card		(optional)
3.	Data to be converted, including control cards and data cards.		
4.	ENDATA		(MD control card)
5.	Blank card		(Required)

More than one data package may be processed; each package must be arranged as described above.

07.03 CONTROL CARDS

The DATA and ENDDATA cards are MD control cards: they must have the 7-8-9 combination punch in column one. The control cards discussed below are a part of the data; column one must be blank, or punched with a "$".

A. ENDRCD

This card will cause a logical end of record flag to be written on the output tape. An 11 punch in column one of an installation class data card will cause an end of record to be written following the data on that card. Each group of successive data cards must be terminated by an end of record. ENDRCD cards not immediately following data are ignored.

B. ENDGRP

This card will cause a logical end of group flag to be written. An ENDGRP card will always have the same effect; the programmer may have two or more in succession. A card with an "*" in column one, and otherwise blank, will also cause an end of group to be written.

C. ENDFILE

This card will cause a logical end of file tag to be written, if the output tape is not SYSMOT. If SYSMOT has been specified, the ENDFILE card is an illegal control card.

It should be noted that it is not possible to read past a logical end of file flag using the buffered input routines.

D. ENDTAPE

This card will cause a logical end of tape flag to be written with the same restrictions as ENDFILE.

E. NOMORG

During execution of his program, the programmer will ordinarily read his converted data with the standard input routine RETAK. In the calling sequence to RETAK, the starting location into which the data is to be read is specified. This starting location may, at the programmer's option, be incremented by the nominal origin associated with the data. The nominal origin may be specified in two ways: through the use of a NOMORG card, or on the data card itself.

The format of the NOMORG card is

NOMORG A, B

where A is the nominal origin in decimal, with optional leading sign

B is described below

1. NOMORG A

This card may appear at any point in the data package. Its effect is to set the nominal origin to 'A'. A succeeding ENDRCD or its equivalent returns the mode of operation to normal. A nominal origin specified in a field of a data card has precisely the same effect as 'NOMORG A', (see 07.07.03).

2. NOMORG A, RECORD

This card may appear only at the beginning of a record. Its effect is to set the nominal origin to 'A' at the beginning of that and

each following record. An ENDGRP or its equivalent returns the mode of operation to normal.

3. NOMORG A, GROUP

This card may appear only at the beginning of a group. Its effect is to set the nominal origin to 'A' at the beginning of that and each following group. An ENDFILE returns the mode of operation to normal.

The mode of operation is always either normal, or that prescribed by the latest NOMORG card or its equivalent. The normal mode of operation is

NOMORG 0

F. FORMAT

This card is used to describe the conversion of programmer class data cards. The format of the card is

FORMAT A, (FORMAT Statement)

where A is one of the programmer class codes. FORMAT cards must occur somewhere prior to the data cards whose format they describe. A programmer class may be redefined at any time. For a complete discussion of FORMAT statements, see Section 07.07.

G. ETC

This card is a continuation card for FORMAT statements which

cannot be contained on one card. As many ETC cards as necessary may be used. The format of the card is

ETC (Continuation of FORMAT statement)

07.04 THE $ CLASS

The $ class is a special programmer class which permits the use of data cards which do not have a class code punched in column one. The "$ mode" is entered upon encountering a FORMAT card with a '$' punched in column one. This "$ FORMAT card" must immediately precede the data to be converted. All subsequent control cards must also have a '$' punched in column one. The only restriction on the data cards is that they must not have a '$' in column one.

Data may be punched in column one; therefore the format specification must begin with column one, rather than with column two, the first data column of ordinary data cards.

A special control card, $ STOP, is required to release the Input Editor from the $ mode.

07.05 ERROR ANALYSIS

The Input Editor performs an extensive error analysis on both data and control cards. Upon encountering an error, an indicative comment is printed on the System output unit. When appropriate, the card number, card column, and card image are printed.

A. Type one errors

Type one errors are those which would lead to loss of control by the Input Editor. The job is deleted immediately upon encountering a type one error, and control is returned to the system.

B. Type two errors

Type two errors are those which would definitely prevent successful execution of the program. In order to locate as many errors as possible, processing of data continues. However, the job is deleted.

C. Type three errors

Type three errors are those which might prevent successful execution of the program. If 'GO' was punched in the variable field of the DATA card, the job is not deleted.

In general, a type three error involves bad data, such as illegal characters, loss of significant bits, or floating point spill. . A bad data flag is written at the end of the logical record in which the bad data occurred. When this flag is encountered by RETAK, the error return

is made. Bad data is replaced by zero, except in the case of a floating point overflow, when it is replaced by $\pm 377777777777_8$.

07.06 ERRORS

Error messages are enclosed in quotes.

A. Type one errors.

1. "Illegal or blank control card."
2. "Binary card not a control card (7-8-9)."
3. "Binary record on tape not column binary in origin."
4. "More than ten persistent redundancy checks on input tape."
5. "Attempt made to process past end of record." This error can occur only if an installation class conversion routine is erroneously programmed.
6. "Physical end of file encountered on input unit." This is a type three error if the end of file is encountered after the ENDDATA card is read.
7. "IREDUN return from INTRAN." This error can occur only if an installation class conversion routine is erroneously programmed.

B. Type two errors.

1. Certain errors encountered in scanning a FORMAT statement.
   a. "Parenthesis trouble, first noticed here."
   b. "The preceding specification is incorrect."
   c. "The specified Hollerith field was not complete on this card."

d. "Illegal character."

e. "A non-zero number must precede this character."

f. "Specification not properly terminated."

g. "Too many columns specified for tape (or card) record."

h. "Character is illegal in this context."

i. "A number must precede this character."

j. "The conversion routines compiled for this job exceed the capacity of storage."

k. "Illegal FORMAT class specified."

2. Any spill produced in converting the field of a NOMORG card, or a nominal origin field in a data card.

3. An illegal character in the nominal origin field of a data card.

4. "Improper specification in a NOMORG."

5. "Illegal class code."

6. "Legal but undefined class code."

7. "ETC not preceded by FORMAT."

8. "Persistent redundancy check on input tape."

C. Type three errors.

1. Spills produced in the process of data conversion.

a. "Numeric part of OCT integer exceeds 2**36."

b. "Absolute value of DEC integer exceeds 2**35."

c. "String of numeric digits exceeds 2**35."

d. "E or B field exceeds 4 characters."

e. "Indicated binary pt. causes loss of left bits."

f. "Binary subfield extends into more than 1 word." This error cannot occur in a conversion defined by a FORMAT statement.

g. "IBCC or IBCW with zero field width." This error cannot occur in a conversion defined by a FORMAT statement.

h. "Floating pt. underflow occurred in conversion."

i. "Floating pt. overflow occurred in conversion."

2. "Character X is illegal in data field."

3. "Field width exceeds 31 columns. Compilation of FORMAT statement will continue." The field is taken to consist of the rightmost 31 columns.

4. "NOMORG RCD/GRP not at beginning of rcd/grp." The required end of record or end of group is written, and the NOMORG card is then processed.

5. "ENDGRP or ENDDATA not preceded by ENDRCD." The required end of record is written.

6. "$STOP or FORMAT not preceded by ENDRCD." The required end of record is written.

7. "STOP card in improper sequence." The card is ignored.

8. "Control card (7-8-9) not an ENDDATA card." The card is treated as an ENDDATA card.

07.07 THE FORMAT STATEMENT

The FORMAT statement describes the arrangement of data in a card, and prescribes the types of conversion to be performed between this data and its binary equivalents. The FORMAT statement is examined by the Input Editor, and a conversion routine is generated, to be executed when data referring to the FORMAT statement is encountered.

The FORMAT statement describes a card by indicating, for each field in the card, starting from the leftmost column available for data:

1. The type of field.
2. The width 'w' of the field, in columns.
3. Other modifiers, as necessitated by the field type.

A. Basic Field Specifications

1. Iw

   The field is assumed to contain a (signed) decimal integer, which is converted to binary.

2. Ow

   The field is assumed to contain a (signed) octal integer, which is converted to binary.

3. Ew.d, Fw.d

   The field is assumed to contain a (signed) floating point decimal

number, 'd' must be specified. Unless a decimal point occurs in the data, the decimal point is assumed to be 'd' places to the left of the rightmost digit of the principal part. If an exponent part does not occur, it is assumed to be zero. The exponent part must begin with an "E' or a sign, or both. The number is converted to floating point binary.

4. Ew.dBb, Fw. dBb

The field is assumed to contain a (signed) floating point decimal number. 'd' must be specified. Unless a decimal point occurs in the data, the decimal point is assumed to be 'd' places to the left of the rightmost digit of the principal part. If an exponent part does not occur, it is assumed to be zero. The exponent part must begin with an 'E' or a sign, or both. The number is converted to fixed point binary. 'b' must be specified. Unless a B part occurs in the data, the binary number is assumed to have its binary point 'b' places to the right of the leftmost bit.

5. Aw

The field is assumed to contain Hollerith-punched data. The data is converted to BCD, and left adjusted, with zeros inserted in unused character positions.

6. wH

The 'w' columns following the 'H' are skipped over in the scan

of the FORMAT statement. Data is converted as though the specification read 'Aw'. Note that the field of a 'wH' must be complete on one FORMAT or ETC card.

7. Nw

The field is assumed to contain a (signed) decimal integer. The integer is converted to binary, and becomes the nominal origin for the following data. A blank field is ignored.

8. NwO

The field is assumed to contain a (signed) octal integer. The integer is converted to binary, and becomes the nominal origin for the following data. A blank field is ignored.

9. wX

The field is ignored.

B. Other Specifications

1. pP

A decimal scale of 10^P is applied to all E and F fields following this specification.

$$\text{Internal number} = \text{External number} \times 10^P$$

The decimal scale continues to apply until another 'pP' is encountered, or another FORMAT statement is referred to, or an end of record is written.

2. Sx.y.z

This specification may follow any of the basic numeric field specifications, and indicates the occurrence of overpunching in the field. The digits x, y, and z refer to the principal part, exponent part, and B part of the field, respectively. 'x' is the number of the column within the field (numbered 1, 2, 3, ..., w from left to right) in which the overpunched sign of the principal part occurs. 'y' is the number of the column within the E part (with the column containing the 'E' numbered 0) in which the overpunched sign of the exponent part occurs. 'z' is the number of the column within the B part (with the column containing the 'B' numbered 0) in which the overpunched sign of the B part occurs. If the 'E' or 'B' do not occur in the field, the 'y' or 'z' are ignored. The absence of an overpunch implies a '+'. A field which is blank except for an overpunch in the principal part is considered to be a blank field.

Any of the values 'x', 'y', 'z' may be omitted from the specification, but the decimal points which define them must be specified. The following are correct specifications:

Ew.dSx, OwSx, Fw.dBbSx..z, Ew.dBbS..z, NwOSx

C. General

The order of the specifications must be followed precisely. Blanks in the FORMAT statement are ignored, except in the 'w' columns following a 'wH' specification. The numeric modifiers called for in the specifications may not in general be omitted. They may in some cases be negative; the sign must then precede the number.

Example: E10.-2B-20

Each specification in the FORMAT statement must in general be terminated by one of the following characters;

,	comma
/	slash
)	right parenthesis

or by the end of the FORMAT statement. The comma may be omitted after the specifications 'wH' and 'pP'.

A basic field specification, except 'wH' and 'wX', may be preceded by a positive non-zero number 'n'. The effect of this modifier is to repeat the field specification 'n' times. If 'n' is not specified, it is taken to be one.

Example: 3E12.5 = E12.5, E12.5, E12.5

Parentheses may be used to enclose a group of specifications which are to be repeated. The modifier 'n' precedes the left parenthesis, may not be omitted, and indicates that the group of specifications is to be repeated 'n' times.

Example: 2(I3, O5) = I3, O5, I3, O5

Parentheses may be nested to any reasonable degree.

Example: 2(I2, 3(F6.1, A2)) = I2, F6.1, A2, F6.1, A2, F6.1, A2, I2, ...

More than one data card may be described in a FORMAT statement. Specifications of succeeding cards are separated by a slash.

Example: 11H FIRST CARD/12H SECOND CARD/11H THIRD CARD

Repeated slashes cause the indicated data cards to be ignored.

Example: 11H FIRST CARD//11H THIRD CARD

Repetition of a data card specification is indicated by 'n*' at the beginning of the specification.

Example: .../2*6E11.4B35/... = .../6E11.4B35/6E11.4B35/...

Parentheses which enclose groups of field specifications must not enclose a slash. In other words, field specifications may be repeated only within a card. A single pair of parentheses, with the modifier 'n' omitted before the left parenthesis, may be used. Normally, when the

end of the FORMAT statement is reached and there is still data to be converted, the FORMAT statement is reentered at its beginning. However, if an unmodified left parenthesis was encountered, the FORMAT statement will be reentered at that point. This pair of parentheses may enclose slashes, but the left parenthesis must occur at the beginning of a data card specification, and the right parenthesis must be coincident with the end of the FORMAT statement.

Example: A60/(6I10/6O10) = A60/6I10/6O10/6I10/6O10/.....

An unmodified left parenthesis occurring at the beginning of a FORMAT statement is obviously redundant and is ignored. In this case only, more than one unmodified left parenthesis may occur in a FORMAT statement.

It is particularly important to ensure that every left parenthesis is matched by a corresponding right parenthesis.

D. Data Conversion

Leading and trailing blanks in a numeric field are ignored in conversion. An imbedded blank is an error. A blank numeric field is ignored, and no corresponding data is written on the output tape, but the nominal origin is incremented. If RETAK is used to read in data from cards containing blank numeric fields, the calling sequence must include the

MZE prefix (which causes RETAK to increment the starting location by the nominal origin). Any word which would ordinarily have been filled with data from a numeric field will be undisturbed, if the field was blank. If the PZE prefix is used, and blank numeric fields occurred in the data cards, the results are somewhat unpredictable.

THE OUTPUT EDITOR

The language of the Output Editor consists of eight macros which the programmer may use in his program to punch Hollerith and to print with full control of page headings, footings, spacing, conversion specifications, etc. The macros - XFORM, XPRINT, XPUNCH, XHEAD, XFOOT, XSPACE, XEJECT, XCOUNT - are expanded by the Compiler as calling sequences to an Output Editor supervisor. The supervisor, which occupies 100 words of core at execution time, interprets the calling sequences, outputting information onto the mediary output tape. During the output phase, this information is read by the Output Editor, and editing proceeds.

The Output Editor requires that OUTRAN be on the system tape.

1. XFORM N

The format of a line of print (or a card) is specified by a Format statement, the contents of which are compatible with those of the Input Editor and Fortran. Format statements are introduced by the macro

XFORM N

where N is the number of flag-words to follow the macro (and is, in fact, the number of Format statements to be introduced by the macro).

Each of the N words has the form

pfx L, T, M

where L, T denotes the beginning of the M words comprising a single format statement; and the prefix is PZE if L, T is direct address, MZE if indirect.

The M BCD words comprising a Format statement appear in core as if (or in fact) a direct result of

```
(L, T)   BCI   m1, A, XXXXXXXX
         BCI   m2, XXXXX...XXX
          .     .        .
          .     .        .
          .     .        .
         BCI   mn, XXXXX...XXX
```

where $m_1 + m_2 + \ldots + m_n = M$, and where A is a BCD character identifying the Format statement.

Unlike the Input Editor's Format statement, the character used to identify the statement may be any BCD alphabetic or $ (0 thru 9 will again be reserved for installation standard format conversion routines.)

Section 08.03 describes the basic field specifications permissible in a format statement.

Example:

To introduce two Format statements

1). A, 2I5 and

2). B, 3E14.6, 2(E9.4B5, F4.3B5) ,

we use

```
XFORM  2
PZE    ALP,0,2
PZE    BET,0,5   ,
```

and somewhere in our program we have

```
ALP BCI    2,A,2I5
BET BCI    5,B,3E14.6, 2(E9.4B5, F4.3B5)      .
```

2. XPRINT A, N, C

To specify that a block of data is to be converted and printed according to a given format, the macro

XPRINT A, N, C

is used,

where A is the identifying character of the Format statement describing the desired conversion; N is the number of flag-words which follow the macro, and C (which may be omitted) specifies that if fewer than C lines are left on the output page, the page should be ejected before the macro is executed.

The N flag-words which follow the macro have the same form as those following the XFORM macro, that is, PFX L, T, M. The location (L, T) referred to is to M words of data to be converted and printed according to the Format statement.

Example:

```
XPRINT  A, 2
PZE     DATA1, 0, 2
PZE     DATA2, 0, 4
```

specifies convert and print the2 words of data which begin at location DATA1 and the 4 words of data which begin at location DATA2 according to Format statement A (which is presumed to have been introduced earlier in the program by an XFORM statement).

3. XHEAD A,N,C

This macro instructs the Output Editor to convert according to format statement A the data specified in the following N flag-words and use it as the page heading for each page of print until suppressed by an XHEAD macro with a blank variable field or by an XHEAD specifying a new heading.

When the XHEAD macro is executed, the page is ejected if necessary (if the line count ≠ 0).

If C is non-zero, the number of lines per page is reset to C. (This is provided since it seems likely that in starting a new heading, the programmer is setting up a somewhat new page format, and might at this time want to reset the number of lines printed on a page. Spacing following the heading may be specified in the Format statement by multiple slashes. It may also be specified by a separate macro, XSPACE C,H, where C is the number of spaces to follow the heading before any subsequent printing. (See page 08.02.05)). Spacing after the heading is normally suppressed so, unless an overprinting is desired, spacing must be provided by one of the aforementioned methods.

4. XFOOT A,N,C

This macro is exactly parallel to XHEAD (above) and causes the information desired to be printed at the foot of the page (before page ejection). (Spacing before footing may be specified in the Format statement or by an appropriate XSPACE).

Note: Page ejection is the normal mode of operation; if it is suppressed, heading and/or footing are also suppressed.

5. XSPACE C,K

This macro is used to specify that there are to be C spaces of type K, where

K = H to specify spacing between the heading and the body of a print page,

K = B to specify spacing between blocks of printing in the body (a "block" of printing is defined to be the printed output resulting from one XPRINT macro),

K = F to specify spacing between the body and the footing, and

K = blank, for all three types of spacing.

XSPACE C,K is modal and will hold sway until another XSPACE of the same type is executed, wherein C may define a different number of spaces or may be blank or zero to discontinue the mode established.

The spacing specified by an XSPACE macro supercedes spacing specifications in a Format statement (see 08.03 for description of these).

6. XEJECT C

The XEJECT macro is used to direct immediate page ejection or to reset the standard number of lines to be printed previous to page ejection.

C = 0 or blank if the page is to be ejected immediately,

C = a decimal integer, if the number of lines per page is to be reset to C.

Interpretation of XEJECT C with C nonzero will cause the page to be ejected immediately unless the line count is zero (that is, unless the page has just been ejected).

An XEJECT C with C = -1 may be used to suppress page ejection altogether.

7. XPUNCH A, N

The XPUNCH macro is identically parallel to the XPRINT macro. XPUNCH A, N instructs the Output Editor to punch in Hollerith on-line (or , more normally, in column-binary-coded Hollerith off-line) the data indicated by the N flags following the macro, converting and formatting the data according to the Format statement whose identification is A.

8. XCOUNT N,S,I,R

To define a counter (as for page numbering) or a series or hierarchy of counters, the programmer may use the macro

XCOUNT N,S,I,R , where

N is the identifying number of the counter (may be 1 through 7),

S is the value to which counter N is to be set initially,

I is the amount by which counter N should be incremented (under control of Format statements -see below and 08.03), and

R is the identifying number of another counter which, when incremented, will cause the counter N to be reset to its initial value (R≠N).

The contents of a particular counter are incremented and displayed under the control of Format statements. Suppose, for example, that the macros

XCOUNT r,1,k and
XCOUNT n,s,i,r

had been executed previously. Suppose, further, that we are in the process of executing an XPRINT, XHEAD, XFOOT, or XPUNCH macro, and there is encountered in the referenced format statement the subfield

rCw+.

The current value of counter r would be printed (or punched) as a decimal integer (with sign) in the next w columns of the output record, the value of counter r would be incremented by the amount k, and finally, the value of counter n would be reset to s.(As described on page 08.03.03)

rCw specifies print (or punch) the value of counter r but do not increment it,

rC+specifies do not print counter r, but increment it (and thence, in our example, reset the value of counter n).

The example in section 08.04 should clarify the use of the XCOUNT macro in conjunction with Format statements.

It should be noted that the counters are limited in capacity (similar to the index register limit) to modulo 32,768.

FORMAT STATEMENT SPECIFICATIONS

As noted in Section 08.02, the parameters of the output macros (XPRINT, XPUNCH, XHEAD, and XFOOT), in addition to specifying the words in core to be transmitted, also specify the identifying name of a Format statement describing the type of conversion to be performed between the internal machine language and the external notation as well as the physical format of the line(card) or lines to be output.

The Format specification describes the line to be printed by giving, for each field in the line the type of conversion and the width (w) of the field.

A. Basic Field Specifications

Type	Description
Iw	The following w characters of output are to be the result of binary to decimal integer conversion. If w exceeds 11, only the 11 right-most characters will be significant, the rest will be blanks. The result will be right-adjusted.
Ow	The following w characters of output are to be the result of binary to octal conversion. If w exceeds 13, the excess will be blanks and the result right-adjusted. Leading zeros will be converted to blanks.
Ew.d.i	The next w characters of output are to be the result of converting the internal binary floating point number to decimal floating point, and the result is to have d places to the right of the decimal point and i places to the left.
Ew.d.iBb	Internal binary fixed point number to external decimal floating number. The binary number is assumed to have its binary point b places to the right of the leftmost bit. The result is to have d places to the right of the decimal and i places to the left.

Fw.d	The next w characters of output are to be the result of conversion from an internal binary floating point number to an external decimal fixed point number, where d is as in the E - type field.
Fw.dBb	Internal binary fixed point number to be converted to external decimal fixed point number, d and b are as in the E-type field.
Aw	The next $\frac{w+5}{6}$ data words contain BCD characters, and w of these characters are to be moved into the next w columns of the output image (are to appear as Hollerith characters in the next w output columns).
wH	The w BCD characters following the H are to be moved directly into the output image (are to appear as Hollerith characters in the next w output columns).
wX	The next w columns of output are to be blank.
Nw	The next w columns of output are to be the result of conversion from the internal signed binary integer to an external (signed) decimal integer which is to be used as nominal origin. Actually, the Output Editor simply treats this specification as if it were Iw. It is left to the programmer to so place it that it will serve him as a nominal origin.
NwO	See above specification. This type results in an octal number to be used as nominal origin.

The method for indicating repetition of a field specification, of a group of specifications, of the specification for an entire format line, as well as the separation of lines, indications of blank lines, etc. are identically as in the Input Editor, as are the means of indicating overpunching and scale factors. See Section 07.07 of this manual.

The format statements for input and output editors are compatible. Parameters required in the statement for one editor and not the other will be ignored by the second editor.

e.g. A format statement, for output, containing Ew.d.i (see above, floating point binary to floating decimal) could also be used intact as a format statement for the input editor. In this latter case, the i parameter is ignored and specification is treated as Ew.d (floating decimal to floating binary). (See 07.07.01, A.3)

B. PRE-LINE SPACING

If it is desired that a number of lines be spaced before printing a given line, the format specification of that line may be preceeded by a decimal integer from 1 to 9 indicating literally the number of spaces desired. This integer must preceed the first field specification of the line and must be separated from it by a comma. In the absence of a pre-print space request, the Output Editor assumes a one.

C. CONTROL OF COUNTERS IN THE FORMAT STATEMENT

Unique to the Format Statement of the Output Editor is a basic field specification,

cC c equal 1 thru 7

governing the treatment of the value of a counter, c, where this counter shall have been previously defined by an XCOUNT macro.

cCw specifies convert the value of counter c to decimal and print (punch) it in the next w columns of the output record.

cCw+ specifies convert and print the value of counter c, then increment it (by the amount indicated in the increment parameter of the XCOUNT macro defining counter c.

cC+ specifies don't print but do increment the value of counter c.

Note: In calculating the number of columns (w), a column must be allocated for the sign. e.g. to print 3 digits of a counter, 4 columns are required.

OUTPUT EDITOR EXAMPLE


```
                           1*        LET US ASSUME THAT WE HAVE A PROBLEM IN WHICH WE ARE
                           2*     ITERATING ON A 4-BY-7 MATRIX AND THAT WE WANT TO DISPLAY THE
                           3*     VALUES OF THE ELEMENTS AFTER EACH ITERATION.    IN THE SAMPLE
                           4*     PROGRAM WHICH FOLLOWS, EFFECTIVE USE IS MADE OF THE COUNTER
                           5*     DEFINING ABILITY OF THE EDITOR.

                           7              XFORM   2                   INTRODUCE AS FORMAT STATEMENTS
13631 -0 62500 0 00104        GO          STL     SYSOED
13634  0 00010 0 13703     8         +3   PZE     FORM1,0,8             1. THE 8 WORDS BEGINNING AT FORM1
13635  0 00013 0 13713     9         +4   PZE     FORM2,0,11            2. THE 11 WORDS BEGINNING AT FORM2.
                          10              XCOUNT  1,1,1          DEFINE COUNTER FOR PAGE NUMBER.
13636 -0 62500 0 00104               +5   STL     SYSOED
                          11              XCOUNT  2,1,1          DEFINE COUNTER FOR ITERATION NUMBER.
13641 -0 62500 0 00104               +8   STL     SYSOED
                          12              XCOUNT  3,1,1,2        DEFINE COUNTER FOR ROW OF ELEMENT.
13644 -0 62500 0 00104              +11   STL     SYSOED
                          13*                THUS, THE ROW COUNTER WILL BE RESET TO 1 EACH TIME THE
                          14*                ITERATION COUNTER IS INCREMENTED.
                          15              XCOUNT  4,1,1,3        DEFINE COUNTER FOR COLUMN NUMBER OF ELEMEN
13647 -0 62500 0 00104              +14   STL     SYSOED
                          16*                THE COLUMN COUNTER WILL BE RESET TO 1 EACH TIME THE ROW
                          17*                COUNTER IS INCREMENTED.
                          18              XHEAD   A                   PRINT AS THE OUTPUT HEADING THE
13652 -0 62500 0 00104              +17   STL     SYSOED
                          19*                INFORMATION SPECIFIED BY FORMAT A.
                          20              XSPACE  10,H
13655 -0 62500 0 00104              +20   STL     SYSOED
                          21              XSPACE  6,B
13660 -0 62500 0 00104              +23   STL     SYSOED

13663  0 60000 0 13727    24    BEGIN     STZ     MTRX                        INITIALIZE THE PHONY MATRIX.
13664  0 77400 1 00024    25         +1   AXT     20,1                        SET FOR 20 ITERATIONS
13665  0 50000 0 13726    26    AGAIN     CLA     L(ONE)
13666  0 40000 0 13727    27         +1   ADD     MTRX
13667  0 60100 0 13727    28         +2   STO     MTRX
13670  0 77400 2 00033    29         +3   AXT     27,2                        SET FOR A SINGLE ITERATION.
13671  0 50000 2 13762    30    ITER      CLA     MTRX+27,2
13672  0 40000 0 13726    31         +1   ADD     L(ONE)
13673  0 60100 2 13763    32         +2   STO     MTRX+28,2
13674  2 00001 2 13671    33         +3   TIX     ITER,2,1                    TEST ALL ELEMENTS IN MATRIX TREATED.
                          34              XPRINT  B,1,12              YES, PRINT ACCORDING TO FORMAT B THE
13675 -0 62500 0 00104               +4   STL     SYSOED
13700  0 00034 0 13727    35         +7   PZE     MTRX,0,28                   28 DATA WORDS BEGINNING AT MTRX.
13701  2 00001 1 13665    36         +8   TIX     AGAIN,1,1
13702  0 07400 4 00042    37         +9   TSX     SYSTEM,4
13703  2 17305 0 06773    38    FORM1     BCI     8,A,50X,16HMATRIX ITERATION,40X,5HPAGE 1C4+
13713  2 27305 0 06773    39    FORM2     BCI     9,B,50X,10HITERATION 2C3+////4*2,7(4H  X(3C2,1H,4C2+,2H)   *
13724  1 33103 7 30267    40         +9   BCI     2,=13,2X)3C+
13726  0 00000 0 00001    41    L(ONE)    OCT     1
                 13727    42    MTRX      BSS     28,0
                 13631    43              END     GO
```

6/15/60

A page of the output from the problem on the previous page.

ITERATION 13

X(1, 1)= 13	X(1, 2)= 14	X(1, 3)= 15	X(1, 4)= 16	X(1, 5)= 17	X(1, 6)= 18	X(1, 7)= 19
X(2, 1)= 20	X(2, 2)= 21	X(2, 3)= 22	X(2, 4)= 23	X(2, 5)= 24	X(2, 6)= 25	X(2, 7)= 26
X(3, 1)= 27	X(3, 2)= 28	X(3, 3)= 29	X(3, 4)= 30	X(3, 5)= 31	X(3, 6)= 32	X(3, 7)= 33
X(4, 1)= 34	X(4, 2)= 35	X(4, 3)= 36	X(4, 4)= 37	X(4, 5)= 38	X(4, 6)= 39	X(4, 7)= 40

ITERATION 14

X(1, 1)= 14	X(1, 2)= 15	X(1, 3)= 16	X(1, 4)= 17	X(1, 5)= 18	X(1, 6)= 19	X(1, 7)= 20
X(2, 1)= 21	X(2, 2)= 22	X(2, 3)= 23	X(2, 4)= 24	X(2, 5)= 25	X(2, 6)= 26	X(2, 7)= 27
X(3, 1)= 28	X(3, 2)= 29	X(3, 3)= 30	X(3, 4)= 31	X(3, 5)= 32	X(3, 6)= 33	X(3, 7)= 34
X(4, 1)= 35	X(4, 2)= 36	X(4, 3)= 37	X(4, 4)= 38	X(4, 5)= 39	X(4, 6)= 40	X(4, 7)= 41

ITERATION 15

X(1, 1)= 15	X(1, 2)= 16	X(1, 3)= 17	X(1, 4)= 18	X(1, 5)= 19	X(1, 6)= 20	X(1, 7)= 21
X(2, 1)= 22	X(2, 2)= 23	X(2, 3)= 24	X(2, 4)= 25	X(2, 5)= 26	X(2, 6)= 27	X(2, 7)= 28
X(3, 1)= 29	X(3, 2)= 30	X(3, 3)= 31	X(3, 4)= 32	X(3, 5)= 33	X(3, 6)= 34	X(3, 7)= 35
X(4, 1)= 36	X(4, 2)= 37	X(4, 3)= 38	X(4, 4)= 39	X(4, 5)= 40	X(4, 6)= 41	X(4, 7)= 42

99.01 APPENDIX A -- SYSTEM SYMBOL DEFINITIONS

Symbol	Definition
SYSAAC - FAC	Data Synchronizer channel activity control words.
SYSADP - FDP	Data Synchronizer channel dispatcher preference control words.
SYSAR1 - AR8	Locations of the I/O unit control words for the reserved tapes on channel A.
SYSASP - FSP	Data Synchronizer channel System priority control words.
SYSAU1 - AU8	Locations of the I/O unit control words for the utility tapes on channel A.
SYSBAD	Entry into the System Bad Spot Routine.
SYSBFD	Entry into the System Buffer Addition Subroutine - BUFAD.
SYSBKS	Entry into the System Logical Backspace Routine - BKSPC.
SYSBLK	Entry into the System Output a Block Flag Routine - BLOCK.
SYSBR1 - BR8	Locations of the I/O unit control words for the reserved tapes on channel B.
SYSBU1 - BU8	Locations of the I/O unit control words for the utility tapes on channel B.

SYSCAP	Entry into the System routine which prints a single line of commentary on the System printer.
SYSCHN	Cell containing the number of Data Synchronizer channels attached to the computer.
SYSCR1-CR8	Locations of the I/O unit control words for the reserved tapes on channel C.
SYSCU1-CU8	Locations of the I/O unit control words for the utility tapes on channel C.
SYSDAT	Cell containing the date in BCl format.
SYSDB1	Storage for the location counter by the first instruction of the Debugging macros.
SYSDB2	Entry into the Debugging Supervisor for interpretation and execution of Debugging macros.
SYSDIS	Entry into the System Dispatcher Recording Subroutine.
SYSDPI	Entry into the System Dispatcher Initiate Subroutine.
SYSDPS	Entry into Dispatching Suppression routines.
SYSDR1-DR8	Locations of the I/O unit control words for reserved tapes on channel D.
SYSDRA	Entry to the Buffering routine K$DRAIN
SYSDSB	Instruction to disable the Data Synchronizer Trap device, if monitor data channel trap routines are used. Otherwise, the instruction is NOP.
SYSDU1-DU8	Locations of the I/O unit control words for the utility tapes on channel D.
SYSENB	Instruction to enable the Data Synchronizer Trap device if data channel trap routines in monitor are being used. Otherwise it is a NOP instruction

SYSER1 - ER8	Locations of the I/O unit control words for the reserved tapes on channel E.
SYSERR	Entry into the System Unexpected Error Routine.
SYSESP	Not entry into the System extra-sensory perception subroutine.
SYSEU1 - EU8	Locations of the I/O unit control words for the utility tapes on channel E.
SYSFR1 - FR8	Locations of the I/O unit control words for the reserved tapes on channel F.
SYSFSI	Storage for floating spill indicators.
SYSFU1 - FU8	Locations of the I/O unit control words for the utility tapes on channel F.
SYSINF	Entry into System Output a Word of Information Routine - INFO.
SYSIOC	Entry into the System I/O check subroutine.
SYSIT1	Storage for the location counter by the first instruction of INTRAN macros.
SYSIT2	Entry into the INTRAN Supervisor for interpretation and execution of INTRAN macros.
SYSLER	Location of logical end of record control world for buffering routines.

SYSMIT	Location of the I/O unit control word for the System Mediary Input Tape.
SYSMOT	Location of the I/O unit control word for the System Mediary Output Tape.
SYSMTL	Entry into the System Mediary Tape Loader.
SYSNPT	Entry into the System Output N Logical Flags routine - NPUT.
SYSOED	Storage for location counter by first instruction of Output Editor macros. SYSOED+1 is the entry into the Output Editor supervisor for execution of Output Editor macros.
SYSORG	Cell containing origin for object programs. This origin is raised by System routines required by the object program at execution time (INTRAN, Debugging, etc.)
SYSOT1	Storage for the location counter by the first instruction of OUTRAN macros.
SYSOT2	Entry into the OUTRAN Supervisor for interpretation and execution of OUTRAN macros.
SYSPCH	Location containing the absolute address of the System punch.
SYSPEF	Location of buffering physical end of file control word.
SYSPER	Location of buffering phase control word.
SYSPIL	Entry into the System Floating Spill Routine.
SYSPIT	Location of the I/O unit control word for the System Peripheral Input Tape.

SYSPOC	Cell containing count of peripheral output lines.
SYSPOT	Location of the I/O unit control word for the System Peripheral Output Tape.
SYSPPT	Location of the I/O unit control word for the System Peripheral Punch Tape.
SYSPRT	Location containing the absolute address of the System printer.
SYSRST	Entry into the System Wotak Interrupt Routine - RSTOR.
SYSRTK	Entry into the System Input a Logical Record Routine - RETAK.
SYSRWD	Entry into the System Rewind Routine - REWIND.
SYSSTR	Entry into the System Store Location and Trap Routine.
SYSTAP	Location of the I/O unit control word for the System Tape.
SYSTAS	Entry into the System Tape Assignment Subroutine.
SYSTDC	Entry into the System Divide Check Routine.
SYSTEL	Location of the problem status indicators.
SYSTEM	Entry into the Supervisor at job end.
SYSTIM	Cell containing the time job processing was initiated.

The next twelve symbols refer to the Transmission Macros.

SYSTMA	Entry into the READ macro.
SYSTMB	Entry into the STEPR macro.
SYSTMC	Entry into the STEPF macro.
SYSTMD	Entry into the WRITE macro.
SYSTME	Entry into the WEOF macro.

SYSTMG	Entry into the BACKR macro.
SYSTMH	Entry into the BACKF macro.
SYSTMJ	Entry in to the BACKT macro.
SYSTMK	Entry into the IN macro.
SYSTML	Entry into the OUT macro.
SYSTMM	Entry into the RUSH macro.
SYSTMN	Entry into the DISP macro.
SYSTMO	Entry into the System Too Much Output Routine.
SYSTMT	Entry into the System Too Much Time Routine.
SYSTOF	Entry into the Floating Point Overflow Routine.
SYSTRC	Entry into the System Tape Redundancy Check Routine.
SYSTRA	Cell into which the Mediary Tape Loader places the address of an END or TCD card.
SYSTRI	Storage for STR indicators.
SYSTRP	Exit switch for normal trap.
SYSTST	Entry into the System Test Channel Status Routine - TEST.
SYSTUF	Entry into the System Floating Point Underflow Routine.
SYSWHT	Entry into the System Output a Non-block Flag Routine - WHYTE.
SYSWTK	Entry into the System Input a Word or Flag Routine - WOTAK.
SYSXTR	Entry into the System External Interrupt routine.

99.02 APPENDIX B - THE SYSTEM TAPE

In order to incorporate revisions to the various files of the System, the programmer responsible for the maintenance of SOS at his installation, should be familiar with the use of the WST (Write System Tape) control card, the operation of the WST file, and the format of the System Tape.

The SHARE Distribution Agency will distribute changes either in the form of a modification package to the current squoze deck of a particular file or in the form of a new squoze deck of that file. This recommends that installation-specific revisions also be accomplished by modifications to the current squoze deck, preferably using "CHANGE" rather than "ALTER" in effecting such modifications.

The squoze deck must be converted to 709-style absolute binary before being used to rewrite the System Tape. Modify and load provides for this conversion.

1. SYSTEM TAPE FORMAT

Each file on the System Tape begins with a one-word BCI file identifier for use by the System Tape loader routine, followed by the absolute code comprising that file. The code is divided into 256-word records, each record having its loading address in its first word. The last record of a file is a one-word transfer address corresponding to the transfer card of the absolute binary deck, followed by a physical end-of-file mark.

An outline of the System Tape format would look like this:

A. Initial End-of-File Mark

B. BCI File Identifier

C. Absolute code in 256 word records

D. Transfer address record

E. End-of-File

F. Terminal Check-Sum File

Repeated for each file on System Tape

(File Identifier is 777777777777)

2. USE OF SYSTEM TAPE WRITER

The WST File is brought into core whenever a WST control card is recognized, whether during Initiation phase or Input phase. The following control cards are recognized by the System Tape Writer:

A. CHANGE X,Y (X and Y are BCI File Identifiers)

The CHANGE card has two possible functions:

(1) If the variable field contains "X, Y", the files on the System Tape from the beginning of X to the end of Y would be deleted. For example,

CHANGE INTRAN, INTRAN

would delete that file from SYSTAP and any or all new files following that CHANGE card and preceding the next CHANGE or END card would be inserted at this point.

(2) If the variable field contains only "X", new files following that CHANGE card and preceding the next CHANGE or END card will be inserted following the file specified by "X".

B. CODE A

A is a file identifier of 1-6 BCI characters. The CODE control card is used to establish the file identifier (see 1-B above) for the subsequent absolute deck.

C. ROW

The ROWcontrol card indicates that the absolute deck following is in row binary. No equivalent control card is required if the absolute deck is column binary.

D. LAST

The LAST control card causes the System Tape Writer to write an End-of-File mark on the new tape.

E. END

The END control card indicates that the last modification file has been read and that the remainder of the input System Tape should be copied directly. The check sums written in the Terminal file will be recomputed before being written.

(Each of the five control cards described is punched in SCAT format with the 7-8-9 combination punch in column 1. The exception is that the variable field (of each control card requiring a variable field) must commence in column 16.)

The System Tape Writer, as presently origined, is capable of processing files of up to $44,000)_8$ words in length. In the WST process, files must be inserted and/or deleted in the order in which they appear on the input

System Tape. The primary reason for this restriction is the possibility of duplicates of the same file (using the same file identifier) appearing on the System Tape for the purpose of reducing tape search time. WST prints an ordered list of file identifiers for this purpose before the updating process is started.

EXAMPLES

Two examples of the input deck format for System Tape modification are furnished below:

A. The purpose of this deck is to replace the current files of M1 (LG) and INTRAN (INTRAN) with new versions thereof:

(1) WST

(2) Blank

(3) CHANGE LG, LG

(4) CODE LG

(5) ROW

(6) New row binary absolute Deck (with Transfer card)

(7) Blank (used in conjunction with ROW. Note below that this is omitted when the absolute deck is column binary).

(8) LAST

(9) CHANGE INTRAN, INTRAN

(10) CODE INTRAN

(11) New column binary absolute deck

(12) LAST

(13) END

B. The purpose of this input deck is to replace the current file of MG8(LG11), the Modify and Load Lister, with a new version and to insert the Output Editor (OUTED) on a tape that previously contained no version of the Output Editor. For this example, we are assuming that the most logical position for the Output Editor is immediately following SNAPTRAN, the Debugging Translator.

(1) WST

(2) Blank

(3) CHANGE LG11, LG11

(4) CODE LG11

(5) Absolute deck of M8 in column binary

(6) LAST

(7) CHANGE SNPTRN

(8) CODE OUTED

(9) ROW

(10) Absolute deck of Output Editor in row binary

(11) Blank

(12) LAST

(13) END

C. This input deck will reproduce a tape.

(1) WST

(2) Blank

(3) END

99.04 APPENDIX C - SQUOZE DECK FORMAT

A. Component sequencing	Required or Not
1. Preface	Yes
2. Heading Table	No
3. Macro Name Table	No
4. Blank	Yes
5. Macro Definitions	No
6. Introduction	No
7. Dictionary 1	Yes
8. Dictionary 2	Yes
9. Footnotes	Yes
10. Text without Commentary	No
11. Text with Commentary	No

Each section begins on a new card, and every card except the blank (which is entirely blank) has the following standard information punched in the first word (9 left or card columns 1-3)

Bits	Information
S	"1" (Indicating a squoze card)
1-5	Word count of this card
6-8	Card sequence number (upper part)
9-11	"101" (Indicating binary card)
12-23	Card sequence number (lower part)
24-35	Check-sum of entire card (except these bits)

B. Preface Card Format

Row or Column			Decrement	Address
9's	Left	1 - 3	(Standard format for all cards)	
9's	Right	4-6	Identification (6 BCI Characters)	
8's	Left	7-9	Identification (6 BCI Characters)	
8's	Right	10-12	Introduction	Head Table
7's	Left	13-15	Macro Name Table Word Count	Words in largest Macro definition
7's	Right	16-18	Not used	No. Dictionary entries
6's	Left	19-21	Relative location of END entry	Footnotes word count
6's	Right	22-24	Dictionary Reference size	Commentary Text word count
5's	Left	25-27	Highest Alter No.	Non-commentary Text word count
5's	Right	28-30	Not used	Macro definition word count
4's	Left	31-33	Compilation Date (6 BCI Characters)	

C. Heading Table Format

The Heading Table consists of one-word entries for each HEAD card, with the decrement containing the Alter number of the HEAD card and the address portion containing the heading character in base 50 encoding.

The Heading Table is omitted if no HEAD cards are present in the program. If the first heading character is employed subsequent to the first card in the deck, an entry will be made in the Heading Table with alter number zero and heading character "blank." The first actual HEAD card will then be encoded as the second entry in the Heading Table.

D. Macro Name Table Format

The Macro Name Table is comprised of a two-word entry for each programmer macro defined in the program. The first word of each entry contains the macro name in BCD (left-justified, with trailing zeros). In the second word of each entry, the prefix and tag, taken together, contain a count of the parameters specified in the definition of the macro, the decrement contains the number of words in the macro definition and the address contains the relative location of the macro definition.

E. The Blank card must always be present and must be totally blank. If a modification package is employed, it must be placed immediately before the Blank card.

F. Macro Definitions

Macro definitions appear in a form similar to squoze text. Successive definitions start in the next word after the end of a previous def-

inition and may extend from card to card. Macro definitions are omitted if no programmer macros exist in the program.

G. Introduction

The Introduction contains a one-word entry for each operation requiring an entry in the Introduction. These operations fall into two classes:

1. Generative operations (DEC, OCT, BCI, DUP, LBR, SQZ, System and programmer macros.)

   Sign = Zero

   Decrement = Alter number

   Address = Number of generated words minus 1.

2. Remarks and associated operations (REM, LIST, UNLIST, DETAIL, TITLE, SPACE and EJECT.)

   Sign = 1

   Decrement = Alter number of first of a continuous set

   Address = Number in continuous set

H. Dictionaries

A two-word dictionary entry is made for each location symbol in the object program. The dictionary references (and their associ-

ated location symbols) are chained together in a "next entry reference" method which is initiated by a special entry in each dictionary (entry reference 0) which refers to the first location symbol in the object program as its next entry reference.

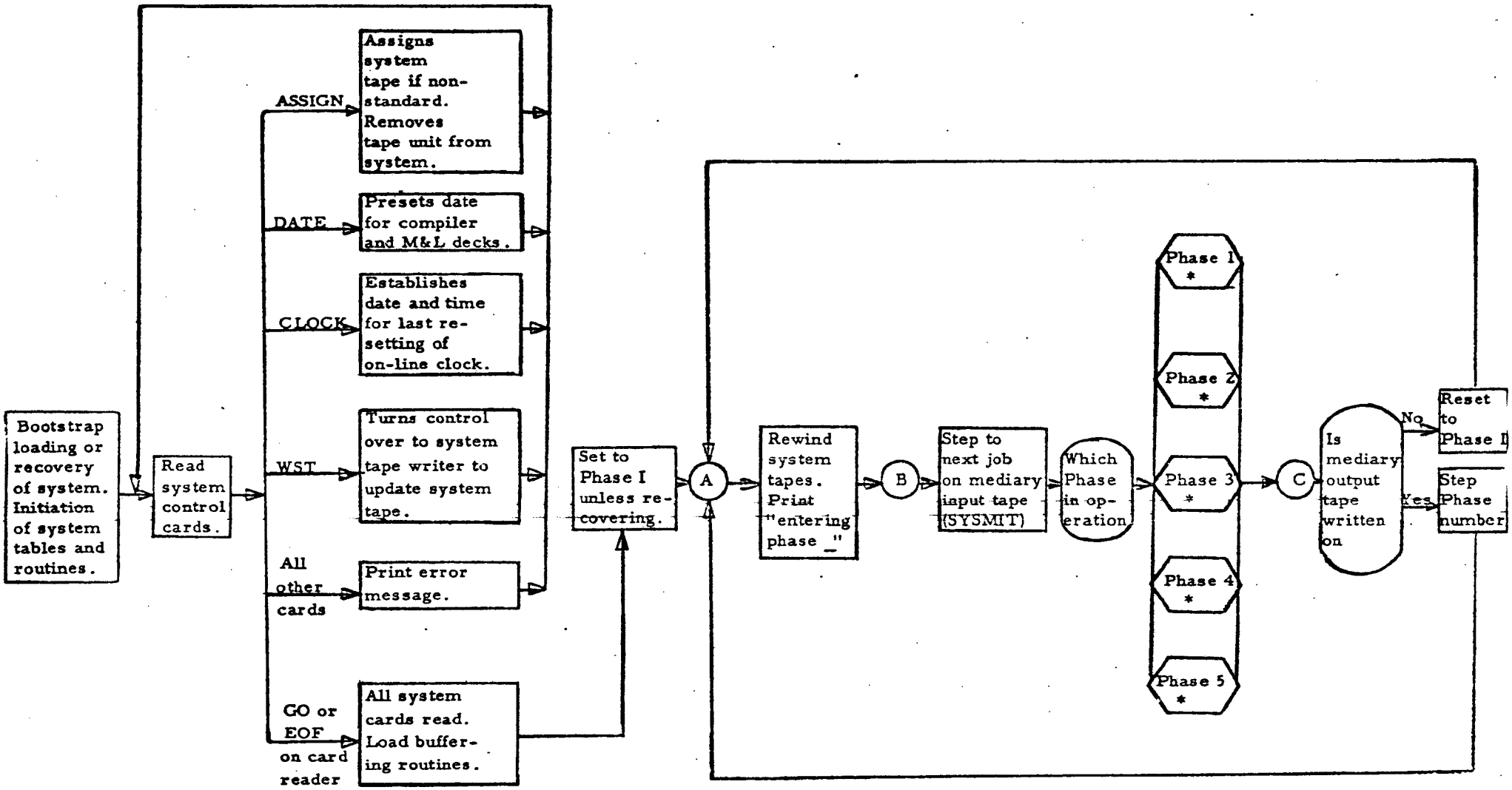
I. Footnotes

A footnote entry is made for each principle pseudo-op in the object program and consists of the variable field of the principle pseudo-op encoded in a form similar to squoze text. Each entry starts in a new word and occupies as many words as are necessary. Since every program must have an END card, footnotes are always present.

J. Text

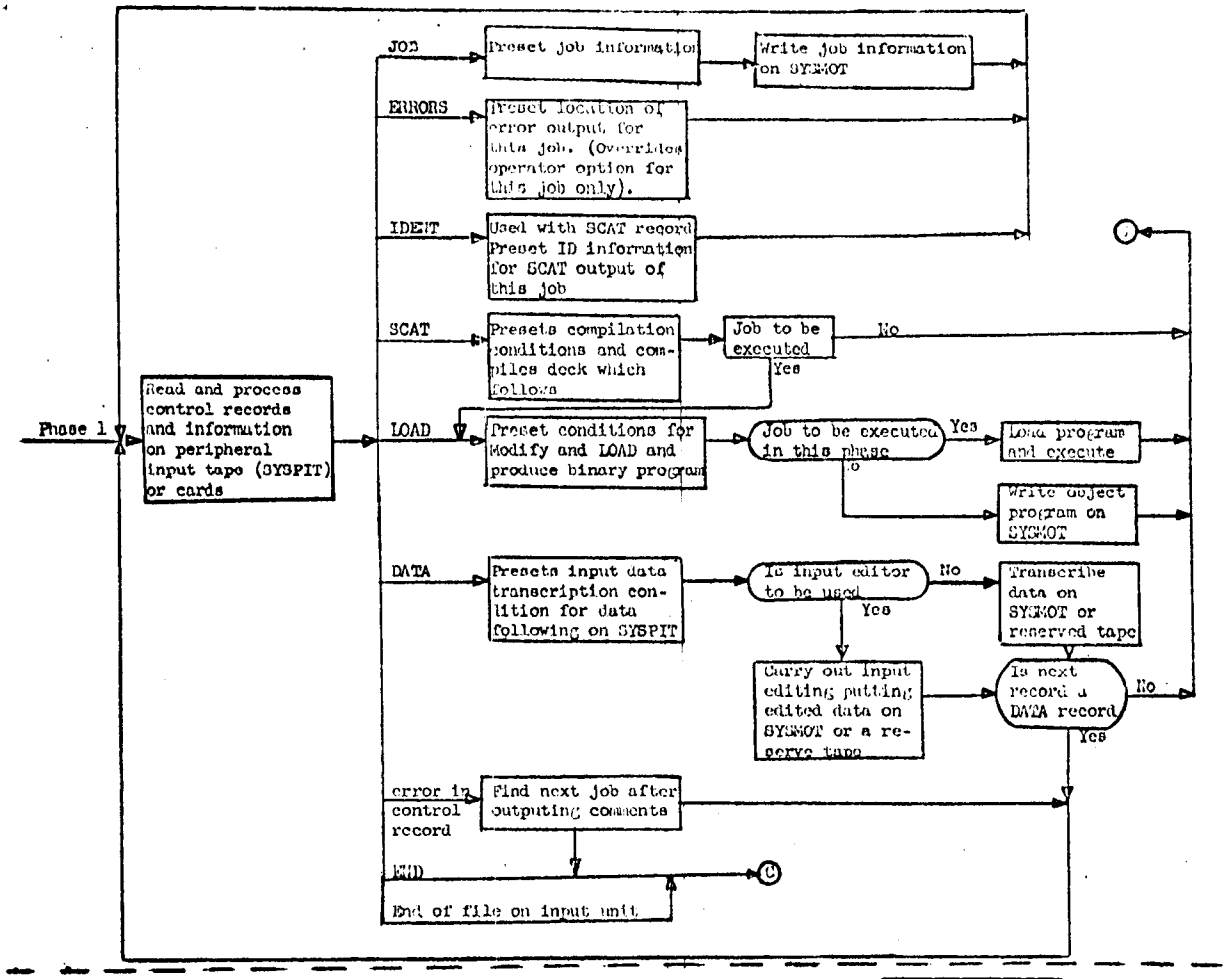
Each section of text consists of the object program entire encoded in a squoze form. The code is continuous through ends of words and of cards. Text without commentary differs from text with commentary only in that remarks attached to instructions and principle pseudo-ops are omitted. One section of text or the other, or both, must be present. Since manual decoding of squoze text is so laborious as to be impractical, a description of text format is not deemed necessary here.

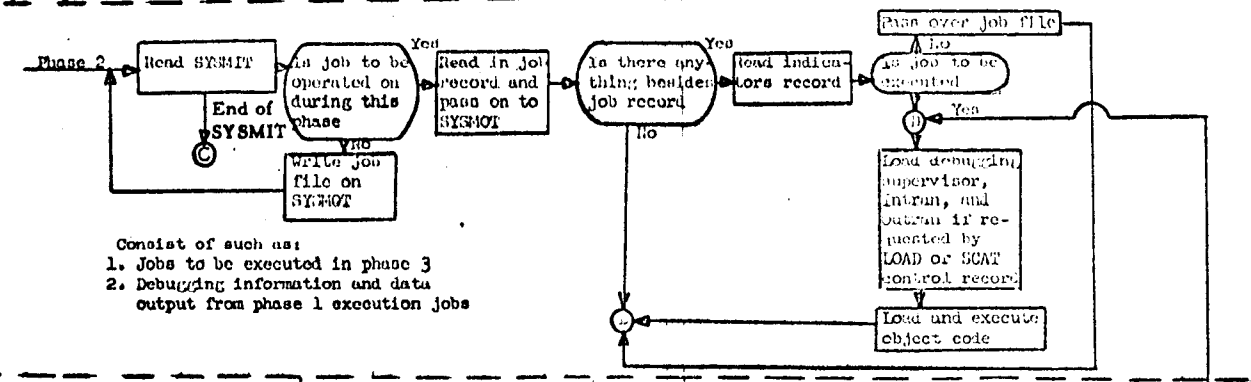
Mockdonald Monitor - Overall Flow Chart



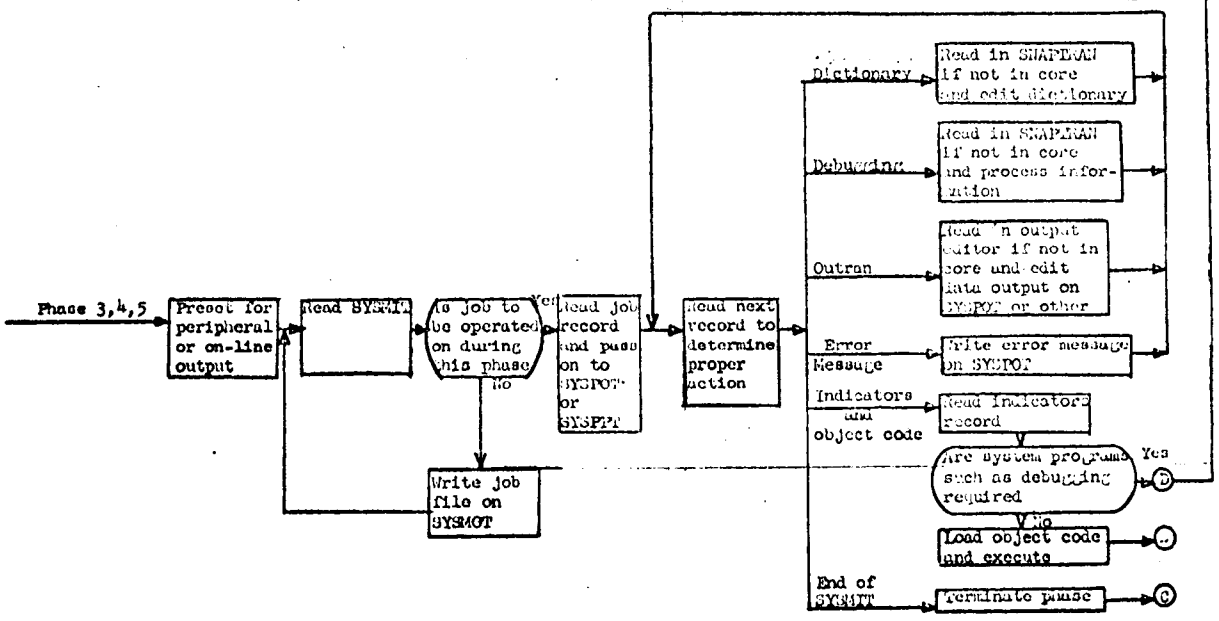
* All jobs on input tape are processed before exiting from phase. See next page.

99.04.06





Consist of such as:
1. Jobs to be executed in phase 3
2. Debugging information and data output from phase 1 execution jobs



Operator's Recap - MockDonald Monitor

Ready system tape on B1

Ready peripheral input tape in any on B2

Ready tapes B3, B4, B5, A1, A2, A3, A4, A5

If tapes not available, use ASSIGN codes in loader.

Sense Switches

File IDs	Input	Printed Output	Tape Assignment	Not Used	Card Output
Don't print	Tape	Tape	No		Tape
Print	Cards	Printer	Yes		Punch
1	2	3	4	5	6

Ready card reader with loader deck containing ASSIGN cards if necessary.

Place input deck after loader if input is from cards.

Press clear button and Load cards button.

If a stop occurs at 45 (41 prior to MD8) see printer for instructions.

If any other stop occurs, transfer manually to 46 (42 prior to MD8) to continue.

Should this fail, initiate the recovery procedure.

MockDonald Operators Manual

I A. Input Deck Makeup

The input deck consists of a stack of intermixed "Compiler" and "Modify and Load" jobs in any order. Each job deck will contain the program control cards necessary for that job. All program control cards will have 7, 8, and 9 punched in column one.

B. The Job Deck

Each job deck consists of a job card, a "Compiler" or "Modify and Load" deck, and possibly a data deck. The following table shows the makeup of each type of deck. The deck must contain cards in the order shown. All control cards with the exception of the ASSIGN cards should be furnished by the programmer.

All jobs must have a JOB card as shown.

Column 1	Column 8	Column 16	Remarks
789	JOB	Variable field	Variable field must not be blank.

The JOB card may be followed by ASSIGN cards.

789	ASSIGN	Variable field	optional See Appendix I

Jobs to be compiled will then have the following deck after the JOB and ASSIGN cards.

Column 1	Column 8	Column 16	Remarks
789	IDENT	Variable field	optional card, not required
789	SCAT	Variable field	card required, variable field may be blank
	blank card		optional
	symbolic card		required
	blank card		required

Modify and Load jobs will have the following deck after the JOB and ASSIGN cards

789	LOAD	Variable field	required card, variable field may be blank

blank card		optional
Squoze deck with modifications if any	must be column binary off-line	required

Either a compiler deck or modify and load deck may be followed by a data deck.. The makeup of the data deck will depend on whether or not the system input editor is used.

Data deck using input editor.

Column 1	Column 8	Column 16	Remarks
789	DATA	EDIT, Variable field	Variable field may be blank.
blank card data cards			optional
789	ENDATA		required
blank card			required

Data deck not using input editor.

789	DATA	NO EDIT, Variable field	Variable field may be blank.
data cards			

If the input deck is to be read in "on-line" through the card reader, data decks which do not use the input editor may contain either row or column binary cards in addition to Hollerith cards. Row binary cards must be preceded by a card containing a "ROW" in columns 8-10 and no other punches. The group of row binary cards must be followed by a blank card. Only one "ROW" card is necessary for each group of row binary cards. Column binary and Hollerith cards do not require any special control cards.

The stack of jobs may be feed one job at a time on line or off line to avoid later input deck separation. Thus EOF's will appear between jobs but will be ignored by the system.

An END control card must follow the stack of jobs previously described. This card must have 7, 8, and 9 punches in column one; and END punched in columns 8-10.

The stack of jobs constituting the input deck are the same whether read in "on-line or off" line except that row binary data cards not using the input editor may not be read in off line.

II A. Operating the System.

The system is put into operation by means of a Loader deck. This deck must be the proper one for the system tape used and may contain six or more cards. The last card will be:

789 GO

The field test tape is set up for a two channel machine with five tapes on each channel. Any variance from this configuration will require the use of ASSIGN CARDS placed before the GO card in the Loader deck. If there are additional tapes available on the machine an ASSIGN card of the form

789 ASSIGN XN=ON

will be required for each additional tape where X is the channel letter and N is the tape number. If five tapes are not available on each channel, an ASSIGN card of the form

789 ASSIGN XN=OFF

will be required for each tape not available. X and N are the channel letter and unit number.

The use of ASSIGN cards in this manner may vary from run to run depending on which tapes are down for maintenance. If five tapes are available on each channel, no ASSIGN cards are necessary.

All tape units on the machine are now mounted with tape and made ready. The units should be dialed from 1 on up consecutively. That is 1, 2, 3, 4, 5 on channel A and 1, 2, 3, 4, 5 on channel B. The system tape, SYSTAP, must be mounted on unit 1 of channel B. The peripheral input tape if any is mounted on B2.

The sense switches are now set as follows:

sense switch		
1	up	don't print file IDs on line
	down	print file IDs on line
2	up	input off line
	down	input on line
3	up	printed output off line
	down	printed output on line
4	up	no tape assignment changes (to be discussed later)
	down	tape assignment change

sense switch		
5	up	always
6	up	punched output off line
	down	punched output on line

The loader deck is placed in the card reader followed by the Input deck if sense switch 2 is down. The machine is cleared and the load cards button is depressed. After period of initialization the printer will indicate that the system will be indicated and a stop will be made with the location counter at 41. If no alteration of the tape assignments as shown are necessary, the operator presses START. As each job is started, the program control cards for that job will appear on the printer. If sense switch one is down, the file being read from the system tape will be printed on line. From time to time procedural instruction concerning the readying or removal of tapes will appear on the printer. Pressing START will continue operation of the system when the operator has insured that these instructions have been carried out where necessary.

If a request for tape assignment is made, the instructions as outlined in appendix 2 should be followed. After each assignment is made, the system will return to location 41 after printing the new status of all tapes on the machine. Tapes designated as free on the status list should be used to fulfill requests for tape assignment. Any illegal or unusual assignments will be noted by the system on the printer. When all assignments have been completed, put sense switch 4 up and press START.

Recovery

If for some reason the monitor should be destroyed during any phase of operation, it may be restored and operation begun with the next job to be processed. The procedure is as follows:

1. Look on the on line printer for the most recent tape status list that was printed. Note the tape assignments for SYSMIT and SYSMOT. Make up an ASSIGN card for each of these. You will then have

789 ASSIGN XN=SYSMIT

where XN is the channel letter and tape number for SYSMIT and

789 ASSIGN XN=SYSMOT

where XN is the channel letter and tape number for SYSMOT.

2. Remove the first three cards of the loader deck and replace with the recovery card.

3. Place two ASSIGN cards just produced in front of the GO card which is the last card in the loader deck. Follow this with the remainder of the input deck jobs if recovery is during input phase with input on line.

4. Hit the clear button and load cards button. Run all the cards in the revised loader deck through the card reader.

5. The system will be initiated and a stop will be made requesting the operator to place the phase number presently executing in the keys. This phase number is determined by looking at the tape status list noted previously. If input phase is indicated, this is phase 1. The execution phase is phase 2 and the output phase is phase 3.

6. After entering the phase number in the keys, press START. The printer will indicate that recovery has succeeded.

7. Restore the loader deck to its original form putting the first three cards back in and removing the two ASSIGN cards.

Note: This recovery card is good only if the system tape in on B1. If non-standard assignments have previously been made for peripheral input, output and punch tapes, it will be necessary to include ASSIGN cards for those as well as SYSMIT and SYSMOT.

Format of output tapes.

SYSPOT: A peripheral printer tape of one file containing job listings from Modify and Load and/or data output from the output editor. This tape is formed during the input or output phases, normally on A1.

SYSDOT: A peripheral printer tape of one file containing job debugging output from execution of jobs in any phase. This tape is formed during the output phase normally on A4.

SYSPPT: A peripheral punch tape containing one file of punch output from SCAT, Modify and Load or the Output Editor. Information on this tape is formed during the Input or Output phase normally on A2. Each job is preceeded by a replica of its job card (without a 7, 8, 9 punch in column 1) and followed by an interlaced card saying END.

Writing a System Tape

Refer to section 99.02.01 of the presently available "Programmer's Manual For The Share Operating System" dated October 13, 1959.

Appendix 1.

In some instances the programmer will need special tapes during the execution of his job. The person setting up the input deck (the setup man) must have information concerning the symbolic addresses of these tapes as used by the programmer. He must also know in which phase the programmer's job will be executed. The setup man may then assign any physical tapes on the machine to these symbolic addresses provided that tape is not already in use. There is no restriction concerning channel. This assignment is made with the use of Assign control card as follows:

789 ASSIGN XN SYSYYY

where SYSYYY is the programmers symbolic tape address such as SYSARI, SYSBU4, etc. XN is determined by the setup man where X is the channel letter and N is the tape unit on that channel. If the programmer is using a symbolic tape as input, it will be necessary for the operator to mount that tape on the unit specified by the ASSIGN card. For example a programmer specifies that he needs tape SYSBR3. If the setup man decides to use tape A4 to meet the requirements, the assign card would be

789 ASSIGN A4 SYSBR3

This card and any other required ASSIGN cards would be placed immediately after the JOB card for this job. An ASSIGN card is required for every symbolic tape unit used by the programmer.

Because of the importance of proper tape utilization to obtain maximum system efficiency it is suggested that a chart similar to the one to be illustrated be set up for each stack of jobs. As an example let us suppose there are a stack of seven jobs with the following requirements:

Job	Symbolic Tapes	Phase for Execution
1	SYSAR1, SYSBU1	3
2	SYSAR1	1
3	SYSAR4, SYSBR2, SYSAU8	2
4	SYSAU2	2
5	SYSBR2	1
6	SYSBR2, SYSAU4	2
7	SYSBU6	2
8	none	2

A table can then be formed on the basis of the phase in which the job wil execute. Normal system tape assignments are assumed. As shown on the table the system tapes used during phases I and III are rather extensive if all operations are to be off line. Thus the majority of the jobs should be executed in phase II. After filling in the system tapes which remain the same during an entire phase, we then set up the tapes required for each job execution so that the operator has time to mount and dismount tapes where necessary. If necessary the tapes assigned as erase tapes (SYSESN) may be used as utility or reserved tapes for phase 1 executions. The system will handle them automatically just as though these were free and available tapes.

Phase	Job No.	Tape No. A1	A2	A3	A4 *	A5	B1	B2	B3	B4 *	B5
		SYSPOT	SYSPPT	SYSMIT	SYSETP		SYSTAP	SYSPIT	SYSMOT	SYSETP	
I	2	SYSPOT	SYSPPT	SYSMIT	SYSETP	-	SYSTAP	SYSPIT	SYSMOT	SYSETP	SYSAR1
	5	SYSPOT	SYSPPT	SYSMIT	SYSET1		SYSTAP	SYSPIT	SYSMOT	SYSET2	-
II	3	-	-	SYSMIT	SYSAU8	SYSAR1	SYSTAP	-	SYSMOT	-	SYSBR2
	4	-	-	SYSMOT	-	-	SYSTAP	SYSAU2	SYSMOT	-	-
	6	-	-	SYSMOT	-	-	SYSTAP	-	SYSMOT	SYSBR2	SYSAU4
	7					SYSBU6		-		-	-
	8	-	-		-	-		-		-	-
III	1	SYSPOT	SYSPPT	SYSMIT	SYSDOT	-	SYSTAP	SYSAR1	SYSMOT	-	SYSBU1

ASSIGN cards for UTILITY and reserved tapes may refer to the same tape unit used as SYSES1 and SYSES2 even though the job is to execute in phase 1. Prior to starting the next job, the system will stop if tapes normally used as erase tapes were used on previous job as reserved or UTILITY tapes. Remove messages will be given if the tapes were used as reserve tapes and ready messages will be given for those erase tapes involved.

The ASSIGN cards would then be:

```
    JOB 1
ASSIGN B2=SYSAR1
ASSIGN B5=SYSBU1
    JOB 2
ASSIGN B5=SYSAR1
    JOB 3
ASSIGN A5=SYSAR1
ASSIGN B5=SYSBR2
ASSIGN A4=SYSAU8
    JOB 4
ASSIGN B2=SYSAU2
    JOB 5
ASSIGN A5=SYSBR3
    JOB 6
ASSIGN B4=SYSBR2
ASSIGN B5=SYSAU4
    JOB 7
ASSIGN A5=SYSBU6
    JOB 8
none
```

Appendix 2.

TAPE ASSIGNMENT

I. Standard Assignment of System Tapes

Tape	Phase 1	Phase 2	Phase 3	May be reassigned
SYSTAP	B1	B1	B1	By loader cards
SYSMIT	A3/B3	A3/B3	A3/B3	Initiation Time
SYSMOT	B3/A3	B3/A3	B3/A3	Between phases
SYSPIT	B2	---	---	Between phases
SYSPOT	A1	---	A1	" "
SYSPPT	A2	---	A2	" "
SYSDOT	---	---	A4	" "
SYSES1	A4	---	---	" "
SYSES2	B4	---	---	" "

II. System Tape Reassignment

Between phases the Monitor will halt after giving the necessary "Ready" and "Remove" messages and displaying the tape status list. At this time, the assignment of some System tapes may be altered (See Section I).

If the operator desires to change the assignment of a system tape, he must:

1. Depress Sense Switch 4.
2. Enter the prefix MTH (7) in the MQ Entry keys.
3. Ready the appropriate ASSIGN cards in the reader.
4. Bump START.

For example to ASSIGN a new SYSMOT the card would be:

789 ASSIGN XN=SYSMOT

An end-of-file or an error will terminate the processing of the ASSIGN cards.

If the Monitor requests assignment of a specific System tape, e.g.,

ASSIGN SYSMOT

the operator may use the method outlined above or may pursue the following course of action:

1. Put Sense Switch 4 down
2. Put PZE in the prefix of the MQ keys.
3. Put the physical address of the tape drive he wants to assign as SYSMOT in the address of the MQ keys.
4. Bump START.

The prefixes PON and PTW are meangless between phases.

III. Programmer Tape Assignment

A. Assignment of Utility and Reserved Tapes

An ASSIGN card of the form ASSIGN XN SYSXYN for each reserved or utility tape desired at execution time should be placed between the JOB and SCAT or LOAD cards. The information on the ASSIGN card is interpreted, encoded and transcribed on SYSMOT as part of the indicators record.

B. At execution time, the computer will halt after assignment of the reserved and utility tapes for the next job to be executed and after instructing the operator to:

READY XN SYSYYY

for each tape. If the tape assignments described are insufficient or erroneous, the operator may correct the conditions through Sense Switch 4, the MQ Entry keys and the Card Reader if desired.

C. Prefixes for Reassignment

	Key Word			
Function	Prefix	Address	Decrement	Analagous Card
General Form	1. PFX	A(physical address)	B(symbolic designation as if it were physical)	-
	2. PFX	A(physical address)	-	-
Assign Reserved Tape	PON	A	B	ASSIGN XN SYSYRM
Assign Utility Tape	PTW	A	B	ASSIGNXN SYSYUM

C. Prefixes for Reassignment (continued)

Function	Prefix	Address	Decrement	Analagous Card
*Assign System Tape	PZE	A	-	ASSIGN XN-SYSYYY
Disassign Reserved, Utility or System Tape	MZE	A	-	ASSIGN XN-ON
Disconnect Physical Unit for Maintenance	PTH	A	-	ASSIGN XN-OFF
Process ASSIGN cards in card reader	MTH	-	-	-
**Alter Sense Swtich Settings between Phases	MON	-	-	-
	MTW			DISPLAY TAPE STATUS

* This entry is made only upon request of monitor. An ASSIGN card must be used when the operator desires to initiate assignment of a system tape, and may be used instead of key entry in the cited case.

** A dummy function available in MD7 for the benefit of operators who do not get switches reset in preceding phase.

D. A programmer may if he desires, check to see if the operator inserted the necessary ASSIGN cards in his job deck. If an ASSIGN card was left out, the monitor will request that the assignment be made from the keys. The operator should put a physical tape address in the address field of the keys, depress sense switch 4, and press START. The operator should then put sense switch 4 up.

APPENDIX E

PART I: PRINTER CARRIAGE TAPE

In order to provide the system with the ability to perform rapid spacing, a method was developed whereby skipping could be performed by the printer. By an internally calculated combination of skips and spaces, any spacing requested can be accomplished in a rapid manner.

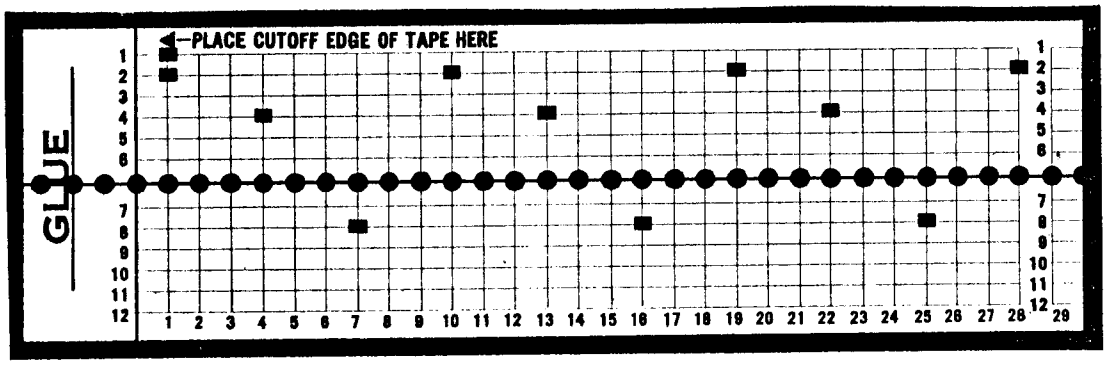
In order to achieve this result, a new carriage control tape is depended upon which contains specific channel punches.

This tape has a regular pattern:

Printing Line	Channel Punch
1	1 and 2
4	4
7	8
10	2
13	4
16	8
19	2

etc., to end of page or to line at installation eject point, i.e., 12 punch, if desired.

EXAMPLE:



The tape is to be used on both on-line and off-line printers. With the on-line printer, it is used with the Mockdonald printer board which is discussed immediately below.

PART II: MOCKDONALD 709-7090 PRINTER BOARD

A. FUNCTIONS AND USAGE

1. Sense Exit Functions.

SPRX	1	Wired directly to skip to channel 1.
SPRX	2	Impulse to set up exits 3, 4, and 5 for skipping mode.
SPRX	3	Extra space or short skip to channel 2.
SPRX	4	Double space or short skip to channel 4.
SPRX	5	Suppress spacing or short skip to channel 8.
SPRX	6	Optional
SPRX	7	Optional
SPRX	8	Conditions the printer to print successive blocks of 24 words, first in the leftmost 72 positions, then the rightmost 48 positions, continually until disconnected.* Normal overflow and spacing are suppressed before the right half cycle.
SPRX	9	Causes printing in rightmost 48 positions
SPRX	10	Optional

Note: If the proper exits are impulsed to cause a skip, the skip occurs only before the first line; succeeding lines are single spaced. If no skip is effected, the prescribed spacing (single or double, and/or extra) occurs with each line.

* When a sense exit of the printer is sent an impulse, the sense relays remain closed for a few milliseconds after the printer is disconnected. This is a printer characteristic. In the case here when sense exit 8 is sensed for a 12 word (or less) line and the printer is disconnected, if another print entry is given immediately, the line might be printed in the rightmost 72 columns of the same line if the sense relays were not yet opened. This condition can be remedied by sensing exit 8 only if the number of words to be printed is greater than 12.

6/15/60

2. General.

The Mockdonald printer board has been developed to effect faster and more efficient spacing. The board is compatible with the SHARE #2 board (with one minor exception which is discussed later) and permits skipping up to 9 spaces at a time, thereby reducing the time required to space more than a double space.

It is essential that the carriage control tape (see page 99.06.01) be used with the printer board. Once the position of the page is determined relative to the carriage tape, it is possible to skip, with or without additional spacing, to any desired new position.

In order to perform the skipping, the following sensing of the printer should take place:

```
      SPRX      2      Sets up skipping mode
      SPTX             Make sure impulse is present
      TRA       *-1    Not present
      SPRX      1      Yes, skip to channel 1 (eject)
or    SPRX      3      Skip to channel 2
or    SPRX      4      Skip to channel 4
or    SPRX      5      Skip to channel 8
```

An example of a skip to channel 8 and then a double space with the printer on DS Channel A might be:

```
WRS       UNIT     Select Printer
RCHA      Y, T     Give command
SPRA      2        Enter skipping mode
SPTA               Skipping mode impulse present?
TRA       * - 1    No
SPRA      5        Yes, skip to channel 8
SPRA      4        Double Space
 *                 Continue
```

3. Compatibility

For purposes of compatibility with the SHARE #2 board, it is possible to eject a page (skip to channel 1) when not in the skipping mode as previously described. Simply sensing exit 1 will effect a page ejection.

Sense exit 9 has been retained in the same capacity as on the SHARE #2 Board for compatibility. (This capacity is to set up control for the second print cycle in positions 73 to 120, and to suppress spacing for this second cycle). Sensing exit 8, on the MockDonald Board, during the first cycle performs the above function in addition to suppressing overflow and conditioning the printer to print positions 1 to 72 on the first cycle. Thus there are 2 methods of double cycling.

An example of printing a line of 120 characters might be:

```
WRS     UNIT
RCHX    Y, T    print 120 characters
SPRX    8       set up printer for 120 characters
```

This also might be accomplished by:

```
WRS     UNIT
RCHX    Y, T    print 72 characters
SPRX    8       print left half
WRS     UNIT
RCHX    Y, T    print 48 characters
SPRX    9       print on right half of line
```

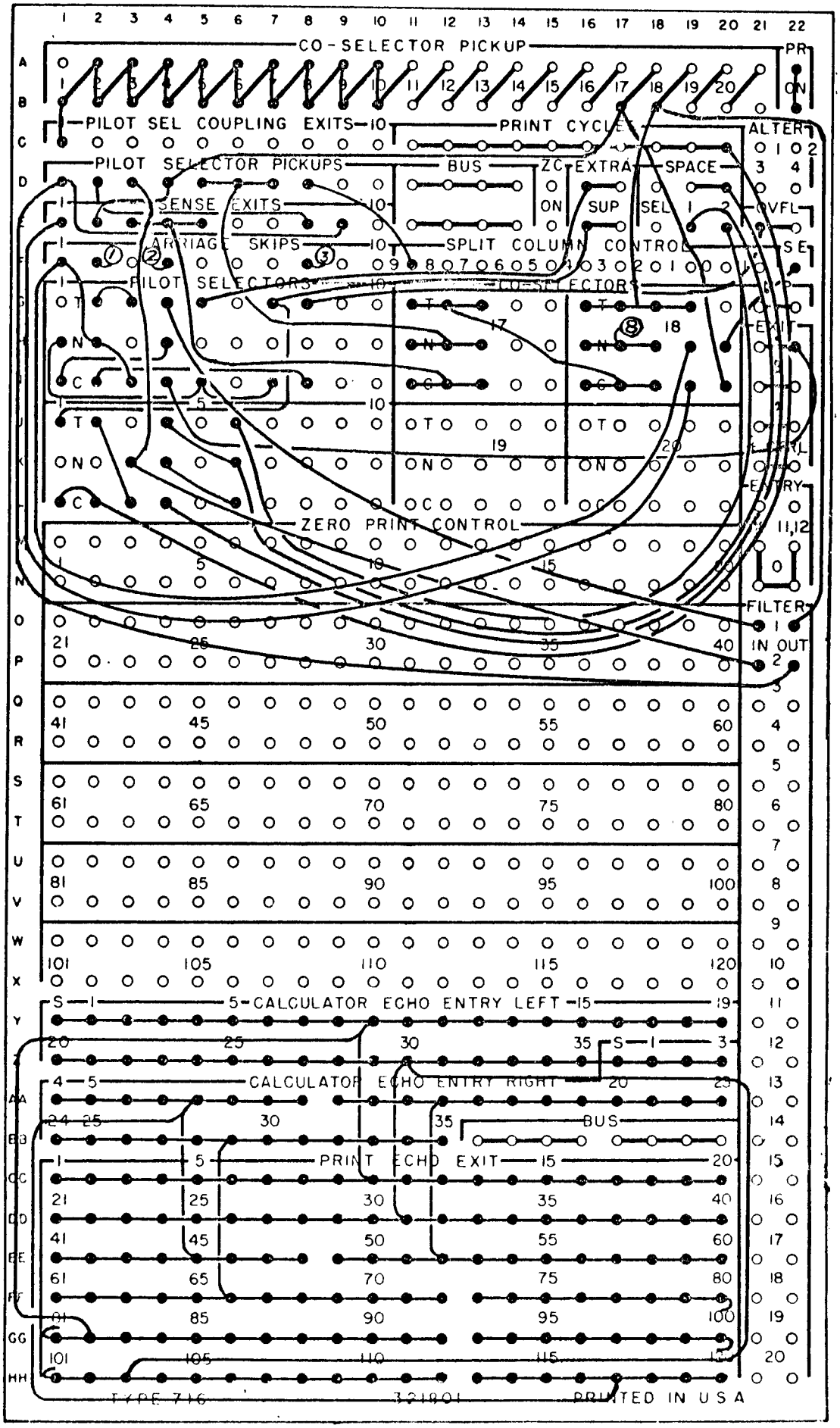
There is one minor point of non-compatibility between the MockDonald printer board and the SHARE #2 board. In order to permit the advantages of skipping, sensing exit #2 on the MockDonald board is necessary as the entrance to the skipping mode. This negates the ability to skip one-half of a page which, on the SHARE #2 board, could be accomplished by sensing exit #2.

4. Optional Sense Exits

Printer board exits 6, 7 and 10 remain available to the installation for clocking routines or any other purpose.

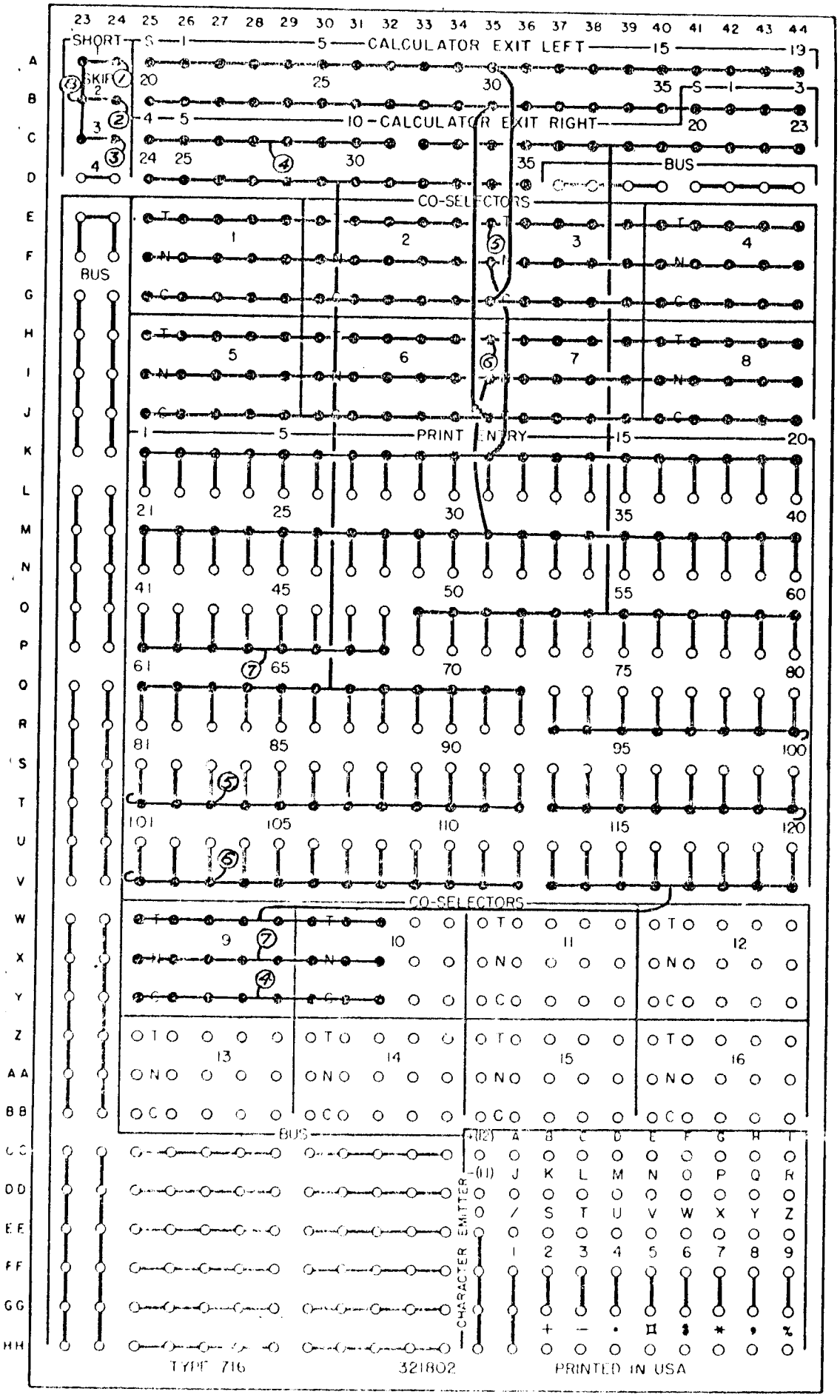
B. WIRING DIAGRAM FOR THE MOCKDONALD PRINTER BOARD

(Left Half)



MOCKDONALD PRINTER BOARD

(Right Half)



6/15/60

APPENDIX F - Notes Concerning Intran and Outran

To facilitate use by the Share Operating System (Mockdonald), several characteristics of Intran and Outran have been changed. In addition to the properties described in Part IV of the SOS Manual, the following information is pertinent to use of the translators in the Mockdonald system.

A. 1. OSCRIB(*), Y, T, C, N OUTPUT TRANSCRIPTION MACRO

Expansion:
```
STL   SYSOT1
TXL   SYSOT2,0,25
LDQ(*) Y, T
PZE   C, 0, N
```

where: Y, T = location of a symbolic unit name (e.g., SYSAR1)

C = number of words in OIMAG buffer
N = mode (see below)

OSCRIB, at present, has eight modes, namely --

STH	Storage to BCD tape
STHB	Storage to binary tape (converts BCD from OIMAG buffer to column binary image for off-line punching)
STHP	Storage to BCD tape with program control character inserted under control of OHEAD and OSPACE
STB	Storage to binary tape
SPH	Storage to on-line printer (assumes that the first character in the buffer is for carriage control, which is simulated)
SPHP	Storage to on-line printer with carriage control under control of OHEAD and OSPACE
SCH	Storage to on-line punch (BCD cards punched on-line)
SCB	Storage to column binary card (on-line punching)

At present, the normal modes of OHEAD and OSPACE are such that STHP and SPHP behave exactly like STH and SPH.

The choice of mode by OSCRIB is based on decoding bits from the absolute unit address found together with N in cases of ambiguity. The table below displays the possibilities:

Symbolic unit	N= 0	1	2	3
SYSPRT	SPH	SPHP		
SYSPCH	SCH	SCB		
Tape	STH	STHP	STB	STHB

2. OTPEND

If OSCRIB finds an end of tape while writing, it will end file and rewind the tape and return to the location set by OTPEND (normally to SYSERR) with the following error codes:

AC PUNIT,,CUNIT
MQ Garbage

3. OREDUN

Return will be made to the location set by OREDUN when

a. Unrecoverable bad spot while writing or tape check while reading:
(normal exit to SYSTRC) :
AC PUNIT,,CUNIT
MQ 0

b. C larger than buffer size set by OIMAG
(normal exit to SYSERR) :
AC CUNIT,,SCRIBC-OIMAGC
MQ 1

c. No unit assigned
(normal exit to SYSERR) :
AC PUNIT,,CUNIT
MQ 2

d. Illegal N or unit name
(normal exit to SYSERR) :
AC PUNIT,,CUNIT
MQ 3

In the above, CUNIT and PUNIT are the current and previous symbolic unit names used by OSCRIB. If OREADY was executed since the previous OSCRIB, CUNIT will be zero.

4. ISCRIB

A new parameter has been added to indicate the presence of look-ahead on the tape to be read. The macro now looks like

ISCRIB Y, T, C, L ,

where a non-zero L indicates that look-ahead is to be looked for.

B. Two new Outran macros have been included

1. OCOLC Y, T

This macro will place in location (Y, T) the present value of the column counter.

2. OFXFLO Y, T, C, N, K, B

This macro will convert a fixed point binary number to floating point decimal. Parameters are the same as those for OFLOAT except for the B which denotes the binary scale factor. This locates the binary point of the fixed point number in storage. This B parameter supplements the effect of an OPOINT. OFXFLO should be used similarly to OFLOAT and is subject to the same restrictions.

C. The B parameter has been added to the OFIX macro, resulting in

OFIX Y, T, C, N, K, B

where B denotes the location of the binary point in the word to be converted.

D. OFLFIX has been modified so that the maximum allowable output field is 16 characters instead of 8 as previously. Therefore any combination of integer and decimal places (N/K) not exceeding 16 will result in a converted number of the field size designated. If the sum of the N/K parameters exceed 16, a spill will occur with c(MQ) equal to 6, in accordance with OSPILL.

E. OSPILL ERROR RETURN

It is possible to return to the object program after an OSPILL error condition by setting an OSPILL macro to the location of coding which terminates with a TRA SYSOT1. This will return control to the object program at the location immediately following the macro which caused the error condition.

1. Normally, the field specified by the macro that spilled will be filled with X's. If X's are not desired, a non-zero value should be placed in the decrement of SYSOT1 before transferring to it. This will advance the column counter beyond the field, with nothing inserted.

2. At the time of the spill, the registers contain :

AC	decr.	column counter
	address	location of macro causing spill
MQ		type of spill (see OSPILL)

3. Analysis after the spill can be made before returning to the object program, but an Outran macro cannot be used. The use of such a macro would destroy preset conditions within Outran and a later return via SYSOT1 would fail.

INTRAN (GENERAL)

1. In conversion entries with N = 0 (implying scan to terminating character) if the scan goes past the last column, it will terminate with the last column number in the decrement and $(77)_8$ in the address of the AC. The number will be converted. Another entry with N = 0 will cause a transfer according to IEØR.

2. The decrement of SYSIT2 contains the address of a word looking like:

PZE BUFFER,2

where BUFFER is the address of the current BCD image.

3. An error return has been provided for ISPILL and ICHAR. To return to the object program in case of one of these errors, the programmer may transfer to SYSIT1 from the error routine which he has set up under an ISPILL and/or ICHAR.

When returning via this error return, two options are available:

(a) A zero in decrement of SYSIT1 results in zeros being stored in the word where the converted result was to go.

(b) A non-zero value in the decrement of SYSIT1 will pass over the word where the converted result was to go without altering its contents.

NOTE: When reentering INTRAN through the transfer to SYSIT1 described above, the values in the index registers will be saved by INTRAN and restored when control returns to the object program. Thus, if the index registers are to be used in the error routine of the programmer, and if the values put in during the object program are important, the error routine should save the values when entered from INTRAN due to ISPILL or ICHAR, and restore them before transferring to SYSIT1.

4. Two new areas have been incorporated under ISPILL. These are:

a. ISPILL with 7 in the MQ which denotes that an IFLOAT entry has resulted in underflow.

b. ISPILL with 8 in the MQ which denotes that an IFLOAT entry has resulted in overflow.

5. Nominal decimal and(for IFIX) binary scale parameters were added to IFIX and IFLOAT. The decimal (D) and binary (B) scale factors do not override, but in fact supplement the existing ISCALE and IPOINT modal values, respectively. The calling sequence of these two macro instructions are now:

a. IFIX(*) Y, T, C, N, D, B

```
STL     SYSIT1
TXL     SYSIT2,,7
STO(*)  Y, T
PZE     C, 0, N
PZE     D, 0, B
```

b. IFLOAT(*) Y, T, C, N, D

```
STL     SYSIT1
TXL     SYSIT2, 0, 6
STO(*)  Y, T
PZE     C, 0, N
PZE     D
```

With the inclusion of the B parameter in the IFIX macro, the relationship between the IPOINT macro and the B value in the data itself, has changed. Whereas, previously, the IPOINT value was overridden by the B value in the data, the two are now additive. The B value in the data, however, will cause the B parameter of IFIX to be ignored.

F. ERROR MESSAGES

Both INTRAN and OUTRAN are free from internal stops. The I/O manual states that some of the modal macros have as the normal case, a recognizable stop within INTRAN or OUTRAN as the case may be. These stops have been eliminated. The normal case of these modal macros will now result in the printing, on SYSDOT, of a message telling what modal macro has met an error condition, and which macro in the object program caused the condition. The contents of the AC and the MQ are printed out in octal for further analysis of the type of error.

Those modal macros which will print out this message when normal are:

INTRAN macros	OUTRAN macros
IBRNCH	OEOR
ICHAR(8)	OREDUN
IEOR	OSPILL
IFILE	OTPEND
IREDUN	
ISPILL	

APPENDIX G - Notes Concerning Debugging

To facilitate use by the SHARE Operating System, the TAPE macro has been changed as follows:

TAPE T, M, R, W, F, P

Expansion

```
STL   SYSDB1
TXL   SYSDB2, 0, 8
VFD   H3/M, 15/T, 6/0, H12/F
VFD   1/P, 2/0, 15/W, 3/0, 15/R
BCI   1, T
```

where

T is the symbolic name of the tape, e.g., SYSAR2

M = D for decimal mode
= B for binary mode (omission of subfield indicated by , , - will be interpreted as binary mode.)

R = -r to read forward r records
= -r to read backward r records
= ±0 to read until file mark encountered

W = -w to read w words at beginning of each record
= -w to read the last w words of each record

F = the format code to be used in listing

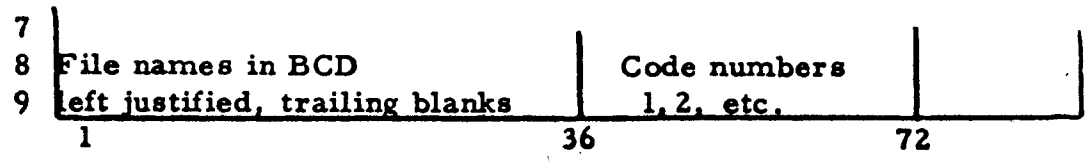
P = 0 or blank indicates that tape is to be repositioned
= 1 if tape is not to be repositioned

SYSTEM TAPE PATCH PROCEDURE

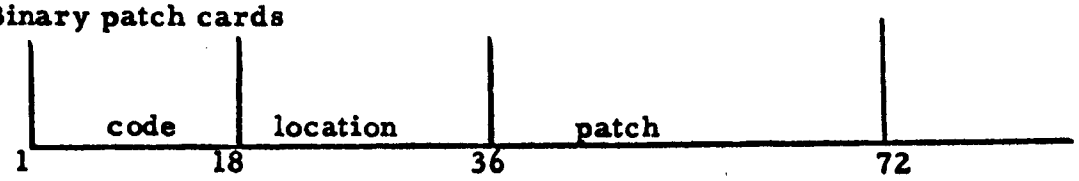
Beginning with SYSTAP 12/12/59 we have a simple and effective procedure for patching up to 12 system files without actually rewriting the system tape.

The system caller is expanded as follows:

1. 3 card loader
2. 1 card table of file names vs. code numbers for files to be patched.



3. Binary patch cards



4. End of patches card with 9's in CC 1, 2, 3.
5. ASSIGN cards, etc. as desired.
6. GO card.

NOTES:

1. File names vs. codes must be all on one card starting at 9 row and working up.

2. EXCCRD should not be listed in file name table. To patch this file use code 0 in patch cards.

3. More than one patch can be on a card. Use successive rows from the bottom.

4. To delete an obsolete patch which is on the same card with other good ones, make its code greater than 1000_8 and it will be ignored.

5. Up to 75 patches with non zero code numbers may be used. Any number with code zero are o.k. since they are made immediately.

SYSTEM PROCEDURE

The name vs. code table and the patches with codes are saved in low core. Each time that a system file is read from SYSTAP it is patched if patches exist for that file.

USERS PROCEDURE

1. Make all indicated entries in PATCH LOG, including new file name vs. code if necessary.
2. If new entry is required on name vs. code card, duplicate the old card before adding the new name to protect yourself in case of a punching error.

SUGGESTED PATCH LOG FORM

PATCH LOG FOR SYSTAP _____ Page _____
date

(Page one only)

File Name	Code	File Name	Code

Write card number, initials, and date on card also.

Card	Code	Loc.	PATCH WORD		Subr.	Initials	Date

SHARE OPERATING SYSTEM

PROGRAMMER'S MANUAL

Preliminary Version

Addendum # 6

Additional pages
Replacement pages
Corrections

8/12/60

Delete Pages

02.03.06
02.03.08
03.04.03

Replace Pages

Cover Page
2nd Page Table of Contents
02.03.02
02.03.04
02.03.05
02.03.06
02.03.07
02.04.01
02.04.02
03.02.02
03.03.02
03.03.03
03.03.04
04.01.01
04.01.02
04.01.03
05.01.02
05.01.03
05.01.04
05.02.09
05.02.11
05.02.13
06.01.10
06.02.02
07.01.01
07.02.01
08.01.01
08.02.04
08.02.08
08.02.09
08.03.03
99.01.02
99.02.05
99.05.00
99.05.01
99.05.02
99.05.05
99.05.07
99.05.08
99.05.11
99.05.12
99.07.06

Add Pages
03.03.05
04.02.02
04.02.03
99.09.01
99.09.02

Corrections
99.08.01
Line 16 should read "R = ±r to read forward ...."
Line 19 should read "W = ±w to read w words at beginning of each record."