Burroughs

B 500

Systems

MASTER CONTROL PROGRAM II

REFERENCE MANUAL

Burroughs
B 500
Systems

MASTER CONTROL PROGRAM II

REFERENCE MANUAL

Burroughs Corporation
Detroit, Michigan 48232

$5.00

Printed in U.S. America 2-72 1057205

Correspondence regarding this document should be forwarded using the Remarks Form at the back of the manual, or may be addressed directly to Systems Documentation, Sales Technical Services, Burroughs Corporation, 6071 Second Avenue, Detroit, Michigan 48232.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont)

LIST OF TABLES

INTRODUCTION

The Burroughs B 500 Disk/Tape Master Control Program (MCP) II is a multiprogramming or batch processing operating system that provides control and maintenance of the User Program Library for tape and or disk configurated systems. MCP II provides control for the execution of one, two, or three independent user programs on a multiprogramming basis.

The software and hardware requirements for the system are defined so that the user may better understand the Executive and Automatic System Control Functions. This includes memory allocation and automatic reassignment of actual to relative address at execution time, the creation of printer back-up tapes for the line printer, the ability to add or delete functions from the disk version of MCP II, and program segmentation.

MCP II also provides Sort Generation Functions for customizing sort programs. Advanced and Basic Assemblers are also provided in both the disk and tape versions that can be initiated manually with an appropriate function call to the Executive.

A section of the document emphasizes some of the programming techniques relating to MCP II, and the capability of including the COBOL Compiler as part of the disk version.

The Executive Routine is always resident in core memory when MCP II is in control of the system. The controller function maintains return linkages between multiprogramming programs. Linkage to the Executive allows for programmatic roll-in/or roll-out, e.g., interruption of processing, storage of the data in core memory on punch cards, disk or magnetic tape, loading of the called program into memory, or the recall and restoration of a stored program at the point of interruption.

At the conclusion of a program the End-of-Job Function will output an End-of-Job message to the system operator. The Executive will then loop continuously through an Interrogate Routine waiting for another job to be initiated.

As an user option, the Data Communication System can be interrogated for an input or output ready status. When a terminal is found in a ready status, the terminal number is stored within the Executive, and a user program. Either the DFI (input) or DFO (output) is automatically called by the Executive.

Peripheral assignment (when in the multiprogramming mode) is the responsibility of MCP II and is of no concern to the programming staff. Programs should be written to handle files via READ and WRITE macros, and not specific peripheral units. When a program is retrieved into memory, the peripheral requirements for that program are reserved. As each file is programmatically opened, the MCP will specifically assign unit(s) from the System I/O Table. Programmers should take into consideration, the sequence that magnetic tape files are declared in their program, and document the operating instructions in such a way that the operator may pre-mount his tapes when preparing for that particular run.

MCP II may be used in conjunction with the Supervisory Control Program (SCP) to provide control of On-Line Teller System installations.

The following is a list of MCP II capabilities:

a. Control for executing one, two, or three completely independent user programs in a multiprogramming mode.

b. Control functions are provided to process the following:

    1) End-of-job.
    2) Date assignment and changes.
    3) Overlay loading.
    4) Systems duplication.
    5) Saving and restoring of memory.

c. Executive routine control of the following I/O devices.

    1) Magnetic tape (BCL).
    2) Card reader.

3) Paper tape reader.
4) Card punch.
5) Paper tape punch.
6) Line printer.
7) Disk file.
8) Supervisory printer.

d. Memory allocation and automatic reassignment of actual address to relative address at execution time.

e. Creation of printer back-up tapes for eventual line printer output is provided automatically if such action is specified.

f. A function to print printer back-up tapes is also provided.

g. User Program Library creation and maintenance.

h. A capability of adding or deleting functions from the MCP disk version provides for minimizing disk storage requirements. This allows the user to customize the MCP to his particular installation.

i. ASOP Assembler Macros provide the user with multiprogramming I/O control and set-up procedures (includes blocking and unblocking of records). The multiprogramming macros are:

1) Begin run.
2) File descriptor.
3) Record descriptor.
4) File open.
5) File close.
6) Read.
7) Write.
8) Position (line printer and magnetic tape).
9) ZIP (to execute another program or function).
10) Accept (input from SPO).
11) Display (output on the SPO).

12) Multiprogramming interrupt.
13) Programmatic function call.
14) Load relative overlay.
15) Relative overlay.
16) Stop run.

j. Program segmentation based on macro statements.

k. ASOP, Advanced Assembler II, and B 500 COBOL Compiler provides for the automatic assignment of relative address codes, which allows the "floating" of programs within core memory.

SECTION 1
GENERAL SYSTEMS SPECIFICATIONS

GENERAL.

The MCP II is a software Operating System designed to provide the following capabilities:

a. Systems control for the following areas.

   1) Tape system control.

   2) Disk file system control.

   3) Executive control for multiprogramming.

   4) Executive control of input/output devices.

   5) Reassignment of actual address to relative address at execution time.

b. Creation and maintenance of a User Program Library.

c. Utility functions to provide many standard functions required by a user. The routines may be initiated through a function call from the card reader, supervisory printer or user program.

d. Sort functions that allows generating of specialized sorts.

e. Programs used in conjunction with the COBOL Compiler that allow the user to compile a program under control of the MCP.

f. Assembler functions that allow assembly of user programs under control of the MCP.

HARDWARE SYSTEM REQUIREMENT.

The following is the minimum hardware requirement for utilization of the operation system.

a. Central Processor (9.6 or 19.2K).

   1) Transfer Branch (TCB) option.
   2) Interrogate Package.

b. Supervisory printer.

c. Card reader.

d. Line printer.

e. One disk file module (or systems memory) and two, or three magnetic tape units.

NOTE

B 500 COBOL Compiler requires 19.2K memory.

SECTION 2
EXECUTIVE AND MANUAL CONTROL FUNCTIONS

GENERAL.

MCP II is designed around an Executive routine that resides in memory. The Executive provides program linking to MCP functions, and a continuous interrogating loop through the card reader, Supervisory Printer, and selectively a Data Communications System.

The functions are divided into two types: Executive and Manual.

EXECUTIVE FUNCTIONS.

The following functions are automatically called by the MCP to control the operation of a normal processing cycle. Executive functions cannot be called by the programmer, or operator.

The Executive Call functions are listed in table 2-1.

Table 2-1
Executive Calls

Call	Description
≥ ≥ ≥DKEX	Disk Version Executive
≥ ≥ ≥TPEX	Magnetic Tape Version Executive
≥ ≥ ≥EOJF	End-of-Job Function
≥ ≥ ≥DCKF	Date Check Function
≥ ≥ ≥NDAF	Numeric Date Assignment Function
≥ ≥ ≥ADAF	Alphanumeric Date Assignment Function
≥ ≥ ≥IFCC	Interrupting Function Call Check
≥ ≥ ≥FCDF	Function Call Check Function (Disk Version)
≥ ≥ ≥FC1F	Function Call Check Function-Table 1 (M.T. Version)
≥ ≥ ≥FC2F	Function Call Check Function-Table 2 (M.T. Version)

Table 2-1 (cont)
Executive Calls

Call	Description
≥ ≥ ≥FC3F	Function Call Check Function-Table 3 (M.T. Version)
≥ ≥ ≥OPNF	File Open Function
≥ ≥ ≥M/PC	Multiprogramming Controller Function
≥ ≥ ≥REJC	Reject Program Function
≥ ≥ ≥SAVD	Save Memory Function (Disk Version)
≥ ≥ ≥SAVE	Save Memory Function (M.T. Version)
≥ ≥ ≥MEMC	Memory Availability Check Function
≥ ≥ ≥FECH	Restore Memory Function (Disk Version)
≥ ≥ ≥GETS	Restore Memory Function (M.T. Version)
≥ ≥ ≥I/OC	Input/Output Configuration Check Function
≥ ≥ ≥IOFD	Input/Output File Declaration #1 Function
≥ ≥ ≥SPFD	Input/Output File Declaration #2 Function
≥ ≥ ≥EOPG	End-of-Program Function
≥ ≥ ≥SEOF	Standard End-of-File Function
≥ ≥ ≥ROLD	Relative Overlay Load Function (M.T. Version)
≥ ≥ ≥CLSF	Close File Function
≥ ≥ ≥DSCL	Disk Version Executive Card Loader Function
≥ ≥ ≥LTSC	Magnetic Tape Version Executive Loader Function

DISK VERSION EXECUTIVE (≥ ≥ ≥ DKEX).

The Disk Version Executive contains controls that allow in-process programs to exit to an End-of-Job Function, handle requests for

program overlays to be retrieved into memory, and a request for MCP functions to be loaded and executed. The Executive also retrieves object programs from the User Program Library, handles interrupts for operator intervention, or optionally interrogates for data communications.

When date constants are required by the object programs, Todays-Date and Report-Date stored in the DKEX Function are called.

When the SWITCH constant contained in the DKEX Function is set ON, it indicates that the program in process has interrupted another program. This program will automatically be restarted at the point of interrupt when an End-of-Job condition is detected in the interrupting program.

MAGNETIC TAPE VERSION EXECUTIVE ($\geq \geq \geq$ TPEX).
The magnetic tape version Executive like the disk version Executive contains controls that permit in-process programs to exit to an End-of-Job Function, and to handle requests for program overlays to be retrieved and executed. This Executive also retrieves object programs from the User Program Library, and handles interrupts for operator intervention. This Executive will not handle Data Communication devices..

A SWITCH constant contained within the Executive is tested to find out whether the Save Memory (SAVE or SAVD) had been performed prior to the loading of the current operating program. If the test is TRUE, the Restore Memory Function (FECH or GETS) will be called to reload the saved program into memory in its original condition and to resume the interrupted job.

END-OF-JOB FUNCTION ($\geq \geq \geq$ EOJF).
The End-of-Job Function is automatically retrieved into memory when the End-of-Job linkage is entered in the Executive. This condition occurs when an operating program transfers control to memory location 000, or when the Central Processor is cleared by pressing CLEAR and then CONTINUE.

The supervisory printer and card reader are interrogated to allow the systems operator to execute a function or a program. If the operator wishes to execute any of the MCP functions or a user program, the INPUT REQUEST key on the supervisory printer must be depressed (or the card reader made READY) in order for the MCP to receive a call. Once a call has been received by the operating system the EOJ Function returns control to the Executive.

DATE CHECK FUNCTION ($\geq \geq \geq$ DCKF).

The MCP will automatically call the Date Check Function to test the second word of all user programs. The DCKF Function determines if the code (in the second word of the user program) will cause a formatted date(s) constant to be constructed by the ADAF or NDAF Functions. User coding for the second word is found in table 2-2.

Table 2-2
Second Word User Coding

Word Position	Code	Remarks
0 and 1	+D	Date test positions.
2	Y	YES, todays-date is required, otherwise, leave blank.
3	Y	YES, report-date is required, otherwise, leave blank.
4	(Type)*	See NDAF and ADAF Function for codes.
5	(Type)**	See NDAF and ADAF Function for codes.
6 thru 8	(Location)	Memory location where todays-date is stored in the user program.
9 thru @	(Location)	Memory location where report-date is stored in the user program.

* todays-date
** report-date

The Date Check Function tests the Todays-Date and Report-Date parameters, and issues a call to automatically retrieve the Numeric (NDAF) or Alphanumeric (ADAF) Date Assignment Function. The dates will be formatted as specified by the type code and stored in the memory location specified. In addition the formatted dates will be displayed on the SPO (supervisory printer) for operator verification. A BOJ (begin run) message will be displayed on the SPO and a branch to the first instruction in the user program will be taken.

NUMERIC DATE ASSIGNMENT FUNCTION ($\geq \geq \geq$ NDAF).
The Numeric Date Assignment Function is automatically called by the DCKF Function to format and store date constants when a numeric or a julian-date is required by a user program. Either todays-date and/or report-date may be formatted by this function. The second word (positions 4 and 5) of the user program may contain any of the following codes to obtain the type of numeric formatting reflected by the following:

Type Code	Format	Type of Date
1	MM-DD-YY	Month-day-year
2	DD-MM-YY	Day-month-year
3	YYDDD	Year-julian-day
4	DDDYY	Julian-day-year

ALPHANUMERIC DATE ASSIGNMENT FUNCTION ($\geq \geq \geq$ ADAF).
The Alphanumeric Date Assignment Function performs exactly as the Numeric Date Assignment Function with the exception of formatting the dates.

Type Code	Format	Type of Date
A	Alpha-month DD, YYYY	Month-day-year
B	DD alpha-month YYYY	Day-month-year
J	Alpha-month DD, YYYY	Abbreviated-month-day year
K	DD alpha-month YYYY	Day-abbreviated-month-year

INTERRUPTING FUNCTION CALL CHECK (≥ ≥ ≥ IFCC).
The Interrupting Function Call Check Routine performs the following actions:

a. Determines the type of interrupt being initiated, e.g., operator, programmatic, or data communications.

b. Displays a SPO message reflecting the function being programmatically called.

c. Calls the Save Memory Function (SAVD or SAVE) to accomplish a roll-out of an operating program prior to calling the requested function.

NOTE
An area within the disk operating system has been reserved for the saving of programs.

d. Tests the Multiprogramming Flag to see if the function being called is consistent with the present operational mode (multi or non-multiprogramming), and displays a SPO message if an inconsistency exists.

e. Exits to either DFI or DFO, after setting up the proper overlay call for data communication interrupt.

FUNCTION CALL CHECK FUNCTION (≥ ≥ ≥ FCDF) MCP II DISK VERSION ONLY.
This routine is an automatic function that performs the tests necessary to determine the validity of an operator initiated function call. It determines if the Save Memory (SAVD) Function has to be called and will initiate the automatic call if required. The Function Table set up for the validating of functional calls is contained within the FCDF Function and has an unlimited capacity. Each table entry contains a 4-character function identity code, and a 7-character disk file starting address where the function resides. Nine table entries constitute a full Function Table segment, each of which are overlayable within the FCDF Function.

FUNCTION CALL CHECK FUNCTION - TABLE 1 (≥ ≥ ≥ FC1F) MCP II MAGNETIC TAPE VERSION.

This routine is an automatic function that performs the tests necessary to determine the validity of a function call. It will also test to determine if the Save Memory (SAVE) Function must be called, and will initiate the function when required. The Function Table set up for the validating of functional calls is contained within the FC1F Function and has a capacity of 20 functions. Each table entry contains a 4-character function identity code.

FUNCTION CALL CHECK FUNCTION - TABLE 2 (≥ ≥ ≥ FC2F) MCP II MAGNETIC TAPE VERSION.

This routine is an extension of the FC1F Function and provides an additional 20 table entries.

FUNCTION CALL CHECK FUNCTION - TABLE 3 (≥ ≥ ≥ FC3F) MCP II MAGNETIC TAPE VERSION.

This routine is an extension of the FC2F Function and provides an additional 20 table entries.

FILE OPEN FUNCTION (≥ ≥ ≥ OPNF).

The File Open Function is called by the Executive whenever a processing program requires the opening of a file. Programs to be multiprogrammed must communicate with the Multiprogramming Controller to obtain the OPEN Function, and the following actions will occur.

a. The I/O Control Segment and the Program I/O Table are retrieved.

b. The I/O unit and number are assigned by the MCP on a next unit-available basis.

c. The assigned I/O unit number is deleted from the system I/O Table to reflect that the unit will not be available for assignment until the file is closed or the program is discontinued.

d. The Multiprogramming Controller is recalled and if the specified type of I/O unit cannot be assigned by the I/O

File Declaration Function (≥ ≥ ≥ IOFD) an exit is initiated to the program END Macro.

e. A FILE OPEN message is displayed on the SPO.

O OMN file-name bbbbbbbeeeeeee

where:

1) O - open
OMN - the Operation Code, M and N variants of the file being opened.

2) bbbbbbb - beginning address of the pertinent file on disk.

3) eeeeeee - ending address of the pertinent file on disk.

f. The UNIT NOT AVAILABLE message is displayed on the SPO whenever the required unit is not available.

g. The O ERR file-name message will be displayed whenever an error is detected in attempting to open the file.

MULTIPROGRAMMING CONTROLLER FUNCTION (≥ ≥ ≥ M/PC).
The Multiprogramming Controller is called into the scratch-pad work area (machine location 490 thru 79@) whenever a program designed for multiprogramming (relative addresses) is loaded. The function is re-called following any interrupt condition which requires the use of the scratch-pad area. The purpose of the M/PC Function is to perform I/O operations, and set up and maintain return linkages to one, two, or three (maximum) multiprogramming object programs. The following messages may be displayed by this function:

a. OMN ER
A read/write error (ER) has occurred on the unit specified by OMN (OP Code, M and N variants). If the message persists, the operator must discontinue the program with (≥ ≥ ≥ DISC).

b. OMN NR

The unit specified by OMN has been tested and found to be NOT READY.

c. FN/O

A read or write operation was specified and the file has an unopened status. All files must be OPENed prior to performing an I/O function; therefore, a programmatic error exists and the program is automatically discontinued.

d. FC MP

A non-multiprogramming program (not relatively addressed) has been initiated and will not operate under the Multiprogramming Controller. The systems operator must restart the program after the multiprogramming schedule has been completed.

REJECT PROGRAM FUNCTION ($\geq \geq \geq$ REJC).

The Reject Program Function is called whenever a condition which prohibits the execution of a program is encountered. The REJC Function contained in the MCP II disk version will insert the first rejected call into a tank for automatic recall whenever the necessary I/O unit, available memory, or the number of programs being operated become less than the maximum number (three) allowed. The following messages may be displayed on the SPO during the Reject Function process.

a. Program-name NO MEM.

The amount of available memory is insufficient to process the program.

b. Program-name INV PT.

An invalid non-multiprogramming program has been called for execution. The systems operator must restart the program after the multiprogramming schedule has been completed.

c. Program-name MIX LT.

The maximum number of multiprogramming programs are presently operating in the mix.

d. Program-name NO I/O.
I/O units necessary to operate the program are not available.

e. Program-name TNK.
The program cannot operate at the present time, because the parameters are stored in the tank area. No action by the system operator is required. When the required I/O unit, the required memory, or the number of programs in the mix become less than the maximum, the program will be automatically recalled and executed.

NOTE

The tank area is capable of holding only one program name. Whenever multiple programs are called for execution and cannot be run, only the first program will be automatically recalled.

SAVE MEMORY FUNCTION ($\geq \geq \geq$ SAVD, MCP DISK VERSION) ($\geq \geq \geq$ SAVE, MCP MAGNETIC TAPE VERSION).

The Save Memory Function is used by the Executive to cause the roll-out of program(s) from memory to a reserved disk area, magnetic tape, or (optionally) cards: Whenever memory requirements are insufficient. The systems operator must be aware of the peripheral assignments for the program being called for execution so that a conflict of I/O unit usage between a called program and the roll-out program can be avoided. The following message may be displayed on the SPO by the MCP II magnetic tape version.

SAVE MEM-DESIG. UNIT

The systems operator must reply with one of the following unit designations to tell the MCP on which I/O unit to dump the contents of memory. The disk version will automatically store the contents of memory into its own area of disk.

where:

a. # - card punch.

b. 1, 2, 3, 4, or 5 - magnetic tape unit. A scratch tape with a write ring must be mounted on the designated physical magnetic tape unit. The function or program to be executed during the interrupt must not use this unit number.

MEMORY AVAILABILITY CHECK FUNCTION (≥ ≥ ≥ MEMC).

The Memory Availability Check Function determines whether or not a relative program can be processed at a given time within a multiprogramming schedule. It checks to see if the MCP is operating in a multiprogramming mode, whether the number of programs in the mix is less than the maximum, and if the program will fit into available memory. If all of the conditions are satisfied, the I/OC Function is called. If the tests fail, the REJC Function is called and the program does not enter the mix.

RESTORE MEMORY FUNCTION (≥ ≥ ≥ FECH MCP DISK VERSION) (≥ ≥ ≥ GETS MCP MAGNETIC TAPE VERSION).

The Restore Memory Function is automatically called by the Executive to restore memory. The following messages will be displayed on the SPO by the magnetic tape version. The disk version will cause the SAVD programs to be rolled into memory from MCP II reserve area.

a. LD CDS TO RESTORE MEM.

The program or function executed during an interrupt is completed. The MCP is ready to restore the interrupted program from the punch cards created by the SAVE Function. The systems operator must place the appropriate auto-load cards in card reader 1, and press START to reinstate the original program.

b. CD SEQ ERR.

Memory restoring auto-load cards are not in their proper sequence. Correct the sequence, and clear the read buffer by accomplishing the following action.

1) Note the location displayed in the Instruction Address Register (IAR).

2) Press CLEAR, LOAD, and CLEAR at the Central Processor.

3) Re-index the noted location into the IAR and press CONTINUE on the Central Processor.

After completion of the Restore Memory Function, the interrupted program is resumed.

INPUT/OUTPUT CONFIGURATION CHECK FUNCTION (≥ ≥ ≥ I/OC).
The Input/Output Configuration Check Function tests to insure that the I/O units required by a program are available for assignment. The following SPO message is displayed when the I/O units are not available.

program-name NO I/O

The program cannot be executed at this time. The program will be tanked and recalled when the required I/O units become available.

INPUT/OUTPUT FILE DECLARATION #1 FUNCTION (≥ ≥ ≥ IOFD).
The Input/Output File Declaration #1 Function stores the following information in the I/O Control Segments.

a. An Interrogate OP Code and M variant.
b. Program I/O Table position and length.
c. An appropriate I/O OP Code and M variant.

If a file requires the card reader or card punch, and the device has not been assigned to the program, a switch is automatically set to discontinue the program.

INPUT/OUTPUT FILE DECLARATION #2 FUNCTION (≥ ≥ ≥ SPFD).
The Input/Output File Declaration #2 Function sets up the linkage to cause a program calling for an invalid I/O unit to be discontinued. The SPFD Function will assign a file to either the line printer, or

optionally, to a magnetic tape back up if specified by the program. However, it should be noted that the COBOL program does not have the printer back-up capability. The file-name will be displayed on the SPO along with the assigned I/O unit. The following messages may be displayed on the SPO.

a. BOJ Program-name.
   The called program has been read into memory.

b. Program-name I/F S.
   An invalid I/O unit has been called by the operating program, and the program discontinued.

END-OF-PROGRAM FUNCTION (≥ ≥ ≥ EOPG).
The End-of-Program Function is automatically called at the EOJ of each program to remove it from the mix, and to restore memory and I/O units to the systems tables.

This function is called by the linkage inserted by the STOP RUN Macro or the COBOL Compiler. In addition, EOPG causes an End-of-Program message to be displayed on the SPO, recalls the M/P Controller if programs are multiprogramming, and exits to the End-of-Job Routine when there are no other programs in process. The following messages may be displayed on the SPO.

a. EOP program-name.
   This message signifies the end of a program.

b. PG ER.
   A MCP error has occurred. The EOPG Function is unable to determine which program to remove from the Memory Requirement Table. All jobs in process will be discontinued.

STANDARD END-OF-FILE FUNCTION (≥ ≥ ≥ SEOF).
The Standard End-of-File Function is called by the Executive when an end-of-tape, end-of-page, or an end-of-disk-area condition occurs during the execution of an I/O instruction specifying the standard EOF

address. The SEOF Function determines the condition that caused it to be called, and performs the following action:

a. End-of-tape.

1) Writes a tape mark.
2) Rewinds the magnetic tape.
3) Displays an end-of-tape message on the SPO.
4) Recalls the Executive.

b. End-of-page.

1) Skips paper to top of page.
2) Recalls the Executive.

c. End-of-disk-area.

1) Displays an end-of-area message (DSKLMTXXXXXXX) on the SPO. This message identifies the last disk segment read or written.

2) The program is automatically discontinued.

RELATIVE OVERLAY LOAD FUNCTION(≥ ≥ ≥ ROLD)MCP MAGNETIC TAPE VERSION. The Relative Overlay Load Function will load the first block of a relative program into a specified overlay memory area. It will store the program block count in a work area, and position the system tape to the first block of the program. This function conditions the Executive to load the program, and to reassign addresses.

CLOSE FILE FUNCTION (≥ ≥ ≥ CLSF).
The Close File Function is called by the Multiprogramming Controller when the files are closed by an operating program. The I/O units used by the operating program are released, and made available to the MCP for further assignment to another program. The FILE CLOSE message is displayed on the SPO and the Multiprogramming Controller recalled. The CLSF Function causes an exit to the END Macro if a file is not in

an OPEN status prior to being CLOSED. One of the following messages are displayed on the SPO whenever a file is CLOSED.

C OMN file-name

C OMN file-name PRTR TP

EXECUTIVE LOADER FUNCTION (≥ ≥ ≥ DSCL), MCP II DISK VERSION. The DSCL Function loads the Disk Executive from the disk file and transfers control to the DKEX Function. The operating instructions are as follows:

a. Load the MCP II disk version (if not already present) to disk file by using the manually initiated LDOP Function located in the tape version.

b. Load the Disk Executive Loader Control Card into memory from card reader 1.

c. Press CLEAR and CONTINUE on the Central Processor.

The format for the Disk Executive Loader Control Card is illustrated in figure 2-1.

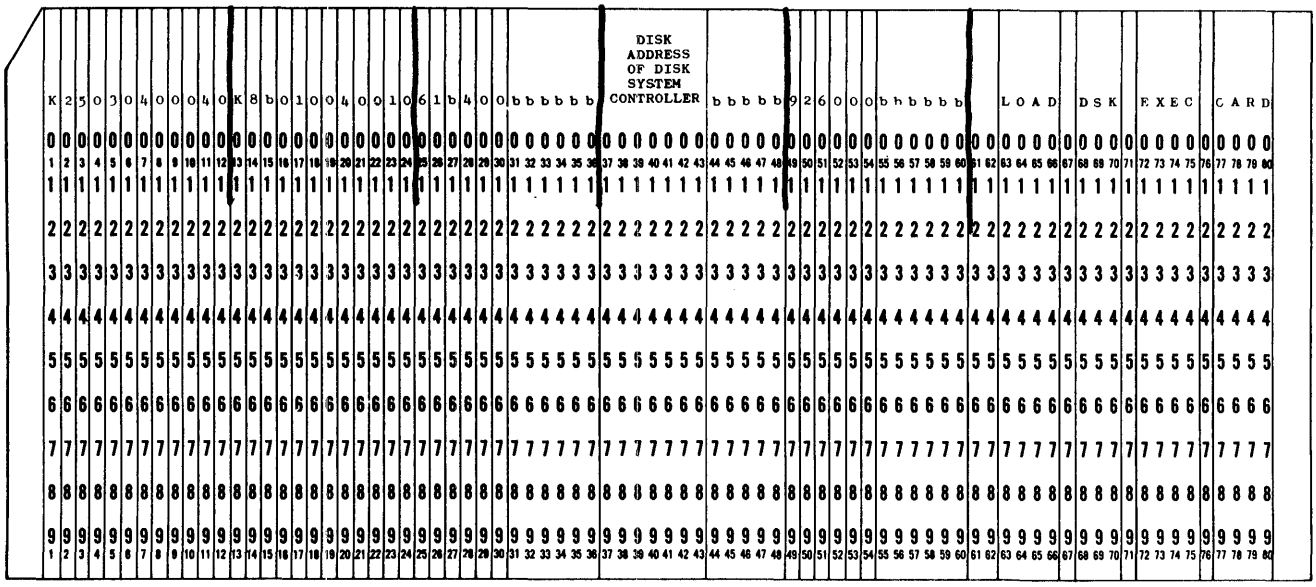


Figure 2-1. Disk Executive Loader Control Card

The Disk Executive Loader Control Card is automatically punched by the LDOP Function (reference item a).

The following control message is displayed on the SPO and indicates that the system is under the control of the MCP II disk version.

DISK EXEC CONTROLLER LOADED

MAGNETIC TAPE VERSION EXECUTIVE LOADER FUNCTION ($\geq \geq \geq$ LTSC).
The LTSC Function loads the Tape Executive from tape unit 6, and transfers control to the TPEX Function. The operating instructions are as follows:

a. Mount the MCP II systems tape (with no write ring) onto MTU-6.

b. Place the supervisory printer in REMOTE status.

c. Load Tape Executive Loader Control Card from card reader 1, and press CLEAR and CONTINUE on Central Processor.

The format for the Tape Executive Loader Card is illustrated in figure 2-2.

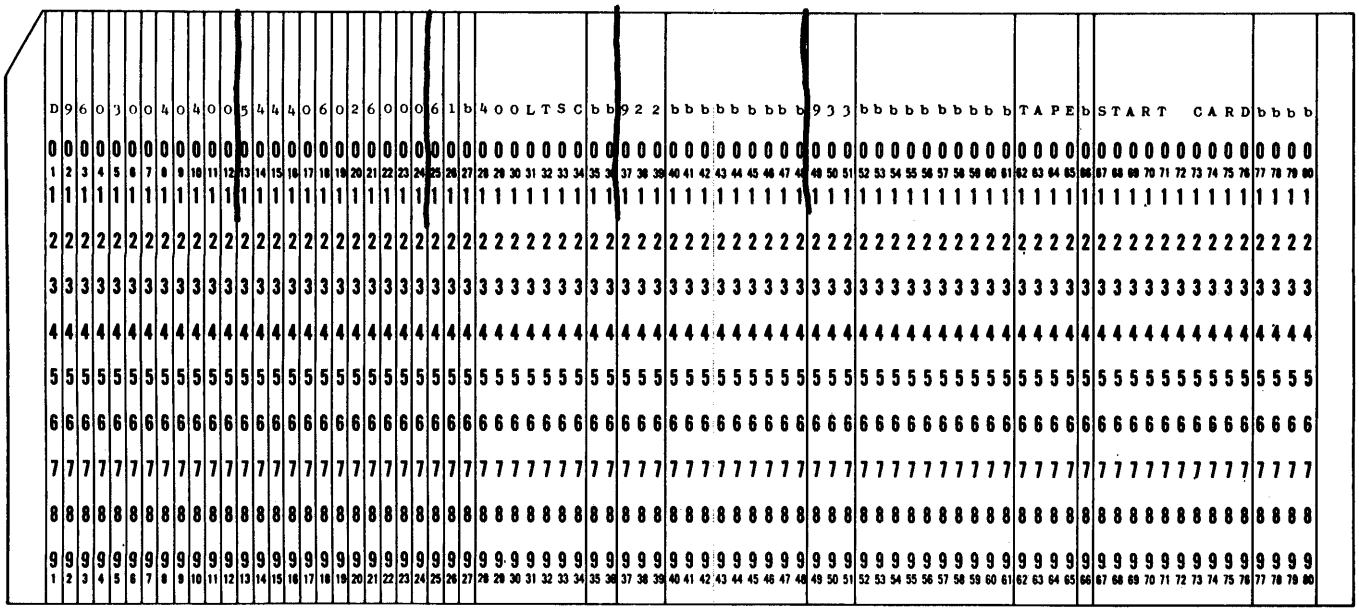


Figure 2-2. Tape Executive Loader Control Card

The following control message is displayed on the SPO to indicate that the system is under control of the MCP II magnetic tape version.

TAPE EXEC CONTROLLER LOADED END OF JOB-LTS

MULTIPROGRAMMING SYSTEM TABLES.

The Multiprogramming system utilizes numerous tables to perform its set up and control procedures. The tables are described in detail on the following pages.

Table 2-3

Flags and Table Positions

Machine Location	Length	Contents
005	Zone Bit	Discontinue Flag
005	7	Base disk address of MCP
016	6	Overlay linkage
028	1	Program/Function switch P=Prog F=Funct
030		Entrance for program interrupt
035	1	Tank Switch
036	6	Today's Date Storage
043	3	Return Address to Interrupted Program
108		Data Comm Interrogate Switch
115	5	Function/Program Ident Hold Area
11#	7	Disk Address of User Program Library
179	1	Interrupt Switch
220		Save Work Area
260	12	Program Table

Table 2-3 (cont)
Flags and Table Positions

Machine Location	Length	Contents
270	6	M/P Call Work Area
276	6	Report Date Storage
290	12	Systems I/O Table
31@	1	Number of programs in mix
356-387		Function call input area
361	5	Program/Function Ident
318	1	Multiprogramming Flag

SYSTEM I/O TABLE.

The System I/O Table is maintained by the Operating System, and it provides the MCP with the unit numbers of all the available units on the system. A function call (≥ ≥ ≥ IOTB) allows the user to specify or change the I/O configuration. The table is located at machine location 290.

When assigned to a program, the units are removed from the Systems I/O Table by the I/O Check Routine and are returned to the Systems I/O Table when the program reaches the EOJ Function. If a program is not terminated by the EOJ Function (for example - discontinued) the operator must return the I/O device to the I/O Table by calling the IOTB Function.

Position	Contents
1	Input reader 1. 1 - card reader 1 is available. A - paper tape reader 1 is available. Blank - not available

Position	Contents
2	Input reader 2. 2 - card reader 2 is available. B - paper tape reader 2 is available. Blank - not available.
3	Output punch. 0 - card punch is available. + - paper tape punch is available. Blank - not available.
4	Reserved for system.
5	Printer output 1. 1 - line printer 1 is available. Blank - not available.
6	Printer output 2. 2 - line printer 2 is available. Blank - not available.
7-12	Magnetic tape units 1-6 - available unit members. Position 7 is for unit 1. Position 8 is for unit 2. . . . Position 12 is for unit 6.

The contents of each position is set to either the unit number or to blank (if the unit is not available).

PROGRAM TABLE.

The Program Table is maintained by the MCP. Its purpose is to provide control information for each of the programs in process. The information determines if sufficient memory is available in consecutive locations for the loading of additional programs. The table format is as follows:

Position	Contents
1-2	Address Modifier (ADM) counter (the number of 480 character blocks the base address assigned) for the first program in memory.
3-4	Number of program blocks in the first program.
5-6	ADM counter for the second program in memory.
7-8	Number of program blocks in the second program.
9-10	ADM counter for the third program in memory.
11-12	Number of program blocks in the third program.

PROGRAM I/O TABLE.

The I/O Check Routine utilizes the Call Record I/O Configuration and the Systems I/O Tables to determine the availability of required I/O units. An I/O Table is created for each program. The Program I/O Table contains the I/O units assigned to the program. The Program I/O Table is used as follows:

a. To store the units deleted from the Systems I/O Table prior to their assignment to a program.

b. Upon execution of file OPEN, the units are deleted from the program I/O Table and assigned to specific files. Storage for I/O commands and related information is provided in each files I/O Control Segment.

c. Upon execution of file CLOSE, units are reassigned to the Program I/O Table. The units may be reused by the program through the execution of additional file OPENS.

d. When the units are released by the program through execution of a file CLOSE and End-of-Job Function, the units are reassigned to the Systems I/O Table.

Position	Contents
1	Input reader 1. 1 - card reader 1 is assigned. A - paper tape reader 1 is assigned. Blank - not assigned.
2	Input reader 2. 2 - card reader 2 is assigned. B - paper tape reader 2 is assigned. Blank - not assigned.
3	Output punch. 0 - card punch is assigned. + - paper tape punch is assigned. Blank - not assigned.
4	Reserved for system.
5	Printer output 1. 1 - line printer 1 is assigned. Blank - not assigned.
6	Printer output 2. 2 - line printer 2 is assigned. Blank - not assigned.
7-12	Magnetic tape input/output. 1-6 - assigned unit numbers. Position 7 is for unit 1. Position 8 is for unit 2. . . . Position 12 is for unit 6.

The contents of each position is set to either the assigned unit number or to blank (if the unit is not assigned).

I/O CONTROL SEGMENT.

An I/O Control Segment is created by the assembler for each I/O file. The purpose is to provide control information to the Multiprogramming Controller for the execution of input/output commands.

The I/O Control Segment format is as follows:

Position	Contents
1-3	I/O error address - set by the user's READ or WRITE Macro. If an user address is not supplied, a standard error routine address is inserted by the assembler.
4-6	End-of-File return address - inserted by the user's READ or WRITE Macro. On output files it is the end-of-tape or the end-of-page address.
7-9	Tape backspace code - inserted by the File Open Routine. It is used as a storage area for the disk file address.
10	I/O n variant storage - used by the File Open Routine to store the variant contained within the program I/O Table.
11-12	I/O code storage for printer tape backup - the I/O File Declaration Routine moves the I/O code to these positions when a printer tape is assigned in lieu of a line printer.
13-14	Reserved for expansion.
16-17	Interrogate OP Code and M variant - supplied by the I/O File Declaration Routine.
18	Maximum number of units for the specified type interrogate n variant - the maximum number of units

Position	Contents
	is set by the I/O File Declaration Routine. The File Open Routine replaces this number with the interrogate n variant.
19-21	Program I/O Table base address for units of the specified I/O type - inserted by the I/O File Declaration Routine.
22-23	I/O command - OP code and m variant are supplied by the File Open Routine. The A, B, and C address is set by the assembler. A - Error address/output area. B - End-of-file address. C - Error address/input area. If the I/O type is a disk file, the assembler sets the addresses as follows: A - Address of the disk file address. B - Input/output address. C - Not ready address (MCP).
34-36	End-of-program linkage - inserted by the assembler, it is a fixed label and must be PRGEND.
37-39	Retry linkage address - I/O execution linkage address. It is inserted by the assembler.
40-42	Return address - inserted by the assembler. Control is returned to this address after the execution of the I/O command, and after one cycle through each program in the mix.

MULTIPROGRAMMING FLAG TABLE.

This is a one position table located at machine location 318, and it is used to indicate the type of processing in effect. The table may contain the following:

0 (Zero) - Multiprogramming in process.

1 - Machine language program or function in process.

BLANK - Neutral-set by End-of-Job Routine.

MULTIPROGRAMMING COUNTER TABLE.

This is a one position table located at machine location 31@, and it contains the number of multiprogramming programs in the process. It may contain either 0, 1, 2, or 3. Its purpose is to determine if additional programs can be loaded. It is also used by the End-of-Program and Discontinue Routines to determine if the End-of-Job Routine or the Multiprogramming Controller needs to be called.

DISCONTINUE FLAG.

The Discontinue Flag is used to halt a program or function. It may be programmatically set by the transfer of a B-bit. The flag is located at machine location 005. If on, the program will abort and the discontinue message and program identification is printed on the SPO.

DISK ADDRESS OF OPERATING SYSTEM.

The base disk address where the MCP was loaded. The address is stored in a seven position location starting at machine location 005. This address is used as a base to calculate the disk address of the Operating System Functions. Within each automatic function is a constant that is computed and added to the base address. It determines the disk address where that function will reside on disk. For this reason the automatic functions should never be deleted from MCP II or rearranged.

OVERLAY LINKAGE.

Overlay linkage is used to store the overlay name and address when programmatically calling for user overlays. This six position storage address is located at machine address 016.

PROGRAM/FUNCTION SWITCH.

The Program/Function Switch located at machine location 028 is used to determine the type of program in execution. It is set to P when a program is executed, and F when a function is executed.

INTERRUPT.

A program interrupt is programmatically achieved by transferring the program return address into machine location 043, and branching to 030. The Executive checks for DATA COMM Interrupts or operator intervention. If an interrupt is not sensed, control is returned to the address stored at machine location 043 of the interrupted program.

TANK SWITCH.

This one position switch is located at machine location 035. The switch is automatically set when a program is called and either I/O or sufficient memory is not available.

When these conditions do not exist, the MCP will automatically call the program from the program tank into memory for execution.

NOTE

Only one program can be
stored in the program tank.

TODAYS AND REPORT DATE STORAGE.

Todays Date is stored in the Executive Routine at machine location 036. The CHDF Function stores Todays Date at this location. The second word in the user program is used to derive different types of dates from this address. The Report Date is used in the same manner, and stored at machine location 276.

FUNCTION/PROGRAM IDENT HOLD AREA.

This five position hold area within the Executive is used to store the identification of the program or function being executed.

USER PROGRAM LIBRARY DISK ADDRESS.

The seven position base disk address location where the User Program Library is stored on disk is machine location 11# within the Executive. When a user program is called for execution the Executive Routine uses the base address to read the program directory. The program directory is searched to find the appropriate program, and the disk address of where the program is stored. The program is then loaded according to program type and the number of blocks.

STANDARD END-OF-FILE CARD (OR RECORD).

The following is the card or record format for a standard End-of-File Record (b = a blank character).

```
≥ ≥ ≥ b E O F
1 - - - - - 7
```

MANUAL CALL FUNCTIONS.

The following pages describe each manually initiated function. The functions are initiated via card reader 1 or through the SPO. The functions are manually initiated by the user during the operation of MCP II. The functions are listed in table 2-4, and are used whenever the situation requires their use.

Table 2-4

Manually Initiated Function Calls

Call	Description
≥ ≥ ≥DUPL	Duplicate MCP Systems Tape Function
≥ ≥ ≥CHDF	Change (or Load) Date Function
≥ ≥ ≥LDOP	Load MCP II Function (M.T. Version)
≥ ≥ ≥STDJ	Switch to MCP II Disk Executive Function (M.T. Version)
≥ ≥ ≥STTF	Switch to MCP II M.T. Executive Function (Disk Version)
≥ ≥ ≥LALG	Load Autoload Program and Go Function
≥ ≥ ≥DISC	Discontinue Multiprogramming Program Function
≥ ≥ ≥SDCM	Set Data Communications Interrogate Function
≥ ≥ ≥MXTB	Multiprogramming Mix Listing Function
≥ ≥ ≥IOTB	Change Systems I/O Table Function
≥ ≥ ≥AXCE	Accept input from Supervisory Printer
≥ ≥ ≥PADD	Program Call-Outs from User Library

DUPLICATE SYSTEMS TAPE FUNCTION (>>> DUPL).
The DUPL Function is designed to create a copy of the MCP II systems tape, and to verify the copy on a word-for-word basis.

The DUPL format is as follows:

```
>>> DUPL
1 - - ---7
```

Operating instructions are displayed at the SPO, and EOJ-DUPL is automatically displayed when the tape has been successfully duplicated and verified.

It is advisable to always have a back up copy of the MCP II systems tape.

At EOJ-DUPL the MCP will return to the interrogate loop waiting for the next call. If an in-process program was interrupted to accomplish the DUPL Function it will cause a roll-in of the interrupted program, and return control to the program.

CHANGE (OR LOAD) DATE FUNCTION (≥ ≥ ≥ CHDF).

The CHDF Function provides a Todays Date and a Report Date for use by any in-process program. By optionally coding the second word of the program with the data parameters described in the NDAF or ADAF Functions: CHDF makes available the two date constants.

The following format is used to display the dates stored.

```
≥ ≥ ≥ CHDF
1 - - ---7
```

The format to change Todays Date and/or Report Date is as follows:

```
≥ ≥ ≥ CHDFTMMDDYYRMMDDYY
≥ ≥ ≥ CHDFTMMDDYY
≥ ≥ ≥ CHDFRMMDDYY
```

Each of the above calls must start at position 1, and the appropriate messages that reflect the dates presently stored in the Executive are displayed on the SPO.

At EOJ-CHDF the MCP will return to the interrogate loop waiting for the next function. If an in-process program was interrupted to accomplish the CHDF Function it will cause a roll-in of the interrupted program and then return control to the program.

LOAD MCP II FUNCTION (≥ ≥ ≥ LDOP).

The LDOP Function reads the MCP II disk version from the master systems tape. Starting at the address specified in the function call, it loads the disk operation system onto the disk. The Disk Executive Loader Control Card used by the DSCL Function will be created and punched by the LDOP Function. A Function Table is created and placed into the Call Check Function (≥ ≥ ≥ FCDF) for the validation of function names during the manually initiated calls.

The Function Table is comprised of entries that reflect a 4-character call name, and a 7-character disk file starting address for each function. A copy of the Function Table is maintained in FCDF as protection against an erroneous entry during a manually initiated call.

The LDOP Function format is as follows:

```
≥ ≥ ≥ L D O P x x x y y y y y y y
1 - - - - - 7 8 -10 11- - - - -17
```

where:

a. xxx - number of functions to be loaded.
b. yyyyyyy - starting disk file address required to load MCP II.

Messages stating the number of functions that were loaded, and their beginning and ending disk addresses, are displayed on the SPO.

The EOJ-LDOP will return control to the tape Executive Controller interrogation for the next function to be initiated.

SWITCH TO MCP II DISK EXECUTIVE FUNCTION (>>> STDJ)
MAGNETIC TAPE ONLY.

The purpose of the STDJ Function is to transfer control from the MCP magnetic tape version to the MCP disk version. The disk version must be resident on the disk file before control can be passed. This is accomplished with the use of the LDOP Function.

The Disk Executive Controller is loaded into memory from the disk file, and control is transferred to Executive.

The STDJ Function format is as follows:

```
>>> S T D J Y Y Y Y Y Y Y
1 - - -  - - 7 8 - - - - - 14
```

where:

a. yyyyyyy - starting MCP II disk address.

When control has been successfully passed to the disk executive version of MCP II, the appropriate DISK EXEC LOADED message is displayed on the SPO.

SWITCH TO MCP II TAPE EXECUTIVE FUNCTION (> > > STTF) DISK ONLY. The purpose of the STTF Function is to transfer control from the MCP disk version to the MCP magnetic tape version. The magnetic tape Executive is loaded into memory from the systems tape, and control is transferred to the magnetic tape Executive.

The STTF Function format is as follows:

```
> > > STTF
1 - - ---7
```

When control is successfully passed to the tape Executive version of MCP II, the appropriate TAPE EXEC LOADED message is displayed on the SPO.

LOAD AUTOLOAD AND GO FUNCTION (>>> LALG).

The purpose of the LALG Function is to permit an object program load and go operation under MCP II control. Programs containing overlays and multiprogramming may not be called by this function.

The LALG Function format is as follows:

```
>>> L A L G a a a i u c
1 - - -  -  -  7 8  - 10 11 12 13
```

where:

a. aaa - address where the program will begin operation.

b. i - input devices where the auto-loads reside
   C - input from card reader 1.
   B - input from magnetic tape.

c. u - MTU designate where auto-loads reside.

d. c - LALG Function call method code S to specify that the function will be called by the SPO. Any other code will result in 42 characters being transferred (starting at this position) to the function call area. This feature allows parameters to be transferred to memory for use by the program that was loaded during operations.

A SPO message will indicate any errors.

DISCONTINUE MULTIPROGRAMMING PROGRAM FUNCTION (>>> DISC).
The purpose of the DISC Function is to remove a specified program from the Program and Return Linkage Tables. The mix is reduced by one, and an appropriate message is displayed on the SPO. The memory space used by the discontinued program is returned to the MCP, and the Multiprogramming Controller recalled when additional programs are in the mix; otherwise, control will pass to the End-of-Job Function. All programs in the mix or waiting in the tank may be discontinued by the system operator.

The DISC Function format is as follows:

```
>>> DISC p   p   p   p   p
1 - - ---7 8   -   -   -   12
```

where:

a. ppppp - program ID to be discontinued. If *ALL* is entered, both the mix and tank are discontinued.

A SPO message will indicate that the program(s) have been discontinued.

Figure 2-3 is an example of a multiprogramming environment >>> DISC Function.

>>> DISC
continued

ONE PROGRAM

```
≥≥≥PADDOP3T1
BEGIN RUN - OP3T1
O #01 C1
O D21 T1
#01 NR


≥≥≥DISCOP3T1
DIS - OP3T1
END OF JOB - M/PC
```

THREE PROGRAMS

```
≥≥≥PADDOP3T1
BEGIN RUN - OP3T1
O #01 C1
O D21 T1
#01 NR
≥≥≥PADDOP3T2
BEGIN RUN - OP3T2
#01 NR
O D12 T
O A01 P
D12 NR
≥≥≥PADDOP3T3
BEGIN RUN - OP3T3
D12 NR
O @00 CP
O D13 TP
D13 NR


≥≥≥DISC*ALL*
DIS - OP3T1
DIS - OP3T2
DIS - OP3T3
END OF JOB - M/PC
```

Figure 2-3. DISC SPO Messages

SET DATA COMMUNICATIONS INTERROGATE FUNCTION (>>> SDCM).
The purpose of the SDCM Function is to notify the Executive Controller that the interrogation of Data Communications Terminals is, or is not required. The setting is recorded in the Executive on the disk and in memory. The set will remain until it is changed by a subsequent SDCM call.

The SDCM Function format is as follows:

```
>>> SDCM
1 - - ---7
```

A SPO message will indicate whether the interrogation of the Data Communication Terminals has been turned on or off.

MULTIPROGRAMMING MIX LISTING FUNCTION (≥ ≥ ≥ MXTB).

The purpose of the MXTB Function is to provide a SPO listing of the programs currently being executed, and the block size and memory addresses for each program. This function is callable, and does not require a SAVE memory prior to the execution of MXTB.

The MXTB Function format is as follows:

```
≥ ≥ ≥ M X T B
1 - - - - - 7
```

The SPO message indicates the program ID, block size, and the memory area assigned to each programs in the mix and available memory. When processing is not in the multiprogramming mode, a message of NOT MP is displayed (figure 2-4).

```
≥≥≥MXTB
PG #1 OP3T1 06 BLKS 800 TO S00
PG #2 OP3T2 07 BLKS S00 TO xx0
PG #3 OP3T3 08 BLKS xx0 TO B+0
AVAIL CORE  17 BLKS B+0 TO 000



≥≥≥MXTB
NOT MP
END OF JOB - MXTB
```

Figure 2-4. MXTB SPO Messages

CHANGE SYSTEMS I/O TABLE FUNCTION (≥ ≥ ≥ IOTB).

The purpose of the IOTB Function is to provide the capability of changing the systems configuration maintained within the Systems I/O Table. The message displayed on the SPO reflects the old and new parameters. This function either recalls the Executive Controller, or exits to the End-of-Job Function; depending on whether a job is in the mix.

The IOTB Function format is as follows:

```
≥ ≥ ≥ I O T B a b c  d  e  f  g g g g g g
1 - - - - - 7 8 9 10 11 12 13 14 - - - -19
```

where:

a. a - card reader 1 if column 8 contains a 1.
paper tape reader 1 if column 8 contains an A.
blank signifies neither is available.

b. b - card reader 2 if column 9 contains a 1.
paper tape reader 2 if column 9 contains a B.
blank signifies neither is available.

c. c - card punch 1 if column 10 contains a 0 (zero).
paper punch 1 if column 10 contains a +.
blank signifies neither is available.

d. d - blank . This position is reserved.

e. e - line printer 1 if column 11 contains a 1.
blank signifies that line printer 1 is not available.

f. f - line printer 2 if column 12 contains a 2.
blank signifies that line printer 2 is not available.

g. *gggggg* - magnetic tape units 1 through 6. Codes the available physical MTUs in their order. If a MTU is not available the corresponding columns must be blank.

For example, if all the MTUs are available the code is 123456, and if MTUs 2 and 4 were not available, the code would be 1b3b56.

NOTE

All 19 positions must be entered when the IOTB Table is revised. If ≥ ≥ ≥ IOTB is entered without units, the function will print the current setting of the systems I/O Table on the SPO (see figure 2-5).

```
≥≥≥IOTB
IOT:          456
D12 NR

≥≥≥IOTB
IOT: 1 0 1 123456
END OF JOB - IOTB
```

Figure 2-5. IOTB SPO Messages

ACCEPT SPO MESSAGE FUNCTION (>>> AXCE).

The AXCE Function is used in conjunction with the ACCEPT Macro. When a message is displayed on the SPO that requires a reply from the system operator, the AXCE Function passes the message to the correct program.

The AXCE Call format is as follows:

```
>>> A X C E i d e n t m e s s a g e
1 - - - - - - 7 8 - - -12 13  - - - -42
```

where:

a. ident - program identification.
b. message - information the operator must pass to the program.

The following is an example of how to use the AXCE Function. In order to inform the operator that information is needed, it is suggested that a display message indicate the program requiring the message, and the type of program expected. This information may be indicated either in the operating instructions or as part of the display message.

In figure 2-6 the system operator is awaiting a YES reply so that a ZIP to another program can be accomplished.

≥≥≥PADDAXTST	entered by operator.
BEGIN RUN - AXTST	printed by operating system.
DISP AXTST,	printed by executing program.
TST DISPLAY	printed by executing program.
AX: AXTST	printed by executing program.
≥≥≥AXCEAXTSTYES	entered by operator.
DISP AXTST,	printed by AXCE Function.
YES	printed by executing program.
EOP - AXTST	printed by end-of-program function.
ZIPPADDOP3T1	printed by operating system.
BEGIN RUN - OP3T1	printed by operating system.
O #01 C1	printed by file open function.
O D21 T1	printed by file open function..
#01 NR	printed by multiprogramming controller.

Figure 2-6. AXCE SPO Messages

PROGRAM CALL-OUT FROM USER LIBRARY (>>> PADD).
The purpose of this function is to provide a capability for retrieving user programs from the MCP II User Program Library via the supervisory printer or card reader.

The PADD Call format is as follows:

```
>>> P A D D i d e n t  Program Parameters
1 - - - - - 7 8 - - - 12 13 - - - - - - - - 54
```

where:

a. ident - the identification of the program being called from the User Program Library.

b. program parameters - any parameter necessary to execute the program, up to a maximum of 27 characters.

SECTION 3
UTILITY FUNCTIONS

GENERAL.

The MCP II utility functions provide for the execution of operations that are standard at most installations. The utility functions can be called by either the operator or a user program. It must be noted that the utility functions do not have the necessary float codes for multiprogramming capabilities: The Save/Restore feature of MCP II may be used by the utility functions.

The following pages within this section provide a detailed description of the utility functions and the operational procedures required to successfully utilize their capabilities.

The functions described in this section are listed in table 3-1.

Table 3-1
Utility Functions

Call	Description
≥ ≥ ≥PRME	Print Memory
≥ ≥ ≥DTSJ	Disk to Tape - Single segment
≥ ≥ ≥TDSJ	Tape to Disk - Single segment
≥ ≥ ≥DTTR	Disk to Tape - Multiple segments
≥ ≥ ≥TTDR	Tape to Disk - Multiple segments
≥ ≥ ≥DDDL	Disk to Tape/Tape to Disk
≥ ≥ ≥DCCN	Disk to Card - Control numbers
≥ ≥ ≥CDCN	Card to Disk - Control numbers
≥ ≥ ≥PRDK	Print Disk
≥ ≥ ≥CLDK	Clear Disk

Table 3-1 (cont)
Utility Functions

Call	Description
≥ ≥ ≥DCWR	Disk Word Change
≥ ≥ ≥DTDK	Disk to Disk
≥ ≥ ≥BTTR	Binary Tape to Tape
≥ ≥ ≥PRTB	Printer Back-up Function
≥ ≥ ≥TUTL	Tape to Print Utility
≥ ≥ ≥CARD	Card Utility Function
≥ ≥ ≥TAPE	TAPE Utility Function.

PRINT MEMORY (>>> PRME).
This function prints the contents of memory and the address of each word. If the beginning and ending parameters are not specified, all of memory is printed.

The Print Memory Function is one block in length, and permits the utility routine to be executed from 400 thru 79@. This allows the execution to be accomplished during an interrupt and without the execution of the Save Memory requirement.

The PRME Function format is as follows:

```
> > > P R M E b b  e  e
1 - - - - - 7 8 9 10 11
```

where:

a. bb - the start print memory address (tens and hundreths positions only).

b. ee - the end print memory address (tens and hundreths positions only).

If the Begin and End addresses are not entered with the function call, memory will be printed from 000 through the end of core.

Figure 3-1 is an example of a >>> PRME Function, and illustrates a print memory starting from machine location 000 through machine location 880.

ADRS	INST/DATA	ADRS	INST/DATA	ADRS	INST/DATA	ADRS	INST/DATA	ADRS	INST/DATA
000	61 050007500	010	61 23#OVRINP	020	7x102405F179	030	61 09E090971	040	61 FND440440
050	12717#005144	060	703066400156	070	702156 38206	080	7x2067150216	090	+10250108240
100	02 260356	110	61 04PRME 00	120	090000009012	130	01 136DSK RD	140	ER 00082404
150	K22144800130	160	K8 160130160	170	J 400156 36	180	121216180216	190	117152144144
200	56221640#150	210	7x220640 216	220		230	7x1238055179	240	7x1248054179
250	7x1258053179	260	7x1268051179	270	090971	280		290	1 0 1 123456
300		310	60 32 32106	320	70232602 270	330	321270152273	340	127274005144
350	61 060>>>PRM	360	E0089+ 12345	370	6+1+	380		390	
400	+>OVR>PRMF01	410	B01	420	504361596440	430	541361496450	440	704666 361
450	702361 503	460	A10670460	470	7014761 461	480	790670 680	490	703503+ 749
500	71050050 685	510	J 010503	520	502503363600	530	J 018499	540	I 018509
550	521506504570	560	561507504490	570	703576681499	580	703586685509	590	61 460
600	A01670	610	+ 77007	620	541318629640	630	7x2616330270	640	521656179000
650	701656 179	660	61 0400000	670		680	650 701656	690	179 660
700		710		720		730	730 730 7	740	30 7 740
750		760		770	MEM PRINTED+	780		790	
800		810		820		830		840	
850		860		870		880			

Figure 3-1. Memory Print Example

DISK TO TAPE SINGLE SEGMENT (≥ ≥ ≥ DTSJ).
This function writes the contents of a specified area of disk onto magnetic tape. The contents are formatted one tape block for each disk segment.

This function resides in memory beginning at machine location 800, fills three blocks of core, and utilizes the end of memory for binary tape writes.

DTSJ contains programmed interrupts to the Executive Controller and may be interrupted by another program. If a running program is interrupted the Save/Restore Memory Function of the MCP is automatically called.

The DTSJ Function format is as follows:

```
≥ ≥ ≥ D T S J b b b b b b b e e e e e e e
1 - - - - - 7 8 - - - - -14 15 - - - - -21
```

where:

a. bbbbbbb - the beginning address of the disk file segment to be dumped onto tape.

b. eeeeeee - ending address of the disk file segment to be dumped onto tape.

Figure 3-2 is an example of a DTSJ Function. This example illustrates an area cleared with Cs, and dumped to tape with ≥ ≥ ≥ DTSJ printed to show single segment ≥ ≥ ≥ DTSJ00000010000050←.

```
001   00000010000050                                                                                              BINARY
101
201
301
401                                                                                   ********************
                                                                                               RECORD NUMBER    01

001   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC      BINARY
101   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
201   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
301
401                                                                                   ********************
                                                                                               RECORD NUMBER    02

001   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC      BINARY
101   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
201   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
301
401                                                                                   ********************
                                                                                               RECORD NUMBER    03

001   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC      BINARY
101   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
201   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
301
401                                                                                   ********************
                                                                                               RECORD NUMBER    04

001   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC      BINARY
101   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
201   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
301
401                                                                                   ********************
                                                                                               RECORD NUMBER    05

001   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC      BINARY
101   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
201   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
301
401                                                                                   ********************
                                                                                               RECORD NUMBER    06

001   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC      BINARY
101   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
201   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
301
401                                                                                   ********************
                                                                                               RECORD NUMBER    07
```

Figure 3-2. DTSJ Function Listing Example

TAPE TO DISK SINGLE SEGMENT (≥ ≥ ≥ TDSJ).

The Tape to Disk Function is used in conjunction with the Disk to Tape Single Segment Function. The TDSJ Function reads the magnetic tape created by the DTSJ Function and writes the contents into a specified area of the disk file. If the disk address specified by the function call differs from the disk address contained in the tape label, a message is printed on the SPO; the decision whether or not to execute the function is made by the user.

The function is based at machine location 800 filling five blocks of core, and contains program interrupts to the Executive Controller. The Save/Restore Memory Function is automatically called when it is required.

The TDSJ Function format is as follows:

```
≥ ≥ ≥ T D S J b b b b b b b e e e e e e e
1 - - - - - 7 8 - - - - -14 15 - - - - - 21
```

where:

a. bbbbbbb - disk address where the first tape record is placed.
b. eeeeeee - disk address where the last tape record is placed.

DISK TO TAPE UTILITY - MULTIPLE SEGMENTS (≥ ≥ ≥ DTTR).
This function writes the contents of an area within the disk file (as specified within the parameters of its call) onto magnetic tape in 10 segment blocks (binary). The last written block may contain from 1 to 10 blocks (depends on the size of the disk area being dumped). DTTR may be interrupted via the SPO so that other functions, utilities, calls, or user programs may be processed (figure 3-3).

The implementation of this routine requires a working knowledge of the operating requirements.

The operating functions are as follows:

a. Scratch tape is mounted on MTU-1.
b. The DTTR Call is either entered via the SPO or card reader 1.

The DTTR format is as follows:

≥ ≥ ≥ D T T R
1 - - - - - 7

The following messages can be displayed on the SPO:

a. ENTER SEGMENT ADDRESS
This is the begin-run message requesting that the operator enter 14 numeric digits. This entry represents the beginning disk address (first seven digits), and the ending disk address (last seven digits) of the area to be dumped.

b. MOUNT WORK TP U1
This message signifies that the physical end-of-reel has been reached, and that another scratch tape must be mounted on MTU-1. CONTINUE on the Central Processor is then pressed by the operator.

c. WR ER UI TM
This message signifies that an attempt to write the final tape mark has failed. The operator may press CONTINUE on

the Central Processor (the tape will not be backspaced or erased), or CLEAR and then CONTINUE to discontinue the function.

```
 001   000000010000020
 101
 201
 301
 401
 501
 601
 701
 801
 901
1001
1101
1201
1301
1401
1501
1601
1701
1801
1901                        CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2001   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2101   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2201
2301
2401    ***************************************************************************************************
                                                                                          RECORD NUMBER

 001   0CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 101   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 201   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 301   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 401   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 501   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 601   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 701   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 801   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 901   CCCC[illegible]CCCCCCCCCCCC[illegible]CCCCCCCCCCCCCCCCCC[illegible]CCCCCCCCCCC[illegible]CCCCCCCCC
```

Figure 3-3. Example of Area Cleared with C's Dumped to Tape and Multiple Segments with > > > DTTR and Printed to Show Multiple Segments. The Header Contains the Parameters of the Area Dumped, and the Control Character (zero) is located in the First Position of Each 2401 Character Block.

TAPE TO DISK - MULTIPLE SEGMENTS (≥ ≥ ≥ TTDR).
The TTDR Function (in conjunction with the Disk to Tape Multiple Segment Function) will write the contents of a tape onto a specified area of the disk file in ten segment increments. This function cannot be interrupted. If this function interrupts an executing program, the Save Restore Memory Function is automatically called.

The TDDR Function executes from memory location 800 and requires three blocks of core.

The TTDR Function format is as follows:

```
≥ ≥ ≥ TTDR
1 - - ---7
```

An information halt will verify the parameters contained in the tape label.

DISK DUMP DISK LOAD (≥ ≥ ≥ DDDL).

The DDDL Function may be used in place of the DTTR and TTDR Functions. The logic differs in that it dumps disk by module number only. This will assure the complete dumping of a particular module. After the disk is dumped the function will read the tape just written and check for missing, duplicate, and the correct number of records. When reloading the disk from tape, it can be used to load an entire module or any portion thereof. It is possible to load one segment or multiple segments anywhere within the limits of that module. By entering additional parameters, several areas or files can be loaded during the same program "Call-In."

This function executes from memory location 800, filling seven blocks of core. The program cannot be interrupted. The Save/Restore Memory Function is called automatically if the function interrupts an executing program.

The DDDL Function format is as follows:

```
≥ ≥ ≥ DDDL
1 - - ---7
```

DISK TO CARD WITH CONTROL NUMBERS (≥ ≥ ≥ DCCN).
This function punches the contents of a specified area of a disk file into cards. It will include the disk file address and card number in each card punched (figure 3-4).

The DCCN Function resides at machine location 800, and fills four blocks of core. The program contains interrupts to the Executive Controller.

The DCCN Function format is as follows:

```
≥ ≥ ≥ DCCN b b b b b b b  e e e e e e e
1 - - ---7 8 - - - - - 14 15- - - - - 21
```

where:

a. bbbbbbb - beginning disk address of segment(s) to be punched into cards.

b. eeeeeee - ending disk address of segment(s) to be punched into cards.

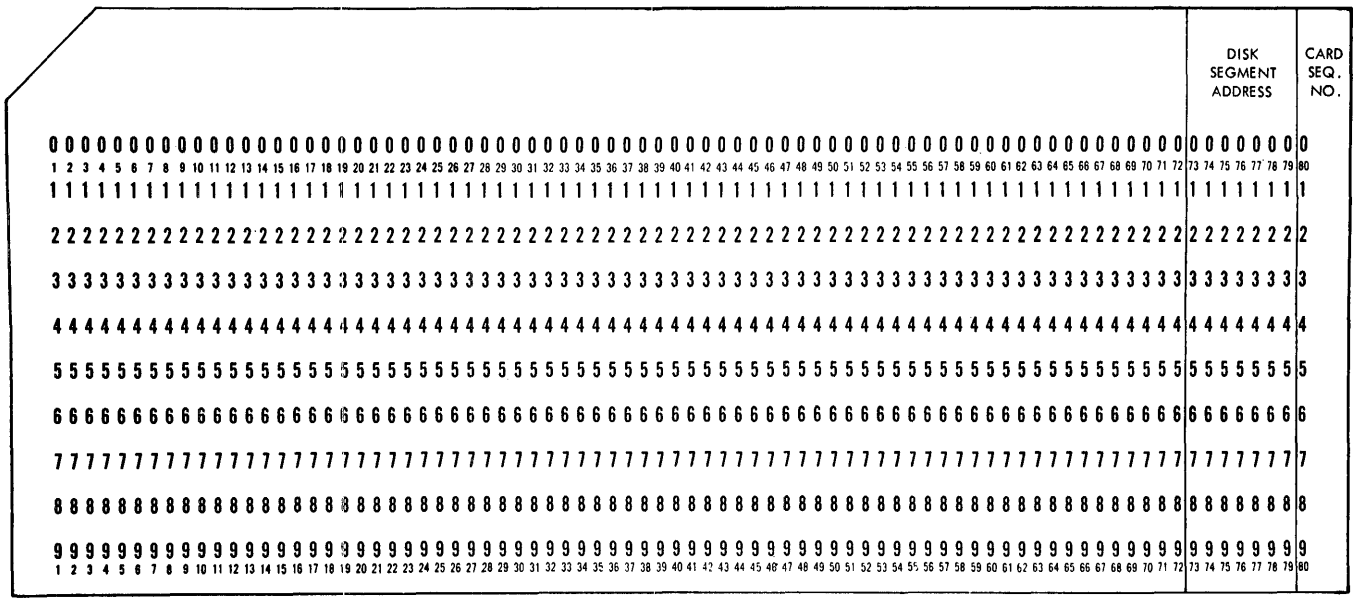


Figure 3-4. Disk to Card Record Format

NOTE

Each disk segment will initiate an output of seven cards. The size of this segment can be 96, 240, or 480 characters.

CARD TO DISK - WITH CONTROL NUMBERS (≥ ≥ ≥ CDCN).
This function is used in conjunction with the Disk to Card Function. The cards punched by the CDCN Function is the source input. The function programmatically checks the control numbers during input, and writes the contents of the input cards into the disk area specified by the parameters (reference figure 3-3).

The function provides the necessary program interrupts to the Executive Controller and initiates the Save/Restore Memory Function whenever necessary. The function is based at machine location 800 and requires four blocks of core.

The CDCN Function format is as follows:

```
≥ ≥ ≥ CDCN b b b b b b b  e e e e e e e
1 - - ---7 8 - - - - - - 14 15- - - - - 21
```

where:

a. bbbbbbb - beginning disk address where the contents of the punched cards are written.

b. eeeeeee - ending disk address where the contents of the punched cards are written.

PRINT DISK (> > > PRDK).

This utility function prints the contents of the disk file specified by the input parameters. PRDK is based at machine location 800 and utilizes five blocks of core. This function provides the necessary program interrupts to the Executive Controller and initiates the Save/Restore Function whenever necessary.

The PRDK Function format is as follows:

```
> > > PRDK b b b b b b b  e e e e e e e  s s s
1 - - - -7 8 - - - - - - 14 15- - - - - 21 22- 24
```

where:

a. bbbbbbb - address of the first disk segment to be printed.
b. eeeeeee - address of the last disk segment to be printed.
c. sss - segment size (096, 240, or 480).

Figure 3-5 is an example of a > > > PRDK Function.

```
DISK ADDRESS    0123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
  0000001       cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccc
  0000002       cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccc
  0000003       cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccc
  0000004       cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccc
  0000005       cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccc
  0000006       cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccc
  0000007       cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                cccccccccccccccccccccccccccccccccccccccccccc
```

Figure 3-5. Example of ≥ ≥ ≥ PRDK or ≥ ≥ ≥ CLDK Listing

CLEAR DISK (≥ ≥ ≥ CLDK).

The Clear Disk Function is used to fill an area of disk with a specific character. It may be useful to programmatically call this function within a program, fill the area with blanks or another character, then return to the first program. (An example is found in the Programming Technique Section.)

This function is based at machine location 800, and fills eight blocks of core. If this function is called for execution while another program is executing, the Save/Restore Memory Function is automatically called. CLDK contains programmed interrupts to the Executive Controller and may be interrupted by another program.

The CLDK Function format is as follows:

```
≥ ≥ ≥ C L D K b b b b b b b e e e e e e e s s s c
1 - - - - - 7 8 - - - - -14 15 - - - - 21 22-24 25
```

where:

a. bbbbbbb - disk address of the first segment to be cleared.
b. eeeeeee - disk address of the last segment to be cleared.
c. sss - disk segment size (096, 240, or 480).
d. c - clear character.

NOTE

If a group mark is entered as the clear character in position 25, an incomplete BEGIN RUN message is printed on the SPO. A YES reply will fill the specified disk area with group marks.

Figure 3-5 is also an example of a CLDK Function.

DISK WORD CHANGE UTILITY (>>> DCWR).
This function permits the user to change any word of any file resident within the disk.

The implementation of this function requires a working knowledge of the operating requirements. The DCWR Call is entered via the card reader or SPO.

The DCWR format is as follows:

```
>>> D C W R
1 - - - - - 7
```

The following messages are displayed on the SPO:

a. ENTER SEGMENT ADDRESS
   This is the begin-run message. The operator must enter the 7-digit disk address of the word to be changed via the SPO.

b. ENTER WORD NUMBER
   This message requests the entry of two numeric digits (the number of words 12-character groups) between the beginning of the segment and the word to be changed.

   Example:

   Assuming 240-character segments:

   1) To change the first word enter 00.
   2) To change the second word enter 01.
   3) To change the last word enter 19.

c. OLD WORD IS ddddddddddddd IS THIS CORRECT
   This message verifies the contents of the specified location containing the expected information. If the data (ddddddddddddd) is correct the operator may

reply YES; if incorrect, a reply of NO will cause the input specification to be ignored. If the data contains a group-mark, the SPO message will terminate after the character prior to the printed group-mark.

d. ENTER NEW WORD
This message requests the operator to enter the required 12-character new word to replace the ddddddddddd parameter in item c.

e. MORE CHANGES
This message is self-explanatory. The operator may enter YES and the routine will continue: NO will cause the routine to proceed to End-of-Job.

DISK TO DISK (>>> DTDK).

The Disk to Disk Function will copy the contents of a specified area of the disk file to another specific area of the disk file.

This function is based at machine location 800 and fills eight blocks of core. If this function is called for execution while another program is executing, the Save/Restore Memory Function is automatically called. DTDK contains programmed interrupts to the Executive Controller and can be interrupted by another program.

The DTDK Function format is as follows:

```
> > > D T D K b b b b b b b e e e e e e e t t t t t t t s s s
1 - - - - - 7 8 - - - - -14 15- - - - -21 22- - - - -28 29- 31
```

where:

a. bbbbbbb - disk address of the first segment to be transferred.

b. eeeeeee - disk address of the last segment to be transferred.

c. ttttttt - beginning disk address of the first segment where the contents will be transferred.

d. sss - disk segment size (096, 240, or 480).

BINARY TAPE TO TAPE (≥ ≥ ≥ BTTR).
This function will copy a binary tape with one or more input reels having the same record length. The parameters that specify the number of input reels and record lengths up to a maximum of 4800 binary characters are passed to this function at begin run time.

BTTR can also be used to position the COBOL Collector Tape so that additional programs can be added to the tape. This can be accomplished by specifying two input reels and 0080 character binary input. When the second input reel is requested, the tape is positioned to add additional programs to the Collector Tape, CLEAR is pressed and then CONTINUE on the Central Processor to discontinue the function.

The BTTR Function is executed from machine location 800 and fills three blocks of core. It also utilizes the upper end of memory for binary tape writes (depending on the number of binary characters to be written).

This function cannot be interrupted to process another function or program. If called by a program, the Save/Restore Memory Function will be called automatically.

The BTTR Function format is as follows:

```
≥ ≥ ≥ B T T R
1 - - - - - 7
```

PRINTER BACK-UP FUNCTION (>>> PRTB).

This function provides the capability of printing printer back-up tapes created by the system. It may also be used to print BCL tapes formatted with the first three characters of each unblocked tape record containing the OP, M, and N variant of the print instruction The remainder of the record is a line of print information that is 120 or 132 characters in length.

The Printer Back up Function is one block in length, thus permitting this utility to be accomplished during an interrupt, and without the execution of the Save/Restore Memory Function requirement.

The format for the PRTB Function is as follows:

```
>>> P R T B
1 - - - - - 7
```

TAPE TO PRINT UTILITY (≥ ≥ ≥ TUTL).

The Tape to Print Function will print any labeled or unlabeled tape written on the system. It has the capability of printing BCL and/or binary records of variable length and inter-mixed records (figure 3-6).

The TUTL Function is executed from Machine Location 800 and utilizes 9.6K core.

The TUTL Function is as follows:

```
≥ ≥ ≥ T U T L
1 - - - - - 7
```

```
09-09          1         2         3         4         5         6         7         8         9                    PG 01
     1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789

 001 0000000  01 0000006  02 0000080  03 0000088  05              0000150  09                                      B.C.L.
 101         0000204  10 0000228  12              0000234  90              0000332  15 0000368  21 0000424
 201  17 0000448  18 E000470  27 0000494  20 0000514  22 0000382  16 0000536  88 G700834  66 0000928  30
 301                          0000474  19              00008B4  24 0001008  85
 401                                                                          0001054  45 00+11720
 501 142 80+11740143 80+11760146 5H01164  47 5H01152  44 Q 01178  97 5H01168  41
 601                                          +118601
 701                                                                                   00+08740
 801 529                           T300790  86 3R+05781471 3R+06060872 3R+06221273 3R+06460874 3R+06420875
 901 3R+06780876 3R+06941077 3R+07141478 3R+07421479 3R+07701069                        61 PR0>>>
1001                      V901052  40 G700792  23              3R+05740270              0000292  14
1101                                                                    0001180  54
1201 *************************************************************************************************************
                                                                                                  RECORD NUMBER     01

 001 561LJ4JLAKJ0J  00?LJ3>>>>702xR7   KL5701LJ5   KL761 KL0000000000000A2000K4000T6000A0000F*000PK000YM00      B.C.L.
 101 09F001+H001K+001T9001AU001EW001QY001Y 00000000000000000      1         2         3
 201    4            STGWCOVR      TYPSFQ                                  0123456789#@0123456789#@0
 301 1234567A9#@0123456789#@0123456789#@LOC  ##                                                    MISSI
 401 NG CONTROL CARD                    INVALID DISK SEGMENT SIZE CODE          INVALID COMPILER BASE ADD
 501 RESS           MISSING OVERLAY                        MISSING END OVR CARD OR OVR NUMBER      ILLEG
 601 AL MAINTENANCE FUNCTION CODE       IMPROPER CARD SEQUENCE NUMBER           IMPROPER MEM ADDR OR OVER
 701 LAY SEQUENCE        #01PROPRO720543Z20Kx90P0501Z27 Q6RL0501Z27 Q7RL0501Z27 Q8RL0734NN3   2W0705Q07
 801  2L161 1M0 62PR0734NJ@   2W07050R7   2L161 1M0 61PR0734NQ7   2W0705RK7   2L161 1M0 63PR0701Z27   /62
 901 701Z27   1P2701Z27   SP0503Z2ARP6Rx060 Rx0000RQ0703Z28   S46703Z28   F76703Z28   1J7703Z28   /165017
1001 33 K@3M3501Z33 L@P81501Z33 M@+10734NQ7   2W0705 P7   2L161 1M0 64PR0      215
1101
1201 *************************************************************************************************************
                                                                                                  RECORD NUMBER     02

 001 561LJ4JLAKJ0J  00?LJ3>>>>702xR7   KL5701LJ5   KL761 KL0000000000000A2000K4000T6000A0000F*000PK000YM00      B.C.L.
 101 09F001+H001K+001T9001AU001EW001QY001Y 00000000000000000      1         2         3
 201    4            STGWCOVR      TYPSFQ                                  0123456789#@0123456789#@0
 301 1234567A9#@0123456789#@0123456789#@LOC  ##                                                    MISSI
 401 NG CONTROL CARD                    INVALID DISK SEGMENT SIZE CODE          INVALID COMPILER BASE ADD
 501 RESS           MISSING OVERLAY                        MISSING END OVR CARD OR OVR NUMBER      ILLEG
 601 AL MAINTENANCE FUNCTION CODE       IMPROPER CARD SEQUENCE NUMBER           IMPROPER MEM ADDR OR OVER
 701 LAY SEQUENCE        #01PROPRO720543Z20Kx90P0501Z27 Q6RL0501Z27 Q7RL0501Z27 Q8RL0734NN3   2W0705Q07
 801  2L161 1M0 62PR0734NJ@   2W07050R7   2L161 1M0 61PR0734NQ7   2W0705RK7   2L161 1M0 63PR0701Z27   /62
 901 701Z27   1P2701Z27   SP0503Z2ARP6Rx060 Rx0000RQ0703Z28   S46703Z28   F76703Z28   1J7703Z28   /165017
1001 33 K@3M3501Z33 L@P81501Z33 M@+10734NQ7   2W0705 P7   2L161 1M0 64PR0      215
1101
1201 *************************************************************************************************************
                                                                                                  RECORD NUMBER     03

 001                                                    K02U05000U20K8 S50U10U3056137357259OP01S      BINARY
 101 8N    X2707T05    73+[[  1K01###B20   T10   A01U40T20   701T31   T75702T46T50U13K22U05000U20K8 T50U1
 201 0T60P01T71   005521372T82000999U00      +[[ 4  97U059R0FE0      923U10      929U10          [1   SYNT
 301 AX ERROR(S) WERE FOUND, COMPILATION ABORTED [X8@##                                  740Q00   V00
 401                                      50 50   U50115 W2/V7/V761   W2      **********************
                                                                                                  RECORD NUMBER     04
```

Figure 3-6. A Listing with both BCL and Binary Variable Length Records

CARD UTILITY FUNCTION (>>> CARD).

The Card Function provides the capability for reproducing a card deck, creating a BCL card record file on tape, or listing the card input on the line printer. A tape mark-group mark (> ←) will cause a short tape record. The Card Function is one block in length and executes from 400 thru 79@. It can interrupt a running program without calling the Save/Restore Memory Function.

The CARD Function format is as follows:

```
>>> C A R D x
1 - - - - - 7 8
```

where:

x - T card to tape.
    P card to print.
    C card to punch.

TAPE UTILITY FUNCTION (≥ ≥ ≥ TAPE).

The Tape Function provides the capability of either punching a BCL card image tape, or making a duplicate copy of an 80 character BCL tape.

The tape is formatted into 80 character BCL code. A tape mark - group mark (≥ ←) will cause a short tape record.

The Tape Function is executed from memory location 400 thru 79@, thus permitting execution without requiring the Save/Restore Memory Function.

The Tape Function format is as follows:

```
≥ ≥ ≥ T A P E x
1 - - - - - 7 8
```

where:

a. x - T tape to tape.
P tape to punch.

SECTION 4
USER PROGRAM LIBRARY FUNCTIONS

GENERAL.

MCP II contains all the functions necessary to create and maintain a User Program Library. The capability of including user programs as call features of the system is also available. The programs added to the system can either be in a multiprogramming and/or a non-multiprogramming mode.

The User Program Library capability improves the systems operation, and replaces the tedious task of manually loading programs with a Call from the User System File.

MCP II contains the functions necessary to initiate and maintain this library. The user can manipulate tape and disk files, or tape and disk libraries by merging, deleting, or adding information.

The pages in this section provide a description of the User Program Library Functions that are listed in table 4-1.

Table 4-1
Program Library Functions

Call	Description
≥ ≥ ≥PADR	Create Program Add Tape Call
	Program Add Record Format
≥ ≥ ≥PLTM	Program Library Tape Merge Call
≥ ≥ ≥TWCR	Magnetic Tape Word Corrector Call
≥ ≥ ≥DPTL	Delete Programs From Library Tape Call
≥ ≥ ≥LLTP	List Library Tape Program Call
≥ ≥ ≥LTON	List Overlay Names Call

Table 4-1 (cont)

Program Library Functions

Call	Description
≥ ≥ ≥DELF	Delete Functions From Users System File Call
≥ ≥ ≥ADDR	Add Functions to MCP II Disk
≥ ≥ ≥LPAT	Load Program Add Tape to Disk Library Call
≥ ≥ ≥CPAT	COBOL Collector Tape to Disk
≥ ≥ ≥DWCR	Disk File Word Corrector Call
≥ ≥ ≥LDPL	List Disk Library Program Call
≥ ≥ ≥DPDL	Duplicate or Delete from User Disk Library Call
≥ ≥ ≥LDON	List Disk Overlay Names Call
≥ ≥ ≥COBL	COBOL Source Program Maintenance
≥ ≥ ≥STMT	Symbol Tape Maintenance
≥ ≥ ≥STUR	Symbolic Tape Update and Resequence Call
≥ ≥ ≥SSTO	Output from Symbolic Program Tape Call

CREATE PROGRAM ADD TAPE (>>> PADR).

The purpose of the PADR Function is to create a Program Add Tape of 480 character binary records, and a Program Add Listing.

This function uses the following steps to build a Program Add Tape.

a. Each Program Add Record is written on tape in the format of a Call or PADD Record. (Reference the PADD record format for Program Add Card specifications.)

b. The object program is written on tape in 480 character binary records.

c. A Program Add Listing is prepared when the Program Add Tape is created.

d. A count of the programs and overlays converted to tape is made. The count is printed on the SPO following the completion of the PADR Function. When updating a User Program Library it should be used as the count.

e. The following checks are made on the programs.

   1) The program identity in the Program Add Record must be the same as the object program record.

   2) Card sequence for detail records must be those supplied by the assembler.

   3) Total blocks written must equal the total blocks specified in the Program Add Record for each program.

f. An End-of-Job file card signals the completion of a program addition.

The PADR Function format is as follows:

```
>>> P A D R  i  s  s  d  d  d  d  d  d  d
1 - - - - - 7  8  9  10 11 -  -  -  -  -  17
```

where:

a. i - designates the type of peripheral input unit used to create the Program Add Tape.
   C - cards
   B - binary tape
   P - paper tape
   D - 96 character - segment disk file.
   E - 240 character - segment disk file.
   F - 480 character - segment disk file.

   NOTE
   Disk file input must be blocked six card image records.

b. ss - system specifications. This entry is only used to create MCP system tape. When a system tape is created, the code SY with binary tape input is used; otherwise, the field is blank.

c. ddddddd - beginning disk file address. If the input is stated as disk file input, this field will indicate the beginning disk address of the auto-loads on disk.

Figure 4-1 is an example of a PADR initiated Program Add Listing.

```
PROGRAM IDENTIFICATION:  OP3T1, AS OF 09-09-71                                                                                    PAGE 001
BLK NO.  WORD NO.        WORD NO. + 0  WORD NO. + 1  WORD NO. + 2  WORD NO. + 3  WORD NO. + 4  ADRS  CARD #  R/A FLAGS

  CALL      00           >>>PADDOP3T1  RN1     1          120012                                 000   0000

  0001      20           +BEGOP3T1A50                +*0C1    840  +D2T1    +00  7L6866450366  800   0002     Y YYS
  0001      25           +             61 SET                          400780  928D10840903  850   0003       *
  0001      30           5A7928921860  7X2869XX 916   >>> EOF                                 900   0004     J,
  0001      35                                                              +                 950   0005

  0002      20           7L6+26450366  +             61 SET                      +78780  +00   0006     S   U
  0002      25           400D10+00+20  7X7+71+00+78   >>> EOF                              +50   0007     JJ
  0002      30                                                              +             A00   0008
  0002      35           703A560PN019  702A6608 270  7K3A80330363  B10866C1      +       A50   0009     SSST

  0003      20                         703916OPN019  70282608 270  7K3B40330363  B70+26T1        B00   0010      SSST
  0003      25           +                          7X8B83840863  +  B92700C30  768928   +78  B50   0011      J,,
  0003      30           7XBC13+00+23  +  C20700790  61 B70        703C36CLS019  702C4609 270  C00   0012     JSSSS
  0003      35           7J0C60330363  C7066C1       703C76CLS019  702C8609 270  7J0D00330383  C50   0013     STSSS

  0004      20           D10+26T1      703016E0P019  702D2615 270  7K0800330370                 D00   0014     TSSS
  0004      24       16 INST/WORDS STARTING AT 040 FILLED WITH BLANKS TO COMPLETE THE BLOCK

  0005      20       20 INST/WORDS STARTING AT F00 FILLED WITH BLANKS TO COMPLETE THE BLOCK

  0006      20       20 INST/WORDS STARTING AT H00 FILLED WITH BLANKS TO COMPLETE THE BLOCK

  0007      20       20 INST/WORDS STARTING AT X00 FILLED WITH BLANKS TO COMPLETE THE BLOCK

  0008      20       20 INST/WORDS STARTING AT K00 FILLED WITH BLANKS TO COMPLETE THE BLOCK

  0009      20       20 INST/WORDS STARTING AT M00 FILLED WITH BLANKS TO COMPLETE THE BLOCK

  0010      20       20 INST/WORDS STARTING AT 000 FILLED WITH BLANKS TO COMPLETE THE BLOCK

  0011      20       20 INST/WORDS STARTING AT 000 FILLED WITH BLANKS TO COMPLETE THE BLOCK

  0012      20       20 INST/WORDS STARTING AT  00 FILLED WITH BLANKS TO COMPLETE THE BLOCK
```

Figure 4-1. Example of a PADR Program Add Listing.

>>> PADR continued

PROGRAM ADD RECORD FORMAT.

Each auto-load program that is used as input to the PADR Function must be preceded by a PADD record.

The PADD format is as follows:

```
≥ ≥ ≥ PADD  i  i  i  i  i  t  b  i  /  o  c  o  n  f  i  g  u  r  a  t  n  p  p  l  l  t  t  t  t
1 - - ---7  8  -  -  -  12 13 14 15 -  -  -  -  -  -  -  -  -  -  -  -  27 28 29 -  -  -  -  -  -  36
```

where:

a. ≥ ≥ ≥ PADD - user program call.

b. iiiii - program identification.

c. t - type of program being stored.
   - M - serial (non-programming).
   - T - on-line transaction routines.
   - R - multiprogramming assembler program.
   - Z - multiprogramming COBOL programs.
   - C - non-programming COBOL programs.
   - S - MCP II systems tape.

d. b - reserved - must be blank.

e. i/oconf - is the input and output configuration required for running the program. This field applies only to multiprogramming programs.

   i - input reader 1.
   - 1 - card reader 1.
   - A - paper tape reader.

   blank - unit is not required.

   / - input reader 2.
   - 1 - card reader 2.
   - A - paper tape reader.

   blank - unit is not required.

o - output punch.
    0 - card punch.
    + - paper tape punch.
blank - unit is not required.

c - blank (reserved for the system).

o - line printer 1.
    1 - printer (no tape back-up).
    T - printer, tape back-up is allowed if a printer is not available (assembler programs only).

n - line printer 2.
    2 - printer (no tape back-up).
    T - printer (tape back-up is allowed).

f - number of magnetic tape units.
    1-6 - number required.
    0 or blank - none required.

f. iguratnpp - reserved for the MCP.

g. ll - program blocks to load when the main portion of the program is loaded into core memory.

h. tttt - total program blocks (main core plus overlays) included with this program.

The following tests are made on input data during the Program Add Tape creation run.

a. The program identity in the Program Add Record must be identical in each card image record.

b. The sequence number field must be identical to that supplied by the assembler.

c. The "Program Blocks" field to load, and the "Total Program Blocks" supplied by the Call or Padd record must agree with the actual input counts.

A standard End-of-File card (or record) signifies the completion of the function.

>>> PLTM

PROGRAM LIBRARY TAPE MERGE (>>> PLTM) MAGNETIC TAPE VERSION. The purpose of the PLTM Function is to create a Tape Operating User System File. The MCP II System Tape (or previous User System File) is merged with the program ADD tape to create a current User System File.

To become part of the User System File, the programs on a Program Add Tape must be in alphabetic sequence. This function will test the sequence of the Program Add Tape, and merge the User System File with the Program Add Tape.

The PLTM Function is executed from machine location 800 and utilizes eight blocks of core using the end of memory for binary tape writes.

The PLTM Function is used in conjunction with the maintenance of the User Program Tape Library. The format is as follows:

```
>>> PLTM
1 - - ---7
```

TAPE WORD CORRECTOR (≥ ≥ ≥ TWCR) MAGNETIC TAPE VERSION.
The word corrector for the User Systems File provides the user with the capability to change any word (12 characters) of any user program in the library. This function is used in conjunction with the maintenance of the User Program Tape Library.

This function is executed from machine location 800 and utilizes six blocks of core.

The following steps describe how the TWCR Function operates.

a. A corrector card or message is read for each word to be changed.

b. The Call or Padd Record for the referenced program is located and used as the starting point for the block count.

c. The block number within the specified program is located and the change made.

d. All changes within the same block are made before the block is rewritten on a new library tape.

e. The corrector cards or messages must be in the following sequence.

   1) All corrector cards and messages must be grouped by block number and program.

   2) The program sequence must be the same as the library tape. The library tape (with the exception of the operating system itself) is arranged by the five-character identification collating sequence.

   3) The block sequence must be maintained within the program sequence. The sequence is:

Call record
Block 0001
.
.
Block nnnn

4) Word number sequence by block is not mandatory, but it is recommended to simplify off-line maintenance.

f. Complete tapes are duplicated in sequence. (Changes are made to affected blocks only.)

g. Changing the Call or Padd Record is accomplished by using CALL as the block number.

h. After each change is made, the record is printed on the line printer.

i. The word corrector listing is grouped by program. Each new program causes a skip to the heading line of a new page.

The Tape Word Corrector format is as follows:

≥ ≥ ≥ TWCRi
1 - - ---78

where:

a. i - input type. Specifies the type of input device used for the word corrector.
C - card reader.
S - supervisory printer.

The format for the detail card or SPO message is as follows:

```
                                                  old  word                                  new  word
≥ ≥ ≥ W C O R i i i  i  i  b  b  b  b  w  w  r  r  o  o  o  o  o  o  o  o  o  o  o  o  o  r  r  n  n  n  n  n  n  n  n  n  n  n  n  n  n
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
```

where:

a. iiiii - program identification in which a word change is made.

b. bbbb - block number in which the word is being changed.

c. ww - word number being changed.

d. rr - reserved for the system.

e. ooooooooooo - old word being deleted.

f. nnnnnnnnnnn - new word. Replaces the information in the old field.

Before the Word Corrector input is executed, the detail information must pass the following validity tests.

a. The program identification number must match a valid program number contained in the library.

b. The block number can not exceed the total number of blocks specified in the Call Record Program.

c. The old word field must compare equally with the actual information contained in the library. The check is a 12-character compare.

An End-of-File card or message is used to signify that all of the corrections have been processed.

DELETE PROGRAMS FROM TAPE LIBRARY (≥ ≥ ≥ DPTL) MAGNETIC TAPE VERSION. The DPTL Function allows user programs to be deleted from the User System File, and is used in conjunction with the maintenance of the User Tape Library Program.

The following steps describe how the DPTL Function is executed.

a. The Tape Operating System is duplicated from the old User System File to a new User System File.

b. Programs from the old User System File are duplicated until a Delete Program (specified by the Delete Card or message) is encountered.

c. The program to be deleted on the old User System File is bypassed.

d. The Call or Padd Record for each deleted program is printed.

e. Delete cards or messages must be arranged in library sequence. The sequence is a low-to-high collating order of the five-character identification field. Out-of-sequence cards or messages are not processed, and a message indicating this condition is printed.

f. An End-of-File record or message specifies that all of the deletions have been processed.

The DPTL Function format is as follows:

```
≥ ≥ ≥ D P T L i
1 - - - - - 7 8
```

where:

a. i - input device. This field designates the type of input used for the delete card or message.
   C - card reader
   S - supervisory printer

The format for the delete card or SPO message used as input for each deleted program is as follows:

```
≥ ≥ ≥ P  D  E  L  i  d  e  n  t
1 - - -  -  -  7  8  -  -  -  12
```

where:

ident - the identification of the program to be deleted from the User System File.

LIST LIBRARY TAPE PROGRAM (>>> LLTP) MAGNETIC TAPE VERSION. The purpose of this function is to obtain a listing of any program in the User System File Library. This function will format a listing that corresponds with the listing created by the PADR Function.

The LLTP Function format is as follows:

```
>>> L L T P i
1 - - - - - 7 8
```

where:

a. i - input device. This field designates the type of input used for the detail record.

C - card reader.

S - supervisory printer.

The format for the detail card or message is as follows:

```
>>> L T A P i d e n t
1 - - - - - 7 8 - - - 12
```

where:

ident - program identification for which the listing is prepared. A detail record must be used for each entry (unless all programs are to be listed). ALLPG is entered in the ident field when all programs are required.

LIST TAPE OVERLAY NAMES (>>> LTON) MAGNETIC TAPE VERSION.
The purpose of this function is to assist the programmer with the assignment of program and overlay names. A listing of all the program and overlay names (includes those used by the MCP) is printed.

When assigning names to programs and overlays, the list can be referenced to prevent duplications.

The LTON Function format is as follows:

```
>>> L T O N
1 - - - - - 7
```

DELETE FUNCTION FROM USERS SYSTEM FILE CALL (≥ ≥ ≥ DELF)
MAGNETIC TAPE VERSION.

This function allows the user to customize the disk version of MCP II by eliminating those functions that are not required by the user operation.

When DELF is used to delete functions that contain overlays (such as Sort or an Assembler) the user can maximize the disk savings by also eliminating each of the associated overlays.

Care must be taken not to delete any of the functions that are automatically called by the Executive or Multiprogramming Controller.

The DELF Function format is as follows:

```
≥ ≥ ≥ D E L F x
1 - - - - - 7 8
```

where:

a. x - input device. This field designates the type of input used for the deletion card or SPO message.
   C - punched card input.
   S - supervisory printer input.

The Deletion Input format is as follows:

```
i i i i
1 - - 4
```

where:

iiii - function identification

The deletion cards or messages must be in the same sequence as they occur on the Master System Tape. The LDON listing of the MCP (after it has been loaded onto disk) will show the function sequence and appropriate function names that can be used for this purpose.

ADD FUNCTIONS TO MCP II (> > > ADDR) MAGNETIC TAPE VERSION.
The purpose of this function is to allow the user to incorporate "in-house" auto-load formatted functions into the MCP II disk version, and to customize the MCP.

The first word of the auto-load programs to be added to the MCP must contain the following format:

+ > 0 V R > i i i z c c

where:

a. iiiz - four character function name. The z parameter must contain a 12 (R) zone if the function is to be executed in memory locations 400 through 79@; otherwise, the function will be called and executed from location 800.

b. cc - block count of the program being added.

The ADDR Call format is as follows:

> > > ADDR
1 - - ---7

NOTE

Programs executing from machine location 400 should not exceed one block. The automatic calling of Save/Restore will not be performed for those programs executing from machine location 400.

LOAD PROGRAM ADD TAPE TO DISK LIBRARY (≥ ≥ ≥ LPAT) DISK VERSION. The purpose of this function is to initialize a User Program Library or to add programs to an existing User Program Library. A program library listing is created that indicates the beginning disk file address of each program, and the number of blocks loaded onto disk. A SPO message will indicate the beginning and ending disk address of the disk file User Program Library. The beginning disk address of the User Program Library is stored in the Executive Routine at machine location 11#.

The switching of libraries may be accomplished with this function. When it is necessary to have more than one User Program Library (e.g., for debugging purposes), control can be passed between User Program Libraries (once they are loaded) by initiating the LPAT Function, stating the number of programs, overlays, and the disk address where the alternate library had previously been loaded. If a tape unit is designated as #2 it must be in local status. When the message IS THIS CORRECT is printed on the SPO, an answer of YES must be returned. When the system hangs on a tape command, press CLEAR and then CONTINUE on the Central Processor to return control to the Executive Routine, and then any program in that User Program Library may be called in the usual manner.

By placing a U in position 14 of the LPAT Function Call instead of the normal beginning disk address, the adding of programs to an existing User Program Library may be accomplished after a library has been established and sufficient room is available for the addition. When initializing a User Program Library, it is advisable to allow for additional programs and overlays.

The input tape must be formatted (by the PADR Function for Assembler Programs) in such a way that the Call Record precedes each program, and the program is in 480 character binary blocks (PADR format).

The LPAT Call format is as follows:

```
≥ ≥ ≥ L P A T  p  p  p  o  o  o  d  d  d  d  d  d  d
1 - - - - - 7  8  -  10 11 -  -  14 15 -  -  -  -  20
```

where:

a. ppp - number of programs being loaded. If for an initial library creation, it designates the anticipated number of programs to be included in the User Program Library.

b. ooo - number of overlays being loaded. If for an initial library creation, it designates the anticipated number of overlays to be included in the User Program Library.

c. ddddddd - base disk address where the User Program Library is to be loaded. If position 14 contains the alphabetic character U, it signifies that the programs being loaded are to be added to the existing User Program Library.

NOTE

The CPAT Function must be used whenever COBOL programs are loaded or added to a User Program Library.

Figure 4-2 is an example of a LPAT initiated library listing.

```
LIBRARY LISTING AS OF=09-09-71.
PROGRAM IDENTIFICATION  INITIAL BLOCKS TO CALL IN  TOTAL BLOCKS  TYPE  DISK FILE BEGINNING ADDRESS

   OP3T1                    12                0012          R              0009016
   OP3T2                    14                0014          R              0009040
   OP3T3                    16                0016          R              0009068
   OP3T4                    14                0014          R              0009100
   AYTST                    08                0008          R              0009128
   CDTAP                    04                0007          7              0009144
   TPPCH                    04                0007          7              0009158
   TPTAP                    05                0007          7              0009172
```

Figure 4-2. Example of LPAT Library Listing

>>> LPAT
continued

LOAD COBOL COLLECTOR TAPE TO DISK LIBRARY (≥ ≥ ≥ CPAT) DISK VERSION. The purpose of this function is to load or add COBOL programs to the User Program Library from a COBOL collector tape.

The use of this function eliminates the need for the PADR Function. (PADR is required with assembler type programs.) The direct loading of COBOL programs to the User Program Library is accomplished with this function. After a library of different types of programs (COBOL and Assembler) has been created on disk, a copy of the library may be dumped to tape using the DPDL Function. The LPAT Function must then be used to reload the back-up-program tape.

The CPAT Function format is as follows:

```
≥ ≥ ≥ C P A T  p  p  p  o  o  o  d  d  d  d  d  d  d
1 - - - - - 7  8  -  10 11 -  13 14 -  -  -  -  -  20
```

where:

a. ppp - number of programs being loaded. For an initial library creation, it designates the anticipated number of programs to be included in the User Program Library.

b. ooo - number of overlays being loaded. For an initial library creation, it designates the anticipated number of overlays to be included in the User Program Library.

c. ddddddd - base disk address where the User Program Library is to be loaded. If position 14 contains the alphabetic character U, it signifies that the programs being loaded will be added to the existing User Program Library.

NOTE

The CPAT Function must be used whenever COBOL programs are loaded or added to the User Program Library.

DISK WORD CORRECTOR (≥ ≥ ≥ DWCR) DISK VERSION.
The purpose of this function is to provide the user with the capability to change any word (12 characters) in any program of the User Program Library.

This function is based at machine location 800 and utilizes six blocks of core.

The following information describes how the DWCR Function operates.

a. Each word to be changed requires a Corrector Card or SPO message.

b. The Call or Padd Record for the referenced program is used to build the address of the specified program block.

c. Call records are located only when program identity changes.

d. All changes within the same block (if in sequence) are made before the block is rewritten onto the disk file.

e. Corrections may be processed in any order; however, a recommended sequence is as follows:

   1) Program identity sequence should be in the same order as the additions to the library (reference the LDON Library Listing).

   2) All Corrector Cards or messages for a program should be together, and grouped by program block numbers.

f. Provision is made for changing the Call or Padd Record by using CALL as a block number.

g. After each change is completed, a record of the change is printed.

h. If a change is not completed, a message is printed indicating this fact (along with the reason).

i. The Word Corrector listing is grouped by program. Each new program causes a skip to the heading of a new page.

j. Before a change can be made, the Corrector Card or message must pass the following validity tests:

1) The program number must be a valid program contained in the User Program Library.

2) The block number cannot exceed the total number of blocks in a specified program.

3) The Old Word Field must compare equally with the actual information contained in the User Program Library. (This check is the reason for requiring a fixed 12-character field for each change.) An equal compare on 12-characters reduces the possibility of changing the wrong field.

k. An End-of-File card or message is used to indicate that all the corrections have been processed.

The Disk Word Corrector Function format is as follows:

```
≥ ≥ ≥ D W C R i
1 - - - - - 7 8
```

where:

a. i - input device. This field specifies the type of input device.

C - card reader.
S - supervisory printer.

The detail card or SPO message format is as follows:

```
                                      old word                      new word
≥ ≥ ≥ W C O R i i i  i  i  b  b  b  b  w  w  r  r  o  o  o  o  o  o  o  o  o  o  o  o  r  r  n  n  n  n  n  n  n  n  n  n  n  n
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
```

where:

a. iiiii - program identification in which a word change is made.

b. bbbb - block number in which the word is being changed.

c. ww - word number being changed.

d. rr - reserved for the system.

e. ooooooooooo - Old Word being deleted.

f. nnnnnnnnnnn - New Word. Replaces the information in the old field.

Before the Word Corrector input is executed, the detail information must pass the following validity tests.

a. The program identification number must match a valid program number contained in the library.

b. The block number cannot exceed the total number of blocks specified in the Call Record Program.

c. The Old Word Field must compare equally with the information contained in the library. The check is a 12-character compare.

An End-of-File card or message is used to signify that all of the corrections have been processed.

LIST DISK LIBRARY PROGRAM (≥ ≥ ≥ LDPL).
The purpose of this function is to obtain a listing of any program in the User Program Disk Library. A listing similar to that created by the PADR Function (when the program was originally loaded to the library) is made (figure 4-3).

The LDPL Function format is as follows:

```
≥ ≥ ≥ L D P L i
1 - - - - - 7 8
```

where:

a. i - input device. This field specifies the type of input device for the program identification record(s).
   C - card reader.
   S - supervisory printer.

The format for the detail card or SPO message is as follows:

```
≥ ≥ ≥ L D S K i d e n t
1 - - - - - 7 8 - - - 12
```

where:

a. ident - the program identification for which a listing will be made. A record must be entered for each program. If a listing of the complete User Program Library field is required, ALLPG is entered in the ident field.

```
PROGRAM IDENTIFICATION: CDTAP, AS OF 09-09-71                                  PAGE 001
BLK NO.  WORD NO       WORD NO. + 0     WORD NO. + 1     WORD NO. + 2     WORD NO. + 3     WORD NO. + 4

 CALL       00         0009144CDTAP     ZN1      1          040007


PROGRAM IDENTIFICATION: CDTAP, AS OF 09-09-71                                  PAGE 002
BLK NO.  WORD NO       WORD NO. + 0     WORD NO. + 1     WORD NO. + 2     WORD NO. + 3     WORD NO. + 4

 0001       00         +≥0VR≥ASI801     502713304470     + 400712480      135429715715     112440720720
 0001       05         542304720420     + 1466400083     703306   722     703306   309     122304369349
 0001       10         703712   513     +  000306        707144   730     127718005725     K02725000550
 0001       15         K82550680550     702304   710     K22730000580     KA2580680580     705715   389
 0001       20         707730   392     707725   220     703722   289     521722630678     703712   083
 0001       25         703712   183     702710   387     7×3722000166     01 686DSK RD      ERR = DISC+
 0001       30         61 000           004000000060     000000000000     0000002
 0001       35         117012392392     K22392400020     543316400060     502181387410     709380   400

 0002       00         117012392392     K22392400020     543316400060     502181387410     709380   400
 0002       05         7×3183410623     702387000256     703450   289     J  000289      221256092256
 0002       10         60 110110080     543126336130     703289000336     703136456169     703146400186
 0002       15         502181387270     T18210000456     702387000256     J  00040058    221256192256
 0002       20         60 210210180     J  00318627      111161161161     541230161156     J  001169793
 0002       25         70125200 161     54216#189150     702453400281     750400000SET     70228104 304
 0002       30         J  000289000     700468ASG400     700568000500     700668000600     734768000700
 0002       35         111350326326     54136#326136     7×1378000324     +090348  000     000000000

 0003       00         +8EGCDTAPA50                      +000000000≥+                      61 A50**0 18
 0003       05         80050 YY  SC     DTAP 0010000     00000000                          9999
 0003       10         995000009258     9+ LABEL8505     0      SCDTAP    002
 0003       15                                                      90050          CDTAP 003
 0003       20         +#0FILE= 900     +02FILE= 900
 0003       25         95050       C    DTAP 004
 0003       30                                  +005     0       CDTAP    005
 0003       35                                                      +5050          CDTAP 006

 0004       00                  +                CARDS00     000000           610700           +LABEL ERR+
 0004       05         A0050       C    DTAP 007702A     56A503967019     59   A927019     59   B027038
 0004       10         00   500T105     90560560A505     0 W,,S CDTAP      00A#018800C7     09727028168≥
 0004       15         0399610540           76897201      +40125836A0     9A0990050 1W      ,)CDTAP 009
 0004       20         702856850396     701963   C02     701963   C28     701963   C32     703C30   500
 0004       25         85050 W,,,SC     DTAP 010T905     905605607018     2#   A087030     2404137002I+
 0004       30         40C71A307020     44C60399C005     0  ,S1WCDTAP      011610540          610A50
 0004       35             702C76C7     0396701963       012701963        022C5050  S      W,,CDTAP 012
```

Figure 4-3. Example of a LDPL Listing of CDTAP

DELETE PROGRAMS FROM DISK LIBRARY (> > > DPDL) DISK VERSION.
The purpose of this function is to delete user programs from the User Program Library.

The following procedure is used during the execution of the function.

a. The Call or Padd Record for each specific program is located, and the program name is replaced with the word DELETE.

b. The End-of-File card or message indicates that all the deletions have been processed.

c. All remaining user programs on the disk file are written onto magnetic tape as follows:

   1) Each call record is checked.

   2) If flagged with DELETE, the record is ignored.

   3) If it is not flagged with DELETE, the CALL Record is written onto magnetic tape. The CALL Record locates the program and formats it from a disk file program to a Program Add Tape Format.

   4) A count of the number of programs converted to magnetic tape is printed on the supervisory printer.

d. The Load Program Add Tape to Disk Library Function (LPAT) can be used to reload the User Program Library.

The DPDL Function format is as follows:

```
> > > D P D L i
1 - - - - - 7 8
```

where:

a. i - input device. This field specifies the type of input device for the delete records.
   C - card reader.
   S - supervisory printer.

The delete card or SPO message used as input for program deletion is as follows:

```
≥ ≥ ≥ P D E L i d e n t
1 - - - - - 7 8 - - - 12
```

where:

a. ident - the identification of the program to be deleted from the User Program Library.

NOTE

The DPDL Function can be used to create a tape back-up of the User Program Library by entering (≥ ≥ ≥ bEOF) a standard End-of-File card or message. The function will copy the User Program Library to tape with no deletions.

LIST DISK OVERLAY NAMES (>>> LDON) DISK VERSION.
The purpose of this function is to prevent the assignment of duplicate names to programs or overlays within the User Program Library.

The LDON Function format is as follows:

```
>>> LDONi
1 - - ---78
```

where:

a. i - input.
L - listing. This entry will list only the program overlays from the User Program Library.
blank - will list all the names and addresses from the User Program Library.

Figure 4-4 is an example of an LDON listing after a User Program Library has been created.

```
DISK OVERLAY NAMES AND ADDRESSES AS OF-09-09-71.
    OP SYSTEM OVERLAYS.
        FOJF
        OCKF
        NOAF
        AOAF        Automatic Functions
        OSCL
        IFCC
        FCOF
        STTF  0007556
        SAVO  0007560
        FECH  0007638
        SOCM  0007640

    PROGRAM OVERLAYS.
        OP3T1 0009016
        OP3T2 0009040
        OP3T3 0009068
        OP3T4 0009100
        AXTST 0009128
        COTAP 0009144
        TPPCH 0009158
```

Figure 4-4. Example of a LDON Listing

COBOL SOURCE PROGRAM MAINTENANCE FUNCTION (>>> COBL) DISK ONLY. The purpose of this function is to provide the user with the capability of maintaining a magnetic tape file of COBOL source programs.

The function capabilities are as follows:

a. Create a Source Program Tape.
b. Add or delete program(s) to or from the Source Program Tape.
c. Revise, add, or delete Source Program records.
d. List or punch Source Program information.

The implementation of this function requires a working knowledge of the operating requirements.

The operating functions are as follows:

a. A scratch tape is mounted on MTU-1 for the new Source Program Tape.

b. The Source Program Tape is updated by mounting a previous tape on MTU-2.

c. If programs are added from an update tape, they are mounted on MTU-3.

The COBL Call format is as follows:

```
>>> C O B L
1 - - - - - 7
```

The first record of each source program must have the following format.

```
i d e n t  b  L A B E L
1 - - - 5  6  7 - - - 11
```

where:

a. ident - a unique five character program name (the first four characters cannot be CALL).

b. b - blank.

c. LABEL - the literal LABEL.

The Source Program Tape Record addition, or revision format is as follows:

```
s s s s s s  x x x x x
1 - - - - 6  7 - - - 80
```

where:

a. ssssss - the record sequence number to be added or revised.
b. xxxxx - Source Program coding as required.

The Source Program Tape record deletion format is as follows:

```
s s s s s s  D E L E T E
1 - - - - 6  7 - - - - 12
```

where:

a. ssssss - the record sequence number where the record is to be removed.

b. DELETE - the literal DELETE.

The following is a list of SPO messages, and the actions to be taken by the system operator.

a. ENTER REQUEST
This message asks what operations the user wants the COBL Function to perform.

The response is as follows:

1) BUILD - creates an unlabeled 80-character card image tape of COBOL source programs. The input is through the card reader, and the new tape is on MTU-1. All of

the programs are sequenced, and a listing made of the new sequence numbers. BUILD is entered via the SPO.

2) LISTPG - returns a WHICH LIST FUNCTION message via the SPO.

The replies to this message are as follows:

a) ALL - prints the complete file.

b) PGMID - prints program identifications. If PGMID is entered via the SPO, a listing of program names submitted through the card reader are printed.

c) program identification - prints that individual program.

3) ADDPGM - returns a DEVICE message via the SPO.

The replies to this message are as follows:

a) TAPE - programs are added from magnetic tape.
b) CARD - programs are added from the card reader.

This entry copies the Source Program Tape from MTU-2 onto MTU-1, and adds new program(s) from either MTU-3 or the card reader onto MTU-1.

4) CHANGE - requires an input deck to reflect the revisions to the Source Program Tape.

The tape mounted on MTU-2 is copied onto MTU-1 until a matching program name is found. At this point the change cards (additions or revisions) replace the existing record(s) from MTU-2.

If the change record includes DELETE neither record is written on MTU-1.

If the record sequence numbers do not match, the input change records are automatically sequenced. A maximum of 50 records can be inserted between each source record. A listing is also made for the user.

An example of the update cycle using the CHANGE option is illustrated in figure 4-5.

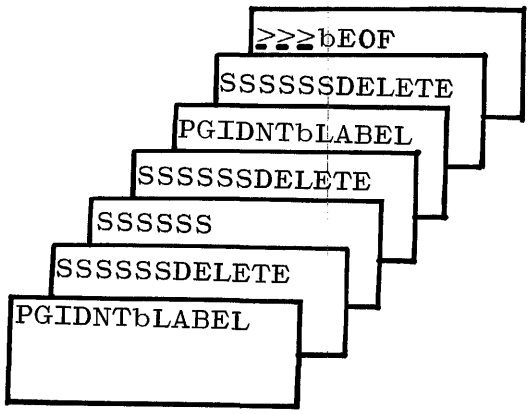


Figure 4-5. CHANGE Deck

NOTE

The input deck must be in the same sequence as the Source Program Tape.

5) PUNCH - returns a PGMID message that requires a five character program name response. This entry identifies the program to be punched.

6) REMOVE - requires an input deck be entered through the card reader. The cards must use the same format as the first source program record (columns 1 through 11). The contents of the tape on MTU-2 are copied onto MTU-1. The programs entered via the card reader are bypassed.

A listing of the programs on MTU-1 is automatically made for the user.

NOTE

All of the options must have a standard End-of-File (>>>bEOF) entry as the last input record.

The B 500 compiler will accept Source Program Tape as input.

The operating functions are as follows:

a. The Source Program Tape is mounted on MTU-2, -3, -4, -5, or -6.

b. The Library Call Card is required by the compiler.

The Library Call Card format is as follows:

```
T A P E u b x x x x x
1 - - 4 5 6 7 - - - 11
```

where:

a. TAPE - the actual TAPE.

b. u - the number of the MTU where the Source Program Tape is mounted.

c. b - blank.

d. xxxxx - the name of the program to be compiled. (The first program is compiled when this entry is blank.)

A 999 halt will occur when a program cannot be found.

SYMBOLIC TAPE MAINTENANCE CALL (≥ ≥ ≥ STMT).
The purpose of this call is to create and maintain separate Master Program Library Tapes at the program level. The input can either be symbolic source language or auto-load card image programs.

The implementation of this function requires a working knowledge of the operating requirements.

The operating functions are as follows:

a. A scratch tape is mounted on MTU-1 for a newly created Master Program Library Symbolic Tape.

b. The Master Program Tape is normally updated by mounting a binary Master Programs Library Symbolic Tape on MTU-1 and the scratch tape on MTU-2.

c. The line printer is made READY.

d. The STMT Call is entered via the card reader or the SPO.

The STMT Call format is as follows:

```
≥ ≥ ≥ S T M T
1 - - - - - 7
```

A control card must precede each source program that the user wants to Add, Delete, or Replace. The formats are as follows:

≥ ≥ ≥ ADD	Adds the following program.
≥ ≥ ≥ REPLACE	Replaces a program with the following.
≥ ≥ ≥ DELETE	Deletes specific programs.

The programs being ADDed must be placed at the end of the update control deck, and programs that are REPLACEd and/or DELETEd must be in the same sequence as the program in the Master Program Library Symbolic Tape. The sequence is determined by the verification listing made during the initial or last update cycle.

The first instruction for every program being ADDed must be as follows:

```
+ ≥ O V R ≥ i d e n  t
1 - - - - 6 7 - - -  11
```

where:

a. ident - program identification.

A Control Deck setup for the update of a Master Program Library Tape is shown in figure 4-6.

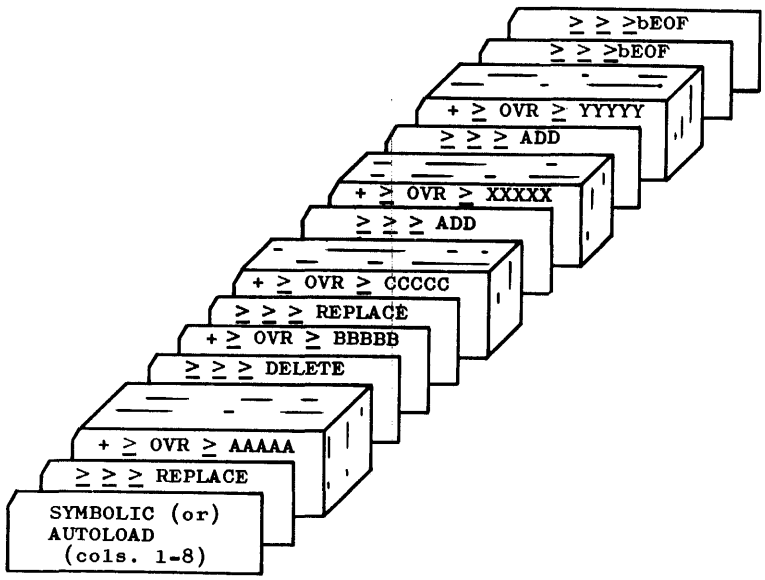


Figure 4-6. Control Deck Example

A pass is automatically provided to check the newly created Master Programs Library Symbolic Tape. This check verifies that the program has been copied correctly, and that the Master Tape was updated as specified. The program identifications are tested, which reflects the sequence of the updated version of the Master Tape.

The following is a list of the SPO messages and the actions to be taken by the user.

a. IS BINARY CARD TO TAPE REQUIRED

The message asks if the run is an initial creation of the Master Program Library Symbolic Tape, or an update.

The response is as follows:

1) YES - initial creation.
2) NO - update run.

The STMT option allows the user to create and verify a binary card image tape of any card file (regardless of context).

b. REMOVE WRITE RING

This message indicates that the update or initial creation is completed, and the verification pass is in progress. The user follows the message instructions and processes the update deck through the card reader by pressing CONTINUE on the Central Processor.

c. VERIFICATION ERROR RESTART

This message indicates that an error has occurred in the update or initial creation pass. The job is discontinued by removing the deck from the card reader, and pressing CONTINUE on the Central Processor.

d. INVALID UPDATE CONTROL CARD

This message indicates that an improperly punched Add, Delete, or Replace update control card is in the process. The job is discontinued by removing the deck from the card reader, and pressing CONTINUE.

e. FORMAT ERROR

This message indicates that the format of the first word of master tape data, or the first instruction in a program card deck is not +≥OVR≥iiiiib. The job is discontinued by removing the deck from the card reader and pressing CONTINUE.

SYMBOLIC TAPE UPDATE AND RESEQUENCE CALL (>>> STUR).
The purpose of this call is to maintain Advance Assembler symbolic Source Programs at the page and line level. The programs must be resident on the Master Programs Library Symbolic Tape created by the STMT Call.

The implementation of this function requires a working knowledge of the operating requirements.

The operating functions are as follows:

a. The symbolic tape is mounted on MTU-1.
b. The scratch tape is mounted on MTU-2.*
c. The line printer is made READY.
d. The Call STUR is entered via the card reader or the SPO.
e. Load the input deck into the card reader and press START.

The STUR Call format is as follows:

```
>>> S T M T
1 - - - - - 7
```

A control card must precede each source program that the user wants to Add, Delete, or Replace. The format is as follows:

```
>>> A D D              (additions)
>>> D E L E T E        (deletions)
>>> R E P L A C E      (replacements)
1 - - - - - - - - - 10
```

The change deck must be in page and line number sequence, because the page and line numbers in the source cards serve as the controlling factor. A control card must precede the source cards that need to be added, deleted, or replaced.

---

* MTU-2 will contain the updated and resequenced symbolic tape after the STUR Function is completed.

An End-of-File Card ≥ ≥ ≥bEOF (columns 1-7) must be placed behind the last source card of the update deck.

A pass automatically checks the newly created Master Programs Library Symbolic Tape. The check verifies that the programs have been copied correctly, and that the updating was accomplished as specified. A listing of program identification is created that reflects the sequencing of the updated Master Programs Library Symbolic Tape.

The following is a list of SPO messages and the actions to be taken by the user.

a. ENTER TAPE NAME TAPE-DISK-SCP-USER
   This is a parameter request message. The user must respond with TAPE/DISK/SCP/USER in order to obtain the sequencing in their respective categories of 1-49999/50000-99999/1-99999/1-99999.

b. FORMAT ERROR
   The last card read does not contain one of the specific control card formats for columns 1-10. After the card is corrected, the STUR Function is restarted.

c. RD ERR U1
   A read error has occurred on MTU-1. The user can press CONTINUE on the Central Processor to try again, or CLEAR and CONTINUE to abort the function.

d. REMOVE WRITE RING
   The new symbolic tape has been made and the verification pass is in progress. The user must remove the write ring from the MTU-2 tape, place the update deck back into the card reader, and press CONTINUE on the Central Processor. This completes the update verification.

e. VERIFICATION ERROR RESTART

The newly created symbolic tape has not been correctly verified. The user must press CLEAR and CONTINUE on the Central Processor to abort the function.

f. RD ERR U2

A read error has occurred on MTU-2 during the verification pass. The user can press CONTINUE on the Central Processor to try again, or CLEAR and CONTINUE to abort the function.

OUTPUT FROM SYMBOLIC PROGRAM TAPE CALL (≥ ≥ ≥ SSTO).
The purpose of this call is to obtain various types of output from Basic or Advanced Symbolic Program Master Tapes.

The implementation of this function requires a working knowledge of the operating requirements.

The operating functions are as follows:

a. The Symbolic Program Master Tape is mounted on MTU-4.

b. Scratch tapes are mounted on MTU-1 and -2 when an assembly is required.

c. The line printer is made ready when a listing is required.

d. The card punch is made ready when punched card output is required.

e. The Call (≥ ≥ ≥ SSTO) is entered via the card reader or the SPO.

f. Follow the instructions displayed on the SPO.

The SSTO Call format is as follows:

≥ ≥ ≥ SSTO
1 - - ---7

The following is a list of the SPO messages and the action to be taken.

a. ENTER 10 CHARACTER PROG HDR ID
This is a parameter request message and will re-occur after the entry is processed, or until a standard End-of-File message is processed. If all the programs on the Symbolic Program Master Tape are to be output, the response is ALL. When individual programs are affected, the response is a 10-character request of +≥OVR≥IDEN.

b. ENTER TYPE OF OUTPUT - ASBL, ASOP, PCH, LIST or P&L. This is a parameter request message and is used in conjunction with the item a message. The response is as follows:

1) ASBL - Basic Assembler Call.
2) ASOP - Advanced Assembler Call.
3) PCH - punches the specified program.
4) LIST - lists the specified program.
5) P&L - punches and lists the specified program.

c. IS OUTPUT TYPE SAME FOR ALL ENTRIES
This is a parameter request message and is used in conjunction with item b. The response is as follows:

1) YES - all subsequent entries will have the same type of output.

2) NO - an ENTER TYPE OF OUTPUT message will appear for all subsequent entries.

d. SAVE MEMORY - DESIGNATE UNIT
This is a parameter request message, and will follow item c after an Assembler has been specified. This message will appear after the first specification, and signifies the type of media on which the SAVE or SAVD Function will store memory before completing the assembly process. The response is as follows:

1) MTU number - memory is stored on magnetic tape.
2) # - punch card memory.

Each subsequent assembly process will bypass the SAVE MEMORY - DESIGNATE UNIT message, and memory will be stored on the previously specified media.

e. IS OUTPUT SAME FOR ALL ASOP PROGRAMS
This parameter request message is displayed when the first request for a program on the Symbolic Program Master Tape

and MCP II Advanced Assembler is received. If the response is YES, it will initiate a subsequent ENTER TYPE OF OUTPUT message (item b). This will automatically call ASOP. This reply will not make it necessary to repeatedly respond to this message. A NO entry will indicate a message that requires assistance from the system operator.

f. CARD OUTPUT FOR ASOP

This parameter request message appears whenever an ASOP parameter is used to assemble a program from the Symbolic Program Master Tape. (Reference item b.) The response is as follows:

1) YES - the auto-load object program card deck is punched.
2) NO - the punch operation will not occur.

g. PROGRAM TYPE - ADVAN, BASIC, AUTOL, 80-80

This parameter request message is used in conjunction with the PCH, LIST, or P&L replies to the item b message. The response is as follows:

1) ADVAN - the requested program ID identifies a program in the MCP II Advanced Assembler symbolic format (reference item a).

2) BASIC - the requested program ID identifies a program in the Basic Assembler symbolic format (reference item a).

3) AUTOL - the requested program ID identifies a program in the auto-load format (reference item a).

4) 80-80 - an exact card image must be produced for the date represented by the program-ID (reference item a).

h. IS PROGRAM TYPE SAME FOR ALL ENTRIES

This parameter request message is used in conjunction with the AUTOL or 80-80 of item g. The response is as follows:

1) YES - all subsequent PROGRAM TYPE message requests will automatically call the specified format.

2) NO - indicates to SSTO that the format type specified in item g will not be the same, and that each PROGRAM TYPE message requires a response.

i. END FLAG SAME FOR ALL 80-80 ENTRIES

This parameter request message is used in conjunction with the 80-80 reply in item g. The response is as follows:

1) YES - all subsequent END FLAG messages will automatically call the specified formats. The 80-80 function will look for the same ending label on each subsequent 80-80 request.

2) NO - all END FLAG request messages require a response.

j. END FLAG LENGTH

This parameter request message is used in conjunction with the item g 80-80 reply. The response to this message is the number of characters in the ending record flag that will be used to delimit the transfer of data to the 80-80 function.

A one character response can indicate from 1 to 12 characters of flag data entered in the following manner:

1, 2, 3, 4, 5, 6, 7, 8, 9, #, @, 0, (#=10, @=11, 0=12).

This message is repeated if an invalid character is entered by the user.

k. END FLAG POSITION

This parameter request message is used in conjunction with the item g 80-80 reply. The response to this message is the MSD (most-significant-digit) character position of the flag field within the ending record that the 80-80 function will seek to delimit the transfer of data.

The reply must be machine words and characters 000 through 067, e.g., 01@ is the 24th position of the record, and 019 the 21st.

This message is repeated if an invalid character is entered by the user.

1. END FLAG CODE
   This parameter request message is used in conjunction with the item g 80-80 reply. The response must be the actual contents of the flag field (1 to 12 characters of significant ending label flag data).

m. PROGRAM NOT FOUND ON TAPE program-ID
   This message indicates that the end-of-tape has been processed, and the specific program-ID was not found.

   The RWD Call should be used as a precautionary measure to assure the user that the tape was properly positioned. After using RWD, the user can re-enter the SSTO Call.

   If the program is truly missing, SSTO can be discontinued by using CLEAR and CONTINUE on the Central Processor.

n. flag data NO END RECORD
   This message indicates that the end of tape has been processed, and the flag data specified in item 1. was not found.

   The SSTO Function is automatically discontinued. The user must re-enter the SSTO Call, and the correct end flag data when it is requested.

SECTION 5
MCP II SORT CALL FUNCTIONS

GENERAL.

MCP II provides the user with the functions to assist in the generation of sort programs. Magnetic tape or disk file working storage can be specified by the user. The Sort IV Mark II Generator will produce object programs that can be executed in a multiprogramming environment.

The following pages within this section describe each of the sorts and their features.

The functions described in this section are:

* a. ≥ ≥ ≥ SG2T Sort/Merge Generator II (tape sorts).

** b. ≥ ≥ ≥ SGIV Sort Generator IV (disk sorts).

---

* Refer to B 500 Sort Generator II Reference Manual (1034279) for detailed information.

** Refer to B 500 Sort Generator IV Reference Manual (1034139) for detailed information.

SORT/MERGE GENERATOR II (> > > SG2T).

This function is divided into two parts: Sort Generator II, and Magnetic Tape Merge Generator.

SORT GENERATOR II.

This function provides the capability of producing efficient sort and merge programs designed to meet individual file and system requirements. Sort or merge programs can be generated for either a 9.6K or 19.2K system, and tape unit requirements range from a minimum of three to a maximum of six. The unbalanced merge technique is employed.

An optional output for the generator may be in Advanced Assembler Symbolic form. This form provides several points for modifying the generated program; otherwise, an object program is generated.

Three input specification cards provide the file parameters and system characteristics required for program generation. Some of the program features are magnetic tape label processing, sort key of up to 60 characters located in up to ten fields within the record, and the sequence in ascending or descending order. The location of the sort key within the record may be specified at generation time or, optionally, at run time. Interrupt and restart capabilities are also provided.

Program generation is based upon the selective routine method, and it in turn depends upon one or more input specification values.

The generator is a single-pass program divided into four phases. The following is a brief description of each of the phases.

AUDIT PHASE. During the audit phase of the generator, the three specification cards are read and their contents audited. The specification cards are printed in an edited format, followed by specification errors (if any). During this phase, the nucleus of a value area needed for the routine selection process is established.

ALLOCATE MEMORY PHASE. Memory requirement allocation for the generated program is based on logic requirements, I/O requirements, and user requirements. The balance of the values area requirement is inserted. If memory requirements exceed memory availability, alternate methods of generation are automatically attempted. If these alternate methods give unacceptable results, generation is discontinued.

PROCESS GENERATED PROGRAM PHASE. Generator program routines are checked and needed routines are selected from the prototype file. The output is in punched cards or written on magnetic tape. An Advanced Assembler Control Card for the generated program will be produced.

GENERATION END/ASSEMBLY CALL PHASE. The necessary housekeeping function for the generator are completed (and if final output was designated as symbolic) the program ends at HALT 999. If Auto-Load Output is specified, the ASOP Assembler is called, and the generated symbolic sort program is assembled to Auto-Load.

The following is a list of features:

a. Files may be sorted into ascending or descending sequence.

b. A generated sort program will process BURROUGHS standard beginning and ending labels for input and output tapes.

c. Unit records up to 1200 characters in length may be sorted.

d. The Sort Key may have a maximum of 60 characters.

e. The Sort Key may be located in as many as ten sort fields within the record. Each field can be alphanumeric or numeric.

f. Total control is provided at the user's option.

g. Input and output blocking of up to 1200 characters is permitted.

h. Logic routines are optimized.

i. Generated coding is optimized.

j. Internal blocking throughout a sort will be the maximum permitted by available memory.

k. Modification points have been provided in every phase of the sort.

l. A restart procedure has been incorporated into each sort.

m. The Sort Key locations can be specified at generation or at run time.

MAGNETIC TAPE MERGE GENERATOR.

Like generated sorts, a merge provides for interrupt and restart capability, along with several user modification points within the program.

The following is a list of features provided for generated tape merge programs.

a. Files may be merged in ascending or descending sequence.

b. A generated merge program will process beginning and ending BURROUGHS standard tape labels for input and output tapes.

c. Records of up to 1200 characters in length can be merged.

d. The Merge Key may have a maximum of 60 characters.

e. The Merge Key may be located in as many as ten fields within the record.

f. Input requirements.

1) Labeled and unlabeled files may be mixed.
2) Input blocking may be mixed for and between files.

g. Provision has been made to provide access to unique modification points for each input file, and common modification points for all files.

h. A restart procedure is incorporated into every tape merge.

SORT GENERATOR IV (>>> SGIV).

Sort Generator IV provides the user with a generator program for customizing sort programs. The generated programs are designed to utilize the disk file during the sorting operation.

SGIV is a single-pass multiple-phase program that generates a tailored Record or Tag Sort in either the Advanced Assembler or Auto-Load form. Programs may be generated for multiprogramming or non-multiprogramming execution. In addition to providing the sort program, the Sort Generator produces a completely documented program listing which includes numerous "modification points." Using this listing the user can modify the generated program at the symbolic level.

The disk sort employs an eight-way balance merge when sorting on a 9,600 character memory system, and a 16-way balanced merge when sorting on a 19,200 character memory system. During the internal sorting phase variable length strings are created. This ensures the least possible number of merge passes. The size of the stringing array is dependent upon the individual file parameters and the system characteristics for the program generated. The sort generator allows for the interruption of the program at any time during the sorting operation, and may be resumed simply by reloading the sort program with RESTART punched into the work area card.

The parameters are passed to the generator by three input specification cards. Prior to generation these specification cards are printed and edited. Any error or inconsistency during this edit phase will cause a message on the line printer, and generation is discontinued.

The following is a list of the Sort Generator IV features.

a. Sorts records of up to 1920 characters in length (1200 characters on a 9.6K system).

b. Sorts blocked or unit records stored on magnetic tape, disk file, punched cards, or punched paper tape.

c. Produces sorted output of blocked or unit records on magnetic tape, disk file, punched cards, or punched paper tape.

d. Sorts files on either a 9.6K or a 19.2K memory system.

e. Permits a sort key of up to 96 characters in length.

f. Allows the sort key to be located in up to ten fields within the record, and permits each of the ten fields to be either alphanumeric or numeric.

g. Arranges the file in either ascending or descending sequence.

h. Allows the termination of a Tag Sort with either address output or control record output.

i. Permits the sorting operation to be interrupted and then restarted.

j. Provides for a record count and block count during input and output phases.

k. Provides for the printing of the readable portion of unreadable input records (with an optional halt).

l. Sorts as many records as can be held on the disk work area.

m. Deletes or selects records during the input phase, under the control of a given character in any specified location within the record.

n. Processes a standard tape header and/or trailer label, also any non-standard tape header label (with or without trailer labels) up to 80 characters in length.

o. Provides for the user modification of generated programs at the symbolic level.

p. Provides an area of memory for the insertion of Translation Tables to utilize the Transfer and Translate command.

q. Operates with or without the Operating System.

r. Generates programs for execution in either a batch or multi-programming mode.

SECTION 6
MCP II ASSEMBLER FUNCTIONS

GENERAL.

MCP II provides Advanced and Basic Assemblers as self-contained functions within the disk and magnetic tape versions. The assemblers (as well as several related functions) can be initiated either programmatically or manually by issuing an appropriate function call to the Executive Controller.

The MCP II Basic Assembler is a subset of the B 300 Basic Assembler (AS014). SPO messages have been added, and most of the halt-operators eliminated.

The MCP II ASOP Assembler is a subset of the free-standing Advanced Assembler (AS016). It will produce object programs that optionally are "memory floatable" to allow multiprogramming. In addition the assembler will automatically create interrupts after the issuance of an I/O operation, so that the multiprogramming capability can be fully utilized.

Also available for use is a modified version of the ASOP Assembler which may be obtained from the latest Software Master. This assembler may be used in lieu of the BASIC and ASOP Assembler. The enhanced features include a 19.2K Disk Assembler with all ASOP features, capability to enter Basic Assembler Routines, and a resident Macro Library. This Assembler will make a significant difference in assembly time whenever a Macro Library is required.

The following pages within this section provide a detailed description of the Assembler and its related functions. These functions are listed in table 6-1.

Additional operating procedures may be found in the MCP II Operations Manual (1043783), and in the Basic Assembler Reference Manual (1035813).

Table 6-1

Assembler Function

Call	Description
≥ ≥ ≥ ASBL	Basic Assembler Call
≥ ≥ ≥ REFR	Re-reference Basic Assembler Symbolics Call
≥ ≥ ≥ ASOP	Advanced Assembler Call
≥ ≥ ≥ RFAZ	Re-reference Analyzer Call
≥ ≥ ≥ CMLT	Create Macro Library Tape Call
≥ ≥ ≥ CSTP	Create Systems Tape

BASIC ASSEMBLER CALL (>>> ASBL).

The Basic Assembler will not operate in a stacked input mode, but returns control to the Executive Controller at the end of every assembler output process. If successive assemblies are required, it will be necessary to place an ASBL Call Card in front of each symbolic deck of the batch.

The MCP II Basic Assembler output is in the form of an auto-load object program deck, and does not contain the "flotation code" needed for multiprogramming.

The peripherals required to use this assembler are: Two scratch-tapes (designated units one and two), card punch, and a line printer. The assembler may be called through the SPO or card reader.

The ASBL Function format is as follows:

```
>>> A S B L x
1 - - - - - 7 8
```

where:

a. x - type of symbolic source input.
   B - binary coded magnetic tape input.
   blank - punched card input.

>>> REFR

RE-REFERENCE BASIC ASSEMBLER SYMBOLICS CALL (≥ ≥ ≥ REFR).
The purpose of this function is to re-assign reference points within a Basic Assembler symbolic source program card deck, and to produce a reference program listing and new source card deck.

This function utilizes the card reader, line printer, card punch, and magnetic tape unit. Symbolic source input is entered through the card reader, and is followed by a standard End-of-File card.

The REFR Call format is as follows:

```
≥ ≥ ≥ R E F R x
1 - - - - - 7 8
```

where:

a. x - variable entries.
   C - stacked programs are assigned consecutive numbers.
   blank - the first page number of every program starts with number 01, and is numbered consecutively until another program in the input stack is processed.

>>> RP&L

RE-NUMBER BASIC ASSEMBLER SYMBOLICS SPECIFICATION CARD (>>> RP&L). The purpose of this card is to reassign specific page and line numbers within the Basic Assembler symbolic program source card deck.

This card must be used in conjunction with the REFR Function call. After the REFR Call is initiated, the following format for the RP&L specification card is used.

```
>>> R P & L p l
1 - - - - - 7 8 9
```

where:

a. p - page number to be assigned at the point of insertion.
b. l - line number to be assigned at the point of insertion.

ADVANCED ASSEMBLER CALL (≥ ≥ ≥ ASOP).

The MCP II Advanced Assembler returns control to the Executive Controller at the end of every assembly output process. If successive assemblies are required it will be necessary to place an ASOP Call card in front of each symbolic deck to be processed in the batch. The output produced by the MCP II Advanced Assembler will be in the form of an auto-load object program deck, an auto-load object program magnetic tape, or the auto-loads may be placed on a specified disk file area for the creation of a Program Add Tape (depending on the coding of the HEAD card). If Multiprogramming Macros are utilized, the auto-load data will contain multiprogramming flotation code.

If a macro library tape is used as input, mount the macro library tape on magnetic tape unit (MTU)#1, ready the line printer, and enter the ASOP Call. Additional operating procedure can be found in the MCP II Operations Manual (1043783). Instruction format for ASOP can be found in the Advanced Assembler II Reference Manual (1042769).

The ASOP Call format is as follows:

```
≥ ≥ ≥ A  S  O  P  t  u  b  b  b  b  b  b
1 - - -  -  -  7  8  9  10 -  -  -  -  15
```

where:

a. t - type of input device code.
   - B - binary magnetic tape.
   - C - card reader.
   - D - disk file (96-character segment).
   - E - disk file (240-character segment).
   - F - disk file (480-character segment).
   - M - BCL magnetic tape.
   - T - paper tape reader.

b. u - unit number of the specified type device. If the type of unit has been specified as being D, E, or F then this

entry must contain the high-order digit (MSD) of the beginning address where the file is to reside within the disk file.

c. bbbbbb - disk file address. The six lower digits of the beginning address where the file is to reside within the disk file. If the type of unit has been specified as being D, E, or F then position 10 may contain an N (if the type of unit specified is not D, E, or F) to suppress the punching of card auto-load output.

The following messages may be displayed on the SPO.

a. TYPE FOR HEADER NOT SPEC AS C - T - M - B - D - E - F
An invalid type of input device has been specified for the HEAD. The system operator recalls ASOP.

b. 1ST RECORD NOT HEAD
The first record of the source input is not the HEAD. The system operator must correct the situation and recall ASOP.

c. WORKING STORAGE INV
The type of working storage specified in the HEAD Record is incorrect. The system operator must correct the card and recall ASOP.

d. WORK TAPES ON x AND y
This message verifies the specifications contained in columns 31 and 32 of the HEAD Record. No system operator action is required.

e. LT bbbbbbb/eeeeeee R W/A bbbbbbb/eeeeeee
This message informs the system operator of the assigned disk file working-storage areas pertaining to the beginning (bbbbbbb) and ending (eeeeeee) addresses of the Label Table (LT) and the symbolic Record Work Area (R W/R).

f. INV I/O

Disk file input has been specified along with a magnetic tape work-storage declaration. The system operator must change the input to some media other than disk file and recall ASOP.

g. WORK TAPES ON x AND y A/L bbbbbbb

This message informs the system operator of the assigned beginning (bbbbbbb) disk file address where the object program auto-load images will be stored. Reference item d for an explanation of x and y.

h. NOT ML

This message informs the system operator that the magnetic tape mounted on MTU-1 does not contain the Macro Library. The systems operator must mount the proper tape and recall ASOP.

i. E S/u

This is a MCP II magnetic tape version request that the system operator enter a MTU. The MTU is used for memory retention. Any assembler work tape can be used for this purpose.

The format for the ASOP Head Record is illustrated in figure 6-1.

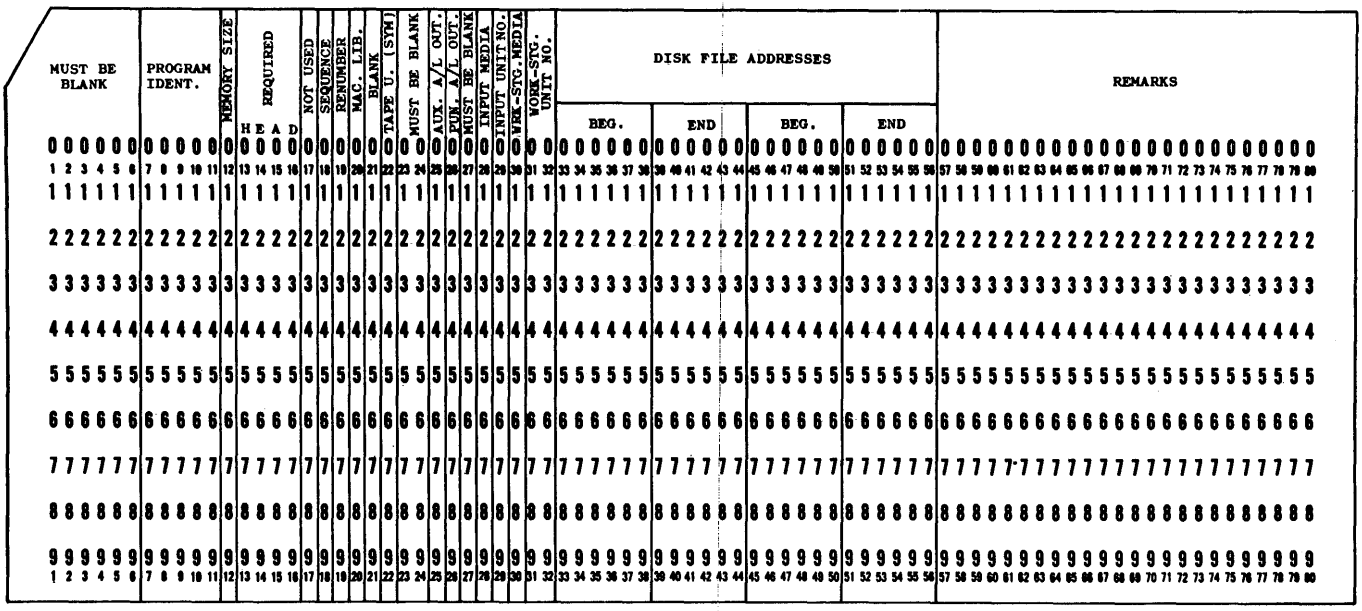


Figure 6-1. Head Card Format

Columns	Description
1-6	This field is reserved and must be blank.
7-11	Program identification that is later punched into columns 72-77 of the resultant auto-load object program deck and placed on each page of the output listing. Any combination of letters and/or digits is acceptable. When loading/updating the program library, this field should (as a reference) contain the current date (MMDDY).
12	Processor memory capacity. The acceptable codes are: a. 2 - signifies 9.6K system. b. 4 - signifies 19.2K system.
13-16	The entry HEAD must be entered in this field to identify a program header card.
17	This field is reserved and must be blank.

Columns	Description
18	Source input sequence check. The acceptable codes are as follows: a. N - sequence checking is required. b. Blank - sequence checking is not required.
19	Creates a renumbered source deck. The acceptable codes are as follows: a. A - renumbered deck is required. b. Blank - renumbered deck is not required. If A is coded, the output listing will reflect the new numbers. If column 18 is blank, the input sequence number is checked.
20	The Macro Library is created prior to, or used during assembly. The acceptable codes are as follows: a. L - Macro Library is created prior to assembly. b. M - the ASOP Macro Library magnetic tape must be present during assembly. c. Blank - macros are not required.
21	This field is reserved and must be blank.
22	Creates a symbolic listing on magnetic tape (binary). Each source input record is placed on tape in a formatted 120-character record preceded by the OP Code and M and N variants of the print instruction. The acceptable codes are: a. 1 thru 5 - indicates which MTU is designated to create a print tape. b. Blank - print tape not required.

Columns	Description
23-24	This field is reserved and must be blank.
25	Auxiliary auto-load output media. The acceptable codes are as follows: a. D - disk file. b. 1 thru 6 - indicates which MTU is designated for binary magnetic tape card images. c. Blank - none are required.
26	Punched card auto-load output/paper tape auto-load output. The acceptable codes are as follows: a. P - punched card output. b. I - paper tape output. c. Blank - indicates that neither is required.
27	This field is reserved and must be blank.
28	Input media is used as input. The codes are: a. B - magnetic tape (binary) card image. b. C - punched cards from the card reader. c. D - disk file - 96 character segments. d. E - disk file - 240 character segments. e. F - disk file - 480 character segments. f. M - magnetic tape (BCL) card image.
29	Input media unit number. The acceptable codes are: a. 1 or 2 - card or paper tape reader. b. 1 thru 5 - magnetic tape unit.
30	Working storage media that is used during process assembly. The acceptable codes are:

Columns	Description
	a. D - disk file - 96 character segments. b. E - disk file - 240 character segments. c. F - disk file - 480 character segments. d. M - magnetic tape.
31-32	Working storage media unit number or the disk file storage unit that is used during program assembly. Two units must be designated. The first unit is entered in column 31 and the second in column 32. The acceptable codes are as follows: a. 0 thru 9 - disk file. b. 1 thru 5 - magnetic tape.
33-38	Beginning address of the working-storage area on disk for the unit specified in column 31.
39-44	Ending address of the working-storage area on disk for the unit specified in column 31.
45-50	Beginning address of the working-storage area on disk for the unit specified in column 32.
51-56	Ending address of the working-storage area on disk for the unit specified in column 32.
57-80	This field may be used for remarks by the user. The remarks will be printed at the top of each page of the output listing.

RE-REFERENCE ANALYZER CALL (>>> RFAZ).
This function is used to produce a cross-reference listing of the source programs written in MCP II Advanced Assembler.

This function will process symbolic input from punched cards or magnetic tape.

The format for the RFAZ Call is as follows:

```
>>> R F A Z i
1 - - - - - 7 8
```

where:

a. i - type of input.
   B - magnetic tape input (binary).
   C - punched card input.

Error messages and additional information may be found in the MCP II Operations Manual (1043783).

CREATE MACRO LIBRARY TAPE CALL (≥ ≥ ≥ CMLT).
This function will create a Macro Library Tape. It converts the Macro Edit Routines to magnetic tape with 480-character binary records. All symbolic macro cards are read and written to tape as binary card image records. The first card of each macro must contain the macro name in columns 2-5, preceded by a plus (+) in column 1. Macro routines in symbolic form may be added to the library tape by maintaining the ascending macro name sequence, and by placing an appropriate "spec-card" with the macro name prior to the symbolic as noted in this paragraph.

The CMLT Function requires that the symbolic macro routines be arranged in an ascending macro name sequence.

The CMLT creates a Macro Directory during the symbolic card-to-tape operation and writes it to tape. This operation also includes a search routine as the last record on tape prior to the tape mark.

The CMLT Function is called by the ASOP Assembler of MCP II when column 20 of the HEAD Record is coded L. It is necessary to previously construct the Macro Library Tape prior to the execution of ASOP, because the macro routines will be required by the source program.

The multiprogramming macros obtained from the latest Software Master are used as input for this function.

A scratch Tape is needed on MTU #1. The CMLT Function format is as follows:

```
≥ ≥ ≥ C M L T
1 - - - - - 7
```

Additional operating instruction and error messages may be found in the MCP II Operations Manual (1043783).

CREATE SYSTEMS TAPE (>>> CSTP).

This function is primarily used to assemble the Operating System. It may be used to assemble Multiple Advanced Assembler Programs from a binary card image tape without manually recalling the ASOP Assembler after each assembly. Output auto-load records will be magnetic tape in binary card image form. The auto-load tape may be used as input to PADR for the creation of a program add tape.

The CSTP Function format is as follows:

```
>>> C S T P n t
1 - - - - - 7 8 9
```

where:

a. n - systems tape.
   1 - Tape/Disk Operating System.
   3 - Multiprogramming Operating System (MCP II).
   4 - Multiprogramming Operating System and 19.2K Supervisory Control Program.

b. t - symbolic listing code specified in the HEAD Record.

The CSTP Function performs all the operations necessary to create a system tape from a symbolic binary tape.

The following format is used to create a binary card image tape.

a. PADD record for a Tape Operating System.

b. Auto-load records of all the programs in the Tape Operating System.

c. Standard End-of-File Record.

d. PADD record for the Disk Operating System, or the Disk Operating System plus the Supervisory Control Programs (SCP).

e. Auto-load records for the Disk Operating System Programs.

f. If MCP II/SCP is specified, auto-load records for SCP.

g. Standard End-of-File Record.

CSTP passes the necessary parameters to call in PADR and create the system tape.

NOTE

The PRTB Function of MCP II is designed to print BCL records; however, this function may be used if the tape reads are manaully changed to a binary read.

SECTION 7
OPERATING SYSTEM ASSEMBLER

GENERAL.

The Operating System Assembler (ASOP) is an enhanced version of Advanced Assembler II. Primary features include name and program point labeling (with entry incrementing and decrementing capabilities), Control Transfer and Edit Commands, a program debugging package, memory dumps, library capabilities, and a large complement of pseudo and macro commands.

INPUT CAPABILITIES.

The Operating System Assembler permits the input source program to be in the form of punched cards, punched paper tape, magnetic tape, or disk file.

PUNCHED CARD.

Punched Card input is any valid character.

PAPER TAPE.

Paper tape input must be an eighty character card image format and contain any valid character. The input tape is not rewound before assembly.

MAGNETIC TAPE.

Magnetic tape input can be in a BCL or binary mode, but both modes must contain card image records. Binary records may contain any valid character. Group marks cannot be used in the BCL mode. The input tape will not be rewound prior to assembly, thus allowing multiple assemblies from the same tape. Output from an assembly may be produced on the same tape as the input source program, but the input source program is destroyed.

DISK FILE.

The disk file input must be blocked as follows:

a. Block size must be 480 characters.
b. Each block must contain six card image records.

The input data must be located at the disk address specified in the program header card. If the head record is on the disk file it may be either the first record of the first block, or located in a separate block. Input is destroyed during the assembly process.

ADVANCED ASSEMBLER LANGUAGE.

Throughout this section references will be made to the coding form illustrated in figure 7-1. The fields and their various uses are described in the following paragraphs.

CODING PROCEDURES.

Each line of coding form represents one entry (an instruction or a constant) that is divided into nine fields. In preparing the symbolic program, it is important that all of the entries be made in the correct fields and columns. References are made in the following paragraph to any "acceptable character." A question mark is not a valid character.

PAGE (COLUMNS 1-3).

The page number is a three-position alphanumeric field that is used to sequence multiple sheets of coding.

LINE (COLUMNS 4-6).

The line entry is a sequence number for each line of user code. This field is usually numeric with the units position (col. 6) reserved to indicate the insertions of additional entries after initial coding is complete.

If sequencing is required, columns 1-6 will be checked during assembly by the standard collating procedure.

SYMBOLIC LABEL (COLUMNS 7-12).

The symbolic label is a group of characters that serve as a name of an address in memory. The assembly process will assign a unique address to each unique symbol appearing in the program. Two types of entries are permissible as symbolic labels: Symbolic Names and Program Points.

Burroughs ADVANCED ASSEMBLER II CODING FORM

PAGE 1 2 3

PROGRAMMER _____ DATE _____

PROGRAM IDENT. _____

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
4 5 6		7 8 9 10 11 12	13 14 15 16	17 18	19 20	21 22 23 24 25 26 27	28 29	30 31 32	33 34 35 36 37 38 39	40 41	42 43 44	45 46 47 48 49 50 51	52 53	54 55 56	57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
01															
02															
03															
04															
05															
06															
07															
08															
09															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															

Printed in U.S. America

Form 1041969

Figure 7-1. Coding Form

SYMBOLIC NAME. A symbolic name is a collection of from one to six alphanumeric characters (including blanks). The first character must appear in column 7 of the Symbolic Label Field. Special characters are permissible in any position of the symbolic label except the first. Characters other than a decimal point should normally be used (figure 7-2).

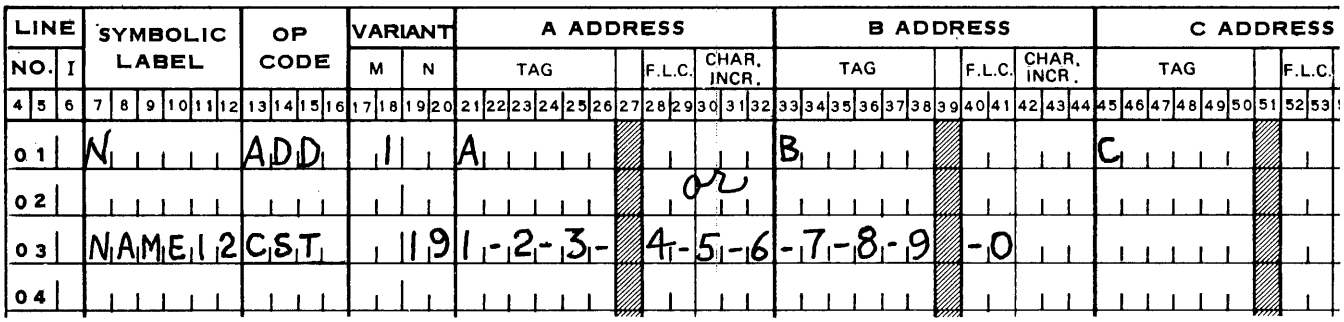


Figure 7-2. Acceptable Symbolic Names

PROGRAM POINTS. These points are reusable two-character labels, the first character is a decimal point and the second character is alphabetic (A through Z). Each alphabetic character may be used a maximum of 100 times. The relative position of each program point address indicates its use.

Between two program point addresses having the same character, a programmer may prefer to minus that specific character (meaning the last program point defined with that letter) or he may prefer to plus that character (indicating the next symbolic program point of that particular character, figure 7-3).

OP CODE (COLUMNS 13-16).
Standard mnemonic operation codes are a left-justified entry. Table 7-1 lists the acceptable mnemonic operation codes. This field is also used for assembler control commands.

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.
01		.A	TFR		3	+F			@00			+R		A
02		.C	TCB		3	+H			+R			-A		003
03		.F	TFR		2	-A		B				+K		
04			BRU			-A								
05		.H	ADD	1	2	-A			+K			+K		
06			HLT	2	0	+F								
07		.K	CST		2	00								
08		.R	BRU			-F								
09		.A	HLT	9	9									
10		.F	BRU			-A								

Figure 7-3. Program Point Usage

Table 7-1

B 500 Standard Mnemonic
Operation Codes

Mnemonic Operation Code	Operation
ADD	Add
SUB	Subtract
MUL	Multiply
DIV	Divide
ADM	Address Modification
CAE	Compare Alphabetic Branch on Equal
CZE	Compare Zone Branch on Equal
CNE	Compare Numeric Branch on Equal
CAU	Compare Alphabetic Branch on Unequal
CZU	Compare Zone Branch on Unequal

Table 7-1 (cont)

B 500 Standard Mnemonic
Operation Codes

Mnemonic Operation Code	Operation
CNU	Compare Numeric Branch on Unequal
BRC	Branch Conditional
BRU	Branch Unconditional
ICR	Interrogate Card Reader
IPR	Interrogate Paper Tape Reader
ICP	Interrogate Card Punch
IPP	Interrogate Paper Tape Punch
ILP	Interrogate Line Printer
IPL	Interrogate Lister
ISP	Interrogate Supervisory Printer
IMR	Interrogate Magnetic Tape Unit Read
IMW	Interrogate Magnetic Tape Unit Write
TSS	Interrogate Sense Switch
BBE	Interrogate Bit branch on Equal
SBT	Interrogate Set Bit
RSB	Interrogate Reset Bit
BBU	Interrogate Bit Branch on Unequal
NOP	No Operation
HLT	Halt and Branch
TFR	Transfer Character
TCB	Transfer Character and Branch

Table 7-1 (cont)

B 500 Standard Mnemonic
Operation Codes

Mnemonic Operation Code	Operation
TFZ	Transfer Zone
TZB	Transfer Zone and Branch
TT1	Transfer and Translate (Table 1)
TT2	Transfer and Translate (Table 2)
TT3	Transfer and Translate (6 bit to 12 bit)
DDC	Data Compress
DEC	Data Expand
MSK	Mask
CRD	Card Read
CRI	Card Read Branch Busy
CRB	Card Read Binary
PCH	Punch Card
PBN	Binary Card Punch
PRT	Print on Line Printer
PRL	Print on Lister
PLN	Print on Lister
PLM	Print on Lister
SKP	Skip/Space on Line Printer
SKL	Skip/Space on Lister
SLL	Slew Lister
SPO	Print on Supervisory Printer

Table 7-1 (cont)

B 500 Standard Mnemonic Operation Codes

Mnemonic Operation Code	Operation
SPR	Supervisory Printer Read
SRD	Sorter-Reader Demand
SRF	Sorter-Reader Flow
CTL	Control Sorter
TRD	Magnetic Tape Read
TWR	Magnetic Tape Write
TER	Magnetic Tape Erase
BSP	Magnetic Tape Backspace
RWD	Magnetic Tape Rewind
MWR	Magnetic Tape Memory Write
BWR	Magnetic Tape Write Binary
BRD	Magnetic Tape Read Binary
PRD	Paper Tape Read
PSF	Paper Tape Space Forward
PSB	Paper Tape Space Backwards
PRW	Paper Tape Rewind
PWR	Paper Tape Write
DFW	Disk File Write
DFR	Disk File Read
DFC	Disk File Check
DFI	Disk File Interrogate

Table 7-1 (cont)

B 500 Standard Mnemonic
Operation Codes

Mnemonic Operation Code	Operation
DCR	Data Communication Read
DCW	Data Communication Write
DCI	Data Communication Interrogate

VARIANT (COLUMNS 17-20).

For transfer type commands or any of the six permissible compare commands, the Variant Field is treated as a four-digit numeric field. For other commands it is separated into M and N variant fields of two characters each. The M and N variants can represent any value indicated by the Systems Reference Manual (1042769). Leading zeros are permitted (but not required) for any numeric variant.

If the variant is part of the machine language operation code, the variant field may be left blank and the assembler will insert the proper character. If it is required to override the assembler, insertion of the M and/or N variant (or both) may be forced by inserting an @ sign in the high-order position of the field, and the desired character in the lower position of the variant (figure 7-4). The insertion character will not be error checked.

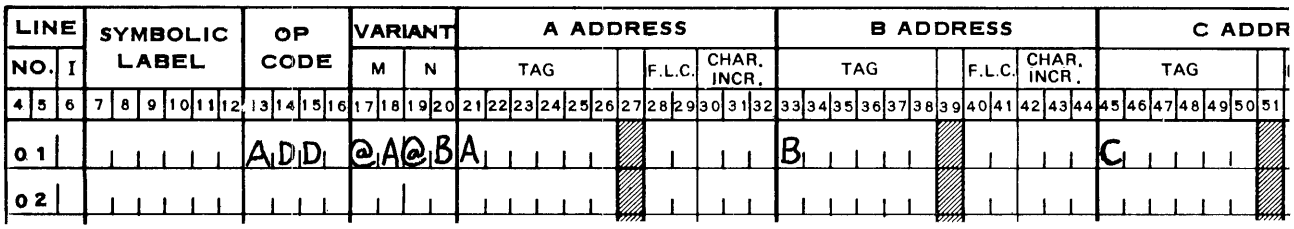


Figure 7-4. Forced M and N Variants

To force the M and N variant in a transfer type command, a single @ sign is entered in column 17, followed by the two characters to be inserted (figure 7-5).

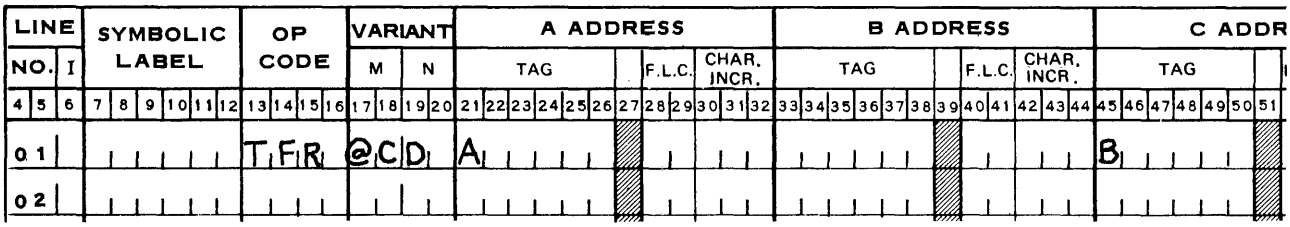


Figure 7-5. Forced Transfer Variants

For transfer and compare type of commands, the variant field is normally a four-digit numeric field indicating the number of characters to be operated on. If the number of characters specified is greater than 132 (transfer type commands), or greater than 12 (compare type commands), an error flag is printed.

A, B, AND C ADDRESS FIELDS (COLUMNS 21-56).
All three address fields have essentially the same characteristics; therefore, only the entries for one will be described. Each Address Field is divided into a tag forced last character and character increment. The Tag Field becomes the base or starting address while the character increment is used as an extension of the base address. The paragraphs that follow describe the coding for each segment within the address field.

TAG. The following type of addressing is permitted.

a. Symbolic name.
b. Program points.
c. Self-addressing.
d. Machine absolute.

Symbolic Name.
Symbolic names refer to the address assigned to the corresponding symbolic name in the label field (figure 7-6).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDR TAG
4 5 6		7 8 9 10 11 12	13 14 15 16	17 18	19 20	21 22 23 24 25 26	27 28 29	30 31 32	33 34 35 36 37 38	39 40 41	42 43 44	45 46 47 48 49 50 51
01			BRU			START						
02			CST		9							
03			CST		4							
04		START	TFR		3	A						B

Figure 7-6. Symbolic Name

Program Point.

Program points are entered as either plus or minus, and followed by an alphabetic character. A minus followed by a single letter (figure 7-7) refers to the last symbolic label identified by a decimal point followed by that particular letter. A plus and a letter refers to the next symbolic label identified by a decimal point, followed by that character. If the symbolic label field is identified by a decimal point followed by a character, and the address field refers to that same character as a program point, the minus program will reference the current line of the program. The plus program point refers to the next symbolic label identified by that character.

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDR TAG
4 5 6		7 8 9 10 11 12	13 14 15 16	17 18	19 20	21 22 23 24 25 26	27 28 29	30 31 32	33 34 35 36 37 38	39 40 41	42 43 44	45 46 47 48 49 50 51
01			BRU			-X						
02												

Figure 7-7. Program Point as an Address

Self-Addressing.

Self-addressing is indicated by an asterisk in the high-order position of the tag field of an address, followed by five blanks. It refers to the address of the OP Code position of the entry being written. Reference figure 7-8. The B Address is transferred to the OP Code Field of this instruction.

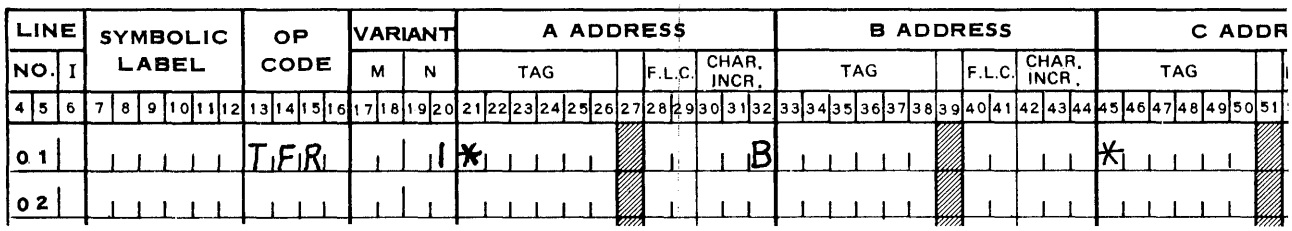


Figure 7-8. Self-Addressing

Machine Absolute.

Machine absolute coding is indicated by an @ sign in the high-order position of the tag field of the address, and followed by a three-character machine address constant user entry (figure 7-9).

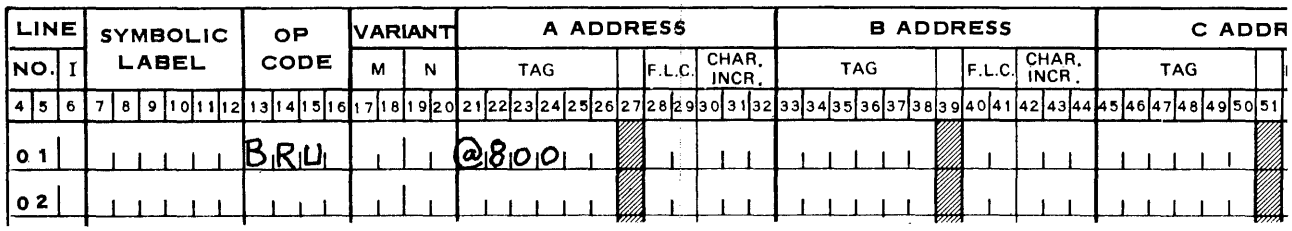


Figure 7-9. Absolute Address

LITERALS. Literals may be packed in the unused fields of an instruction (figure 7-10).

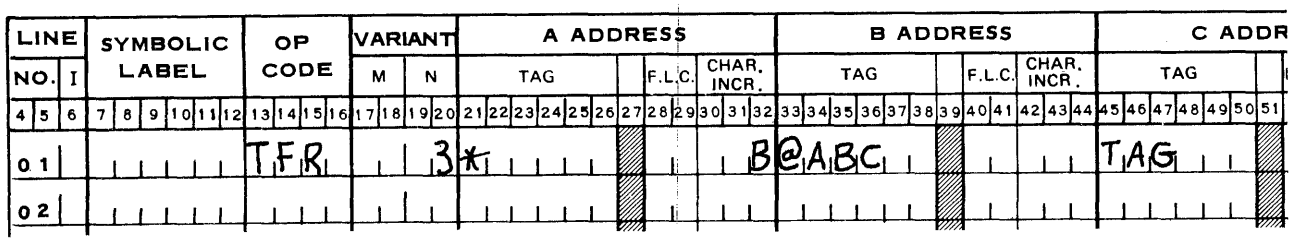


Figure 7-10. Packing Literals Within An Instruction

In addition, literals are permitted in the A Address of an ADM command, and the B address of DEC and DCC commands. This special case applies only to these commands. The literals are written with a # sign, followed by one to four numbers whose value is 0 through 1199 (figure

7-11). The numeric is left-justified and refers to the number of characters to be added. The number is converted to the appropriate number of words and characters for inserting a machine language command. Preceding zeros may be used provided the length of the literal (including the # sign) does not exceed five characters.

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDR TAG
01			ADM			#13						
02				or								
03			ADM			#013						
04				or								
05			ADM			#0013						

Figure 7-11. ADM Literal

CHARACTER INCREMENT. Character incrementation will cause the tag address to be modified by the number of words and characters entered in the Character Increment Field (figure 7-12). The two most significant digit positions represent the word(s) increment, and the least significant digit position represents the character increment. Character increment is described as follows:

a. 0 - zero increment.
b. ∅ - OP code position of word.
c. 1 or M - M variant position of word.
d. 2 of N - N address position of word.
e. 3 or A - A address position of word.
f. 4 or 5 - Fourth or fifth position of word.
g. 6 or B - B address position of word.
h. 7 or 8 - Seventh or eighth position of word.
i. 9 or C - C address position of word.
j. # or T - Tenth position of word.
k. @ or E - Eleventh position of word.

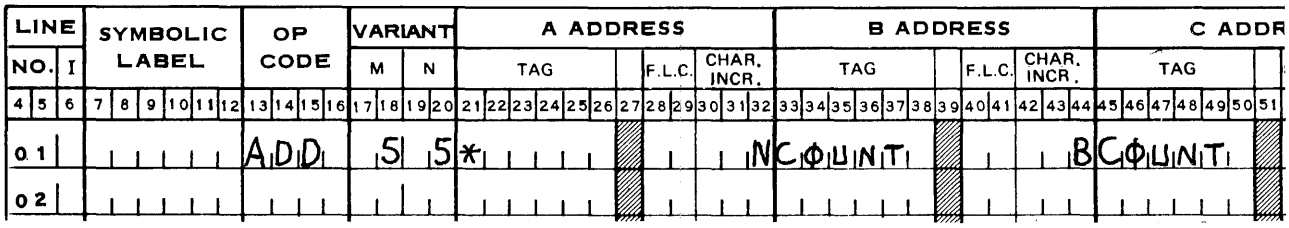


Figure 7-12. Character Increment

F.L.C. (FORCED LAST CHARACTER). The F.L.C. entry is the final step for developing an address by the assembler. This entry causes the specific character to be inserted into the low-order position of the assembled address (figure 7-13). The forced last character takes precedence over the resultant address formed by the tag and character increment.

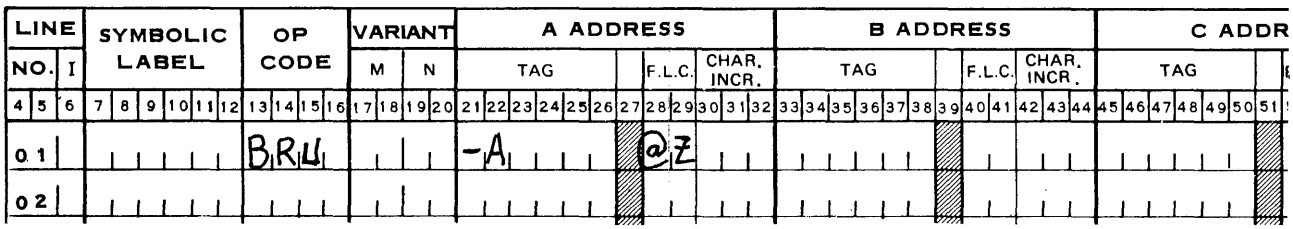


Figure 7-13. Forced Last Character

REMARKS (COLUMNS 57-80). This field provides the user with a means of documenting his output listing. Any acceptable character may be entered in this field.

PSEUDO INSTRUCTION.

Pseudo and macro instructions are important features of this assembler. The two types of instructions are written in the same general format as machine instructions, but their affect on the assembly program is quite different.

Macro instructions can be described as generative type instructions; whereas, pseudo instructions may be defined as information given the assembler in the form of a symbolic instruction that describes the

program being assembled, or a specified manner of assembly. A pseudo instruction does not usually generate object code in an assembled program.

Pseudo operation code is written in the operation field while the variant and address fields contain the information to define the effect of the pseudo operation. If a symbolic label is associated with a pseudo instruction (other than EQU) it will be assigned the current setting of the location counter (the internal counter the assembler uses to keep track of assigned addresses) prior to any adjustment of the location counter by the pseudo operation.

The following pseudo operations are treated as entries, and are documented as follows:

SLC (SET LOCATION COUNTER).

The SLC is a pseudo instruction that sets the location counter to the value entered in the A Address Field. The A address may be a machine language address or a previously defined symbolic name (see figure 7-14). Character incrementation may be used to advance the location counter ahead of a current value. If the location counter is set backward, or if the A address contains a non-existent machine address for a specific object system, an error message will be printed.

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDR TAG
01			SLC			@M0I0						
02					ar							
03			SLC			INPUT						
04					ar							
05			SLC			*		25				

Figure 7-14. Adjusting the Set Location Counter

ALC (ADJUST LOCATION COUNTER).

The ALC instruction will adjust the location counter indicated by the M variant. If the low-order position of the M variant has an entry, the location counter will be set forward, (if required) so that its low-order position equals the low-order position of the M variant. If the high-order position of the M variant has an entry, the location counter equals the high-order position of the M variant. If both the high and low-order positions of the M variant are filled, then both the units and tens position of the location counter will be adjusted in that order(figure 7-15).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C AD TAG
01			ALC	0								
02			ALC	9								
03			ALC	90								

Figure 7-15. Adjusting the Location Counter

EQU (EQUATE).

The label in the Symbolic Label Field is assigned the same address as the entry in the A Address Field. The A Address Field entry must either be machine language, or does not exceed the present address of the location counter (except by character incrementation). Figure 7-16 illustrates the entries that equate A and B to machine address.

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDR TAG
01		A	EQU			@Z96						
02		B	EQU			-A						

Figure 7-16. Equate Statements

CST (CONSTANT).

The length of a constant is specified by an entry in the M and N Variant Field. A maximum of 60 characters of data may be entered in each constant line starting at the high-order position of the A Address Field (figure 7-17). Constants longer than 60 characters will use the first 60 characters of the entry, and the balance of the field will be filled with blanks.

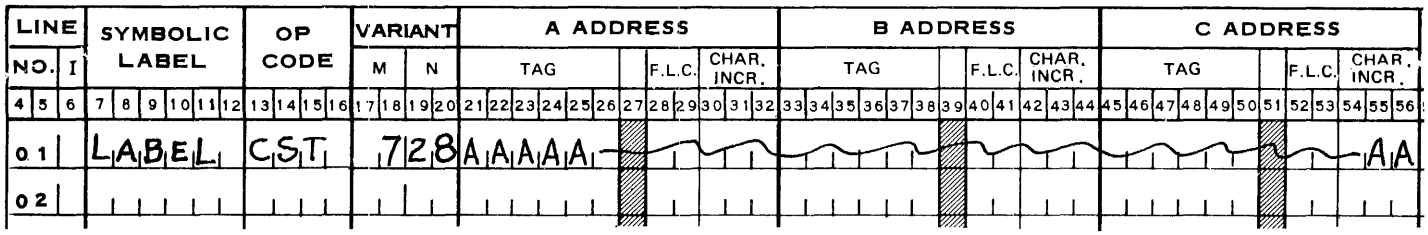


Figure 7-17. A Constant with 60 Data Characters and 668 Blanks

RSV (RESERVE MEMORY).

This entry is similar to CST, except that the entire area is set to blanks (figure 7-18). Any part of the Address or Remark Field can be used for comments.

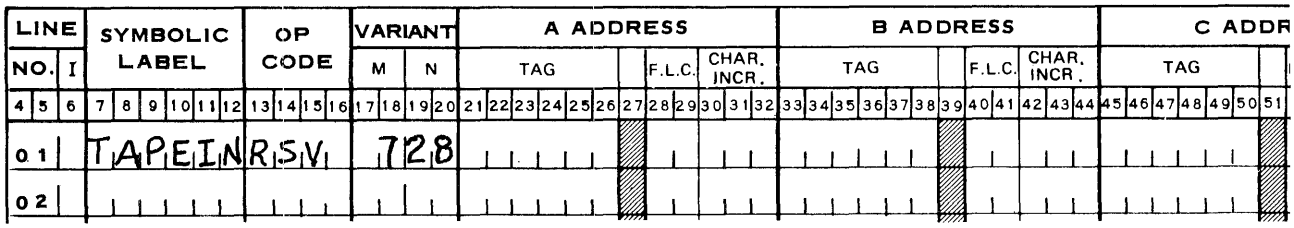


Figure 7-18. Reserving 728 Character Positions Labeled TAPEIN

HDG (HEADING).

This instruction is used to insert remarks on an assembly listing. All information contained in this record is printed. The M and N variants of this pseudo operation controls the spacing and skipping, before printing (M variant) and after printing (N variant). Error checking is not performed on the variant coding. If the M and N variants are left blank, a single space will not be executed (figure 7-19). These options provide form spacing for segmenting the program listing.

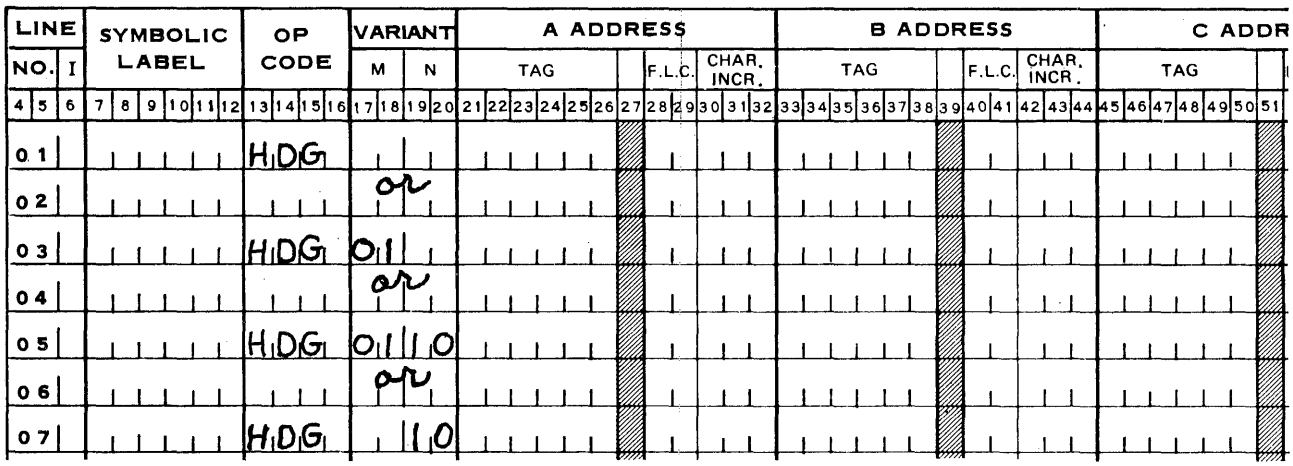


Figure 7-19. Normal Heading

OVR (OVERLAY).

This instruction defines an overlay starting with the next instruction and continuing to an OVR or END command. The next instruction will have the address specified by the A address that has the same requirements as the SLC command (figure 7-20). If this is the first OVR

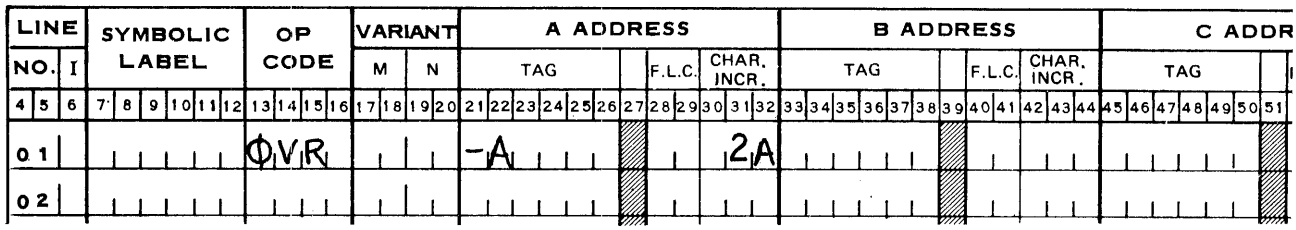


Figure 7-20. Typical Overlay Card

entry encountered in the program, the auto-load records for the object program will be produced to the end of the object memory as specified by the HEAD card (unless the location counter is set to 000). Subsequent overlays will terminate after the last character is used.

The last auto-load output of each overlay segment will contain asterisks in columns 61-63, a sequential overlay segment number in column 64-65. (01-xx), program identification in columns 72-76, and card sequence number in columns 72-80. The remaining columns will be blank.

Overlays may start or terminate at a non-module-0 address. All auto-load cards (except the last) will contain 60 characters with the applicable beginning address in columns 61-63 and word character count in columns 64-65.

An overlay entry will always start at the beginning of a new page on the program listing; therefore, an OVR will force a skip to channel one.

SAD 3 (SYMBOLIC THREE-CHARACTER ADDRESSES).
This instruction generates a three-character address constant (figure 7-21).

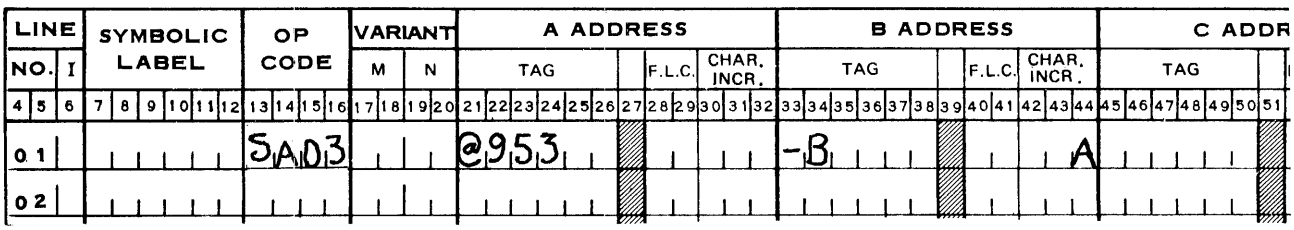


Figure 7-21. 6-Character - 2 Part Constant

GPMK (GROUP MARK).

This instruction generates a one-position constant consisting of a group mark (figure 7-22).

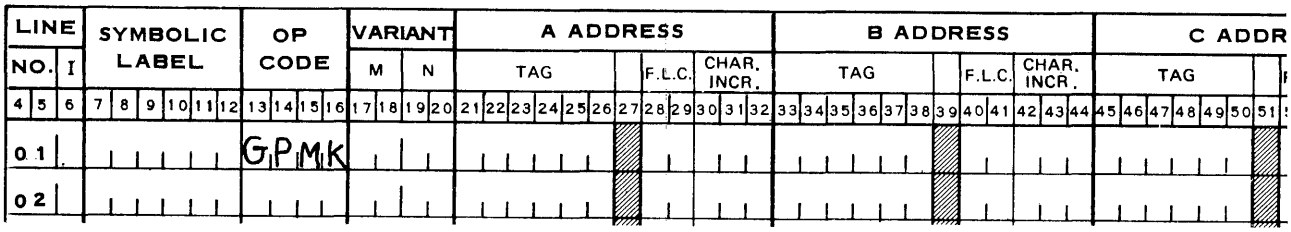


Figure 7-22. 1-Character Group Mark

TPMK (TAPE MARK).

This instruction generates a two-position constant consisting of a tape mark followed by a group mark (figure 7-23).

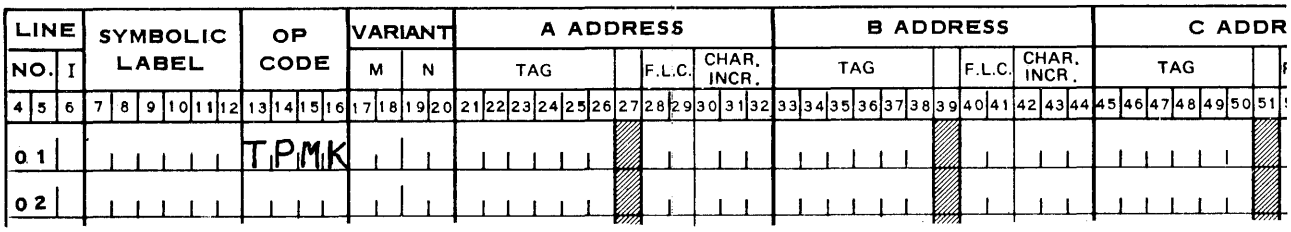


Figure 7-23. Tape Mark Coding

END (END OF PROGRAM).

This instruction must be the last program entry (figure 7-24).

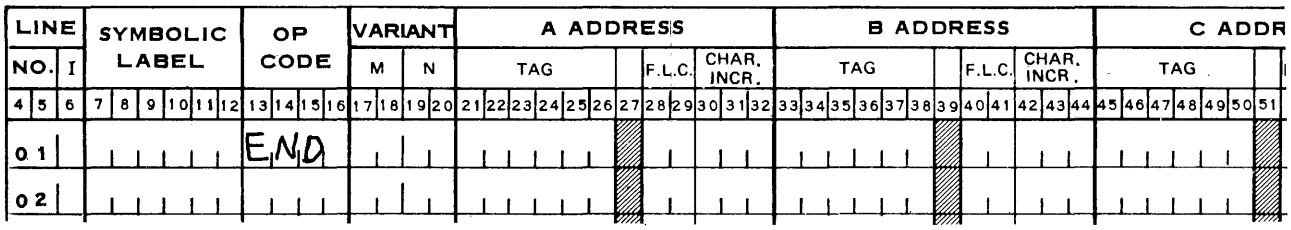


Figure 7-24. End Coding

MACRO INSTRUCTIONS (NON-MULTIPROCESSING).

Macro instructions cause the assembler to place various address parameters into the skeleton taken from the Macro Library. The skeleton

then becomes the desired subroutine and is assembled into the symbolic program. The macro instruction is included in the program by writing the name of the macro instruction in the OP Code Field, and the parameters of the macro instruction in the A, B, and C Address Fields.

The following macro instructions are treated as entries, and documented as follows:

LNK (LINK TO ROUTINE).
This entry will create a two instruction linkage to a routine. The A address is the required branch-to address. The B address is the exit location of the sub-routine. The C address is the address where control will return after the routine is completed. If the C Address Field is blank, control will return to the instruction following the LNK entry (figure 7-25).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDR TAG
01			LNK			RTN			EXIT			
02			LNK			RTN			EXIT			X

Figure 7-25. Link to a Subroutine With a Return to the Next Instruction and a Return to X

SET (SET EXIT ADDRESS).
This macro instruction is similar to LNK except that a branch is not taken. The A Address Field contains the routine exit location. The B Address Field specifies where control will return after the routine has been completed.

If the B Address Field is blank, control will return to the instruction following the SET entry (figure 7-26).

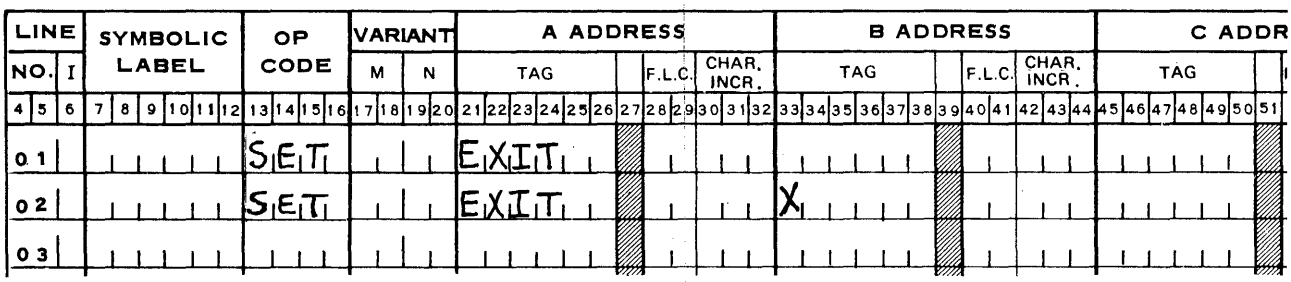


Figure 7-26. Set Exit for Return to Next Instruction, and Set Exit for Return to X

OUTPUT CAPABILITIES.

The Operating System Assembler allows the assembly process output to be either a printed copy, magnetic tape, or punched paper tape records. A multiple output media may be specified for either the program listing and/or the auto-load assembly output, but a program listing must be printed.

PROGRAM LISTING.

A portion of a program listing is illustrated in figure 7-27. Except for the following exceptions, the format is generally the same as the Basic Assemblers.

The listing is single spaced with a normal End-of-Page sensing. Page control is provided within the Header (HDG) entries. An Overlay (OVR) entry will cause an automatic skip to a new page. Auto-load output information is also shown on the program listing with a card number shown for each entry. A Reserve (RSV) entry shows only the beginning auto-load number; however, a Constant (CST) entry indicates the auto-load information up to the first sixty (60) positions of the entry.

```
IDENTIFICATION OP3T1                    OP003 TEST PROGRAM                    AS OF: 09-09-71          PAGE # 0002
ERRORS    LOC  OMNAAABBBCCC SY/LAB OP/C MMNN A ADRS FIELD B ADRS FIELD C ADRS FIELD REMARKS.............. SEQ. # CRD#

  S       ***                      OVR       @  0   TRACE ROUTINE              SET AAA TO CORE LOC USED     102

                                   HDG  2020        *** TRACE ROUTINE ***                                  103

           0  710800ENDTTO STR     TFR    12 @800        @END          OP          SET AAA TO REG ADDR & ST  104   0001
          /0  703  6   U/9         TFR     3 STR       6              EXIT       9 SET UP TRACE EXIT         105   0001
          S0  703  3   VZ@         TFR     3 STR       A              LOC          TFR LOC TO PRT            109   0001
          T0  710TT0   W 4         TFR    12 OP                       LN=OP        ALPHA TFR  OP TO PRT      110   0001

                                   HDG  2020        *** SET UP PRINT ***                                   111

          U0  703 U6WS4 X9         TFR     3 *         B P=LN         TFR        C SET UP TFR TO PRT         112   0001
          V0  832VZ@US6 W3 SET=UP  MSK   3 2 LN       01 MASK       6 NX         A SET UP TFR AAA ADRS     113   0002
          W0  703XXXX   X3 NX      TFR     3 @XXX        @X           TFR        A TFR AAA ADRS TO TFR     114   0002
          X0  710XXX    XXX TFR    TFR    12 @XXX                     @XXX         TFR WORD TO PRINT         115   0002
          Y0  J  003 W3WW1 LAST    ADM      @003        NX           A NX=LN        ADM TO NEXT ADDR          116   0002
          Z0  J  013 X9WZ# FIN     ADM      @013        TFR          C NX=LN     45 ADM BY 15 TO NEXT WORD    117   0002
         / 0  503 X9 Z9U 0         CAE     3 TFR       C FIN          C PRT=T      IF FINAL BR TO PRINT      118   0003
         //0  543 X9 Y9 W0         CAU     3 TFR       C LAST         C NX         IF NOT EQ CCC LOOP        119   0003

                                   HDG  2020        *** SET UP BRANCHES ***                                121

         /S0  703/S6V/6/T3         TFR     3 *         B TABLE        TT         A SET UP TABLE TFR          122   0003
         /T0  704XXX    SV2 TT     TFR     4 @XXX        @            TRE        N TFR TABLE ENTRY TO TFR    123   0003
         /U0  551SV2/U0SY0         C7U     1 TRE       N *            MSK          FALL THRU IF NUMERIC      124   0003
         /V0  J  004/T3T31 TLP     ADM      @004         TT           A @T31         ADM TABLE TFR             125   0004
         /W0  541SV3TT0/T0         CAU     1 TRE       3 OP           TT           IF OP=CODE = TABLE ENTRY  126   0004
         /X0  501SV4/T6ST0         CAE     1 TRE       4 TT         6 NOTT         IF M=VAR NOT USED         127   0004
         /Y0  542SV3TT0/T0         CAU     2 TRE       3 OP           TT           IF OM = TABLE ENTRY       128   0004
         /Z0  502TTOSS6SS0 L10     CAE     2 OP          TRCC       6 TRCC         IF OM = L1                129   0004
         S 0  541TTOSS8ST0         CAU     1 OP          TRCC       8 NOTT         IF OP=CODE = T=INTR       130   0005
         S/0  T89ST0    TT2        BRE   8   NOTT                     OP         N IF N=VAR B=BIT  INTR=I/O  131   0005
         SS0  703TV9L1TTT9 TRCC    TFR     3 BR=ADR    C @L1T         OP         C SET UP DCI & INTR=I/O    132   0005
         ST0  702ST6TV0SV3 NOTT    TFR     2 *         B BR=ADR       TRE        A SET UP TFR BRS           133   0005
         SU0  701SV5    SV@        TFR     1 TRE       5              TRE       11 SET TFR BRS TO OP LOC    134   0005
         SV0  709XXX    TTX TRE    TFR     9 @XXX                     OP     @X    TFR BRS TO OP             135   0006
         SW0  543TT0/V9SY0         CAU     3 OP          ILP          9 *      24 CAE FOR SPEC PRT=INT B7   136   0006
         SX0  703SX6TT0TT3         TFR     3 *         6 OP           OP         A  IF EQ  BZ BR ON SELF   137   0006
```

Figure 7-27. Operating System Assembler Program Listing

AUTO-LOAD.

Auto-Load output consists of 160 or 320 cards in the standard, module-0, five-instruction-per-card format (figure 7-28).

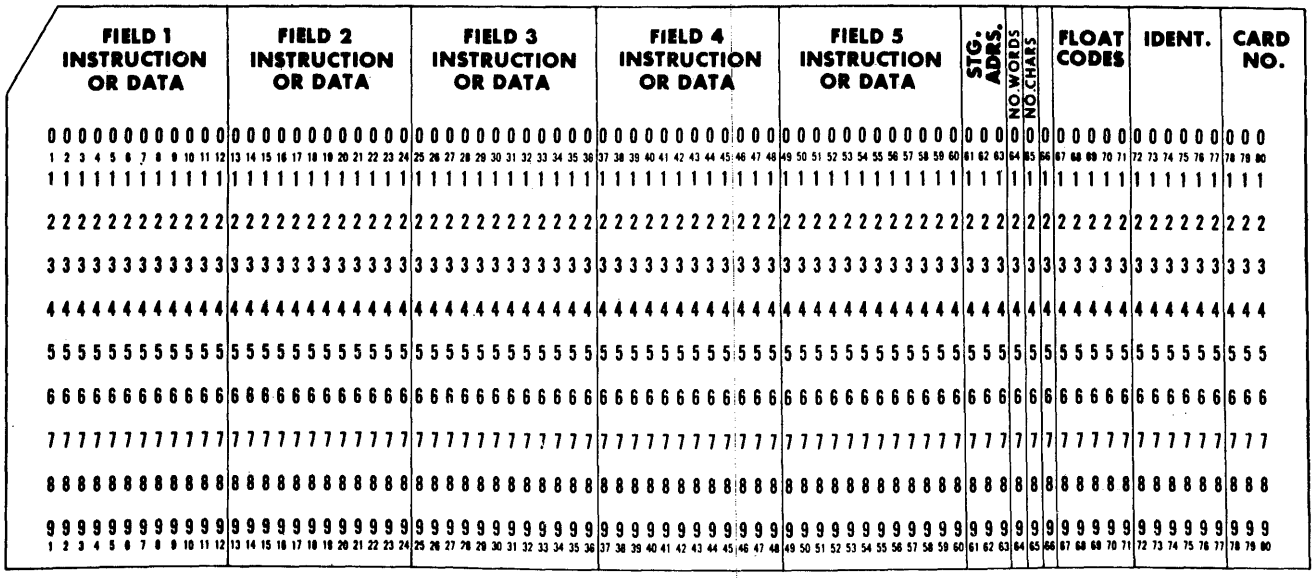


Figure 7-28. Auto-Load Program Card

Auto-load Record Contents	Card Column
Instruction or data	1-12
Instruction or data	13-24
Instruction or data	25-36
Instruction or data	37-48
Instruction or data	49-60
Storage address	61-63
Number of words	64
Number of characters	65
Relative Address Codes	67-71
Identification	72-77
Card number	78-80

Auto-load output for program overlays (except the last card) contain 60 characters of data with the applicable beginning address and word-character count. An overlay segment can begin or terminate at a non-module-0 address. The program listing includes the auto-load card number for each entry. An end of overlay segment output will be produced using the following format.

Card Column	Contains
1-60	Blanks
61-63	* * *
64-65	Sequential overlay No. (01-99)
66-71	Blank
72-76	Program identification
77-80	Card sequence number

The following is a list of multiprogramming MCP II Relative Address Codes. Columns 67-71 of the auto-load record may contain these codes.

BCL Code	Bit Configuration	Field(s) To Be Incremented
/	AB1	OP
S	AB2	A
T	AB21	OP & A
U	AB4	B
V	AB41	OP & B
W	AB42	A & B
X	AB421	OP, A & B
Y	AB8	C
Z	AB81	OP & C
,	AB82	A & C

BCL Code	Bit Configuration	Field(s) To Be Incremented
%	AB821	OP, A & C
≠	AB84	B & C
=	AB841	OP, B & C
]	AB842	A, B & C
"	AB8421	OP, A, B & C
Blank	AB	NONE

The bit codes are as follows:

1 = OP Code Fields
2 = A Address Field
4 = B Address Field
8 = C Address Field

AUTO-LOAD OUTPUT ON MAGNETIC TAPE. The output on magnetic tape consists of 80-character binary card image records.

AUTO-LOAD OUTPUT ON PAPER TAPE. The output on paper tape consists of 80-character auto-load image records.

AUTO-LOAD OUTPUT ON DISK. The disk output consists of 80-character auto-load image records blocked six records per 480-character block. The area immediately following the last specified work area is used as the output area if disk auto-load is specified.

RENUMBERED SYMBOLIC PROGRAM DECK. The ASOP Assembler provides the user with an option to request that a renumbered symbolic program deck be punched on cards.

The program output listing will reflect the new page and line number; however, if a sequence check is desired, the original input page and line number will be checked.

Method of Specification.

The ASOP Assembler requires a program HEAD Card preceding each program being assembled. (Reference the ASOP Assembler Function for Header information.)

The HEAD Card provides the assembler with the output requirements for the assembly process, specifies the work storage units, defines the object system, and indicates the media for the input source program. The HEAD Card is edited prior to the assembly process.

SECTION 8
MCP II ASOP MACRO INSTRUCTIONS

GENERAL.

When the multiprogramming capabilities of MCP II are utilized, the programmer does not communicate with the B 500 to perform I/O operations, but rather through MCP II.

MACRO INSTRUCTIONS.

The Operating System Advanced Assembler allows the use of 79 multiprogramming macros. The Macro Library containing the multiprogramming macros must be created under control of MCP II and the CMLT Function.

The following is a list of multiprogramming macros that are described in the following pages of this section.

Macro	Description
ACPT	Accept input from SPO.
BEGN	Begin run.
CLOS	File close.
DISP	Display output on SPO.
FILE	File descriptor.
LDRO	Load relative overlay.
M/PI	Multiprogramming interrupt.
OPEN	File open.
PF/C	Programmatic function call.
POSN	Position magnetic tape or line printer.
READ	Read an input record.
RECD	Record description.
ROVR	Relative overlay.

Macro	Description
STOP	Stop run.
WRIT	Write an output record.
ZIP	Stop program in process and call another program.

The following routines are supplied with the Macro Library and used by certain multiprogramming macros as automatic object code inserts, whenever an applicable macro is called by the source program.

Routine	Description
CLOO	Write standard EOF record and close file.
PERR	Macro error routine.
RE<D	Disk file input blocked is less than 133 characters.
RE<E	Disk file output blocked is less than 133 characters.
RE<I	Input blocked records are less than 133 characters.
RE<O	Output blocked records are less than 133 characters.
RE>D	Disk input blocked records are greater than 132 characters.
RE>E	Disk output blocked records are greater than 132 characters.
RE>I	Input blocked records are greater than 132 characters.
RE>O	Output blocked records are greater than 132 characters.
REUD	Disk file input unit records.
REUE	Disk file output unit records.
REUI	Input for unit records.
REUO	Output for unit records.
REUP	Output for line printer records.
WRIP	Write or position line printer file.

LINKING OF MACRO ROUTINES.

Macro routines are linked to the object program in the following manner.

a. The next input record is saved.

b. Detail macro records from the Macro Library tape are read.

c. Parameters are assigned to all equal addresses contained within the macro routine. The equal addresses must be within the following specifications.

Field	Length	Type
SY LABEL	6 POS	ANY
M VAR	2 POS	NOT CST/RSV
M VAR	4 POS	ONLY CST/RSV
N VAR	2 POS	NOT CST/RSV
A ADRS	12 POS	NOT RSV
A ADRS-FLC	2 POS	NOT CST/RSV
A ADRS-CI	3 POS	NOT CST/RSV
B ADRS	12 POS	NOT CST/RSV

Remainder of B ADRS field same as A, and C ADRS field same as B.

d. If the 12-position A, B, or C Address Fields reference parameter N (and parameter N is blank), parameter M, if coded within the field immediately following parameter N, will be inserted to replace parameter N.

e. Supplies each symbolic record to the assembler following parameter assignments.

f. END operation code is replaced with the HDG operation code.

g. All switches are reset and the next symbolic record is restored to the input area.

MACRO DEFINITIONS.

The following macro formats define the syntax to be used in the construction of the multiprogramming macros. Included with each description is a detailed listing of the symbolic code that is emitted from the Macro Library at the time of assembly.

AXCE (ACCEPT SPO MESSAGE).

This function is used in conjunction with the ACPT Macro. In order to pass the necessary information to the program, the operator will use the following format.

```
≥ ≥ ≥ A X C E i d e n t m e s s a g e
1 - - - - - 7 8 - - -12 13- - - - - 80
```

where:

a. ident - program identification.
b. message - message to be passed to the specified program.

A Display Macro (DISP) is usually implemented prior to ACPT so that the operator is aware of what programs are waiting for a message entry through the SPO. Three programs can be in operation, and all independently waiting for a different reply. Considering this possibility, the programmer should include the program identification with the display message to inform the operator which program needs action.

ACPT (ACCEPT).

The purpose of this macro is to obtain input data from the supervisory printer. The symbolic label to be assigned to the accepted message is entered in the A Address Field, and the B Address contains the exit address. If the B Address is blank, the next address in sequence will be inserted. The C Address is always blank (figure 8-1).

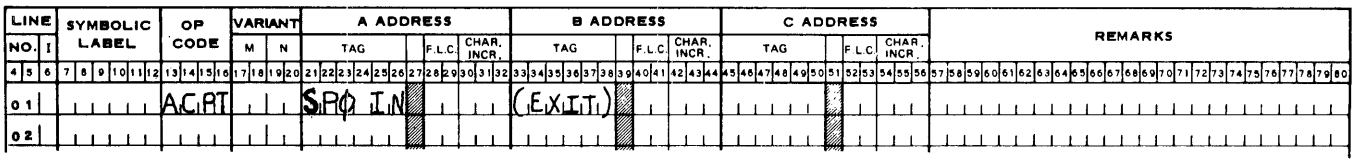


Figure 8-1. ACPT Supervisory Printer Message

Figure 8-2 illustrates an example of symbolic code emitted from the macro library at the time of assembly.

```
                        ACCEPT MACRO

        HDG 2020              ACCEPT SUPERVISORY PRINTER MESSAGE
        TFR     5×BEGLB       4              *          29SET UP PROGRAM I D
        SPO       *           1M                          PRINT ACCEPT MESSAGE
        CST    12+AX: IDENT←                              MESSAGE CONSTANT
./      TFR     5×ADRTB       B*             ×ADRTB       ALINK TO M/P
        TCB     2-/           B×INTER        ×ADRTB       C       CONTROLLER
        TCB    42@366          %2%0          *          10STORE MESSAGE
.×      EQU       *
%1      CST    48                                        MESSAGE STORAGE
./      EQU       *
        END 2020                                             *** END OF MACRO
```

Figure 8-2. ACPT Listing

BEGN (BEGUN RUN).

The purpose of this two-card macro is to set up the I/O configuration needed to run a particular program, and to remove the required I/O units from the Systems I/O Table. This macro also sets up the necessary date linkage. The A Address contains the beginning execution address entry.

The B Address specifies the I/O configuration needed to execute the object program, and the date codes are entered in the C Address Field. This macro is used whenever an object program run in a multiprogramming mode is required.

The entry codes for I/O configuration are as follows:

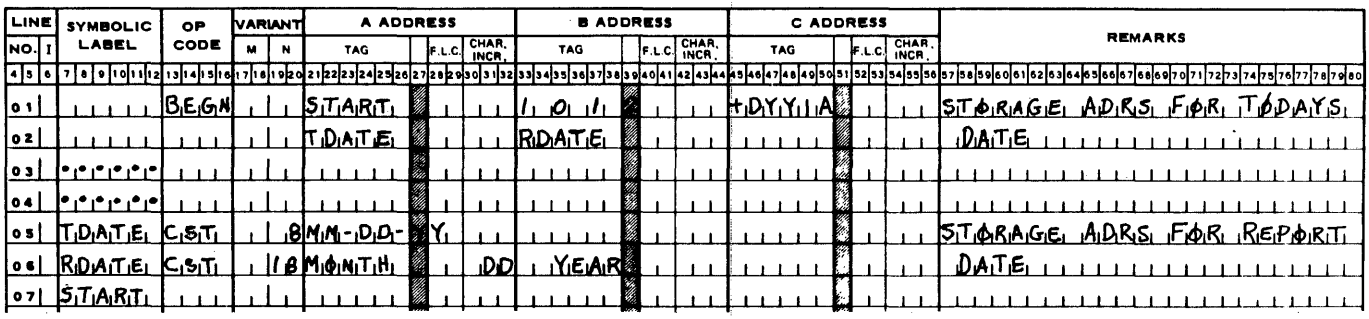
Column (B Address)	Contents
33	Input reader 1. 1 - card reader required A - paper tape reader required. Blank - none required.
34	Input reader 2. 2 - card reader required. B - paper tape reader required. Blank - none required.
35	Output punch. 0 - card punch required. + - paper tape punch required. Blank - none required.
36	Blank - Reserved for the system.
37	Line printer 1. 1 - line printer only. T - line printer or printer tape. Blank - none required.
38	Line printer 2. 2 - line printer only. T - line printer or printer tape. Blank - none required.
39	Magnetic tape. 1 thru 6 - number of units required.

The coding for the dates is as follows:

Column (C Address)	Contents
45	Must contain a + (plus zero).
46	D if either date is required.
47	Y if today's date is required.
48	Y if report date is required.
49	Today's date format. A - alpha month, day, year. B - day, alpha month, year. J - abbreviated alpha month, day, year. K - day, abbreviated alpha month, year. 1 - numeric month, day, year. 2 - numeric day, month, year. 3 - Julian YYDDD. 4 - Julian DDDYY.
50	Report date format - same as the above Today's date.

Data pertaining to the BEGN Macro is entered as follows:

The A Address must contain a symbolic address for today's-date, B Address contains the symbolic address for the Report-date, and the program blocks (to load and total) are entered in the C Address Field (figure 8-3).



LINE NO.	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01		BEGN			STARRT			101			+DYYYYA			STORAGE ADRS FOR TODAYS
02					TDATE			RDATE						DATE
03														
04														
05	TDATE	CST		8	MM-DD-	Y								STORAGE ADRS FOR REPORT
06	RDATE	CST		18	MONTH		DD	YEAR						DATE
07	STARRT													

Figure 8-3. BEGN I/O Macro with Data Linkage

Figure 8-4 illustrates an example of BEGN symbolic code from the Macro Library at the time of assembly.

```
                    BEGIN RUN MACRO

      HDG 2020              PROGRAM ADD RECORD
      HDG 20                BEGIN RUN MACRO
      OVR     @750                                  BEGIN RUN MACRO
      CST    7≥≥≥PADD
      CST    5%7                                   PROGRAM IDENTITY
      CST    2RN                                   TYPE & BEG HALT CODES
      CST   16%2                                   I/O CONFIGURATION
      CST    6%6
      CST   24  PROGRAM ADD CARD
      HDG 2020              PROGRAM LABEL
×BEGLBCST    4+BEG                                 LABEL
      CST    5%7                                   PROGRAM IDENTITY
      SAD3    %1                                   BEGINNING EXECUTION ADRS
      HDG 2020              DATE CONSTANT
      CST    6%3                                   DATE CONSTANT
      SAD3    %4           %5                      DATE STORAGE ADDRESSES
      HDG 2020       EQUATES FOR ASSEMBLER MACROS
×INTEREQU     @030
×ADRTBEQU     @040
×EXI/OEQU     @450
×IOSEGEQU     @366
×ERROREQU     @600
×STDEREQU     @700
×N EOFEQU     @780
×OVRWAEQU     @270
×SYIOTEQU     @290
×OVRNMEQU     @019
×OVRCLEQU     @330
×OVRTSEQU     @363
×BGCSTEQU     @304
×ROLDREQU     @670
×P F/CEQU     @020
×F/CAREQU     @356
×PRLABEQU     @804
02FDKEEQU     @100
SW SPOEQU     @260
RETADREQU     @399
RETLNKEQU     @540
      HDG   10**************************************************************
      END 2020                                    *** END OF MACRO ***
```

Figure 8-4. BEGN Listing

CLOS (FILE CLOSE).

This macro is used to close a file. It writes the last block, tape mark, and rewinds the tape. This macro will return the I/O unit(s) to the Program I/O Table. The A Address Field must contain the file-name of the file to be closed. The B and C Address Fields are blank (figure 8-5).

These parameters are converted to the following:

a. %1 - file-name.

b. %2 - file-name with CI of 2B.

c. %3 - file-name with CI of 10.

d. %4 - file-name with CI of 2A.

e. %5 - file-name with CI of 60.

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01		EØF1	CLØS			TP-ØUT									CLØSE TAPE ØUT FILE
02															
03															
04															
05															
06															
07															

Figure 8-5. CLOS and I/O Unit Placement to Program I/O Table

Figure 8-6 illustrates an example of CLOS symbolic code emitted from the Macro Library at the time of assembly.

```
                    FILE CLOSE MACRO

HDG  2010                FILE CLOSE MACRO
HDG    10################################################################
TFR     3*                B@CLS          ×OVRNM       LINK TO EXEC CONTROLLER
TFR     2*                B@09           ×OVRWA
TCB    12*               10×OVRCL        ×OVRTS          TO CALL FILE
SAD3    *                10%2                               CLOSE
CST     6%1                                                  ROUTINE.
HDG    10****************************************************************
END    20                                                *** END OF MACRO ***
```

Figure 8-6. CLOS Listing

DISP (DISPLAY).

This macro provides output data to the SPO. The A Address entry contains the address of the message to be displayed. The B Address and C Address Fields are blank (figure 8-7).

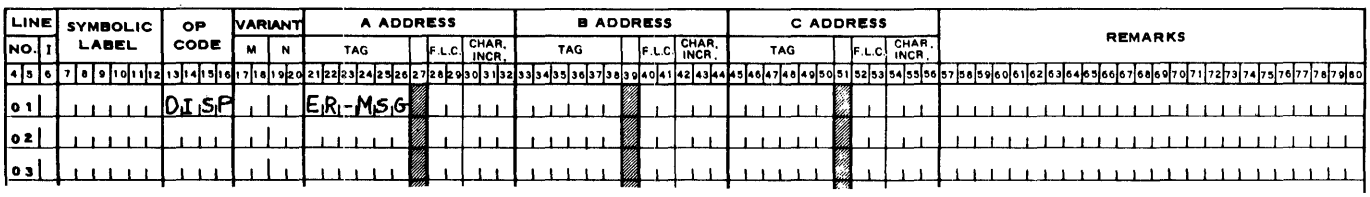


Figure 8-7. Output Data for SPO

Figure 8-8 illustrates an example of DISP symbolic code emitted from the Macro Library at the time of assembly.

```
                        ***  BEGIN DISPLAY MACRO  ***

HDG  1  1
HDG     10##################################################################
TCB      5xPRLAB        *            20*          17STORE PROGRAM IDENT
CST     11DISP IDENT,
GPMK
SPO       =/                                      PROGRAM IDENT
SPO       %1                                      PROGRAM INFORMATION
HDG     10**********************************************************************
END     20                                        **  END DISPLAY MACRO **
```

Figure 8-8. DISP Listing

FILE (FILE DESCRIPTOR).

The macro provides the I/O codes for the files used by the program. The file name is entered in the A Address Field (six positions) and the B Address contains the I/O codes. The C Address Field is blank (figure 8-9).

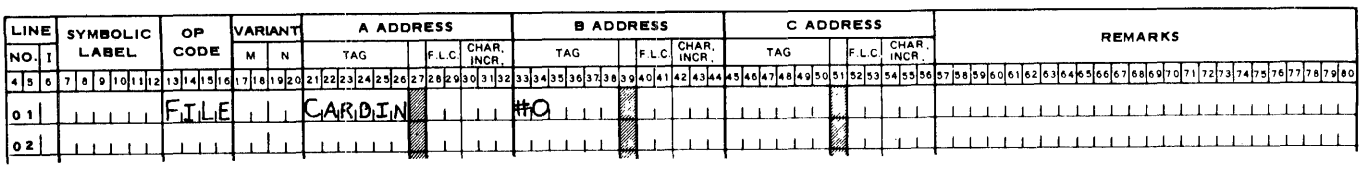


Figure 8-9. File Code Entry

The I/O codes for the B Address Field are as follows:

Code	Information
#0	Card reader input file.
E2	Paper tape output file.
F1	Paper tape input file.
F2	Paper tape input file (direct).
D1	Magnetic tape input.
K2	Disk input file.
@0	Card punch output file (BCL).
@1	Card punch output file (BULL).
@2	Card punch output file (ICT).
A0	Printer file (120 positions).
A+	Printer file (132 positions).
D2	Magnetic tape output file.
K0	Disk output file.
P0	Printer tape output file (120 positions).
P+	Printer tape output file (132 positions).

Figure 8-10 illustrates an example of FILE symbolic code emitted from the Macro Library at the time of assembly.

```
                FILE DESCRIPTOR MACRO

HDG 2010              FILE DESCRIPTOR MACRO
HDG   10##############################################################
CST    1+
CST    2%2                                        I/O TYPE
CST    6%1                                        FILE NAME
SAD3    %1                                        FILE NAME
HDG   10**************************************************************
END   20                                            *** END OF MACRO ***
```

Figure 8-10. FILE Listing

LDRO (LOAD RELATIVE OVERLAY).

This macro calls a multiprogramming routine to load a relative overlay into an available area. The A Address Field must contain the overlay name. The address of the first instruction to be executed in the overlay must be inserted in the B Address. If the B Address Field is blank, the first address of the overlay is inserted. The C Address is blank (figure 8-11).

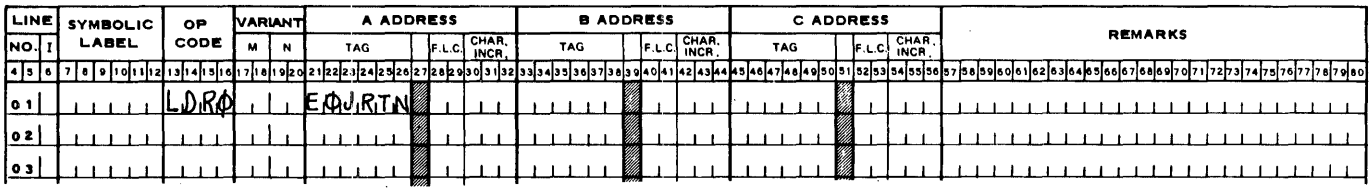


Figure 8-11. LDRO Load

Figure 8-12 illustrates an example of LDRO symbolic code emitted from the Macro Library at the time of assembly.

```
                    LOAD RELATIVE OVERLAY MACRO

HDG  2010          LOAD RELATIVE OVERLAY MACRO
HDG    10##########################################################################
TFR     2×BEGLB                          ×BGCST       STORE ADM COUNTER
TFR     3*              B%1              ×BGCST      NSTORE OVERLAY ADDRESS
TFR     2*              B%2%1            ×ADRTB      ASET EXIT ADRS
TFR     6*             1B                ×OVRTS       LINK TO M/P CONTROLLER
$AD3     @61             ×ROLDR          %1                 TO CALL OVERLAY
CST     3%1
HDG    10**********************************************************************
END    20                                                *** END OF MACRO ***
```

Figure 8-12. LDRO Listing

M/PI (MULTIPROGRAMMING INTERRUPT).

The purpose of this macro is to transfer control from one program to another. M/PI is not required when I/O Macros are used. If there is little I/O activity in the object program, the Multiprogramming Interrupt Macro should be used to accomplish transfer control. The A, B, and C Address Fields are always blank (figure 8-13).

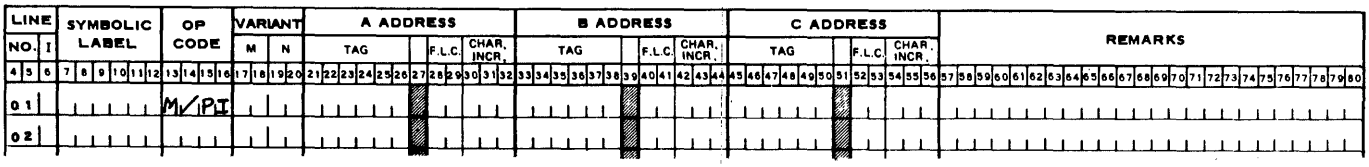


Figure 8-13. Control Transfer

Figure 8-14 illustrates an example of M/PI symbolic code emitted from the Macro Library at the time of assembly.

```
                    MULTI-PROCESSING INTERRUPT MACRO

HDG  2010                MULTI-PROCESSING INTERRUPT MACRO
HDG    10################################################################
TFR     5xADRTB      B%1%0          xADRTB      ASHIFT TABLE OF RETURN
TCB     2-+          BxINTER        xADRTB      C  ADDRESSES AND ADD THIS
EQU      *                                       ONE TO THE TABLE.
HDG    10****************************************************************
END    20                                          *** END OF MACRO ***
```

Figure 8-14. M/PI Listing

OPEN (OPEN).

The purpose of this macro is to open a file by reading the first available record. The A Address contains the file-name, B Address the beginning disk address, and C Address contains the ending disk address of the file (figure 8-15). The B and C Address Fields are blank if a file other than disk is required.

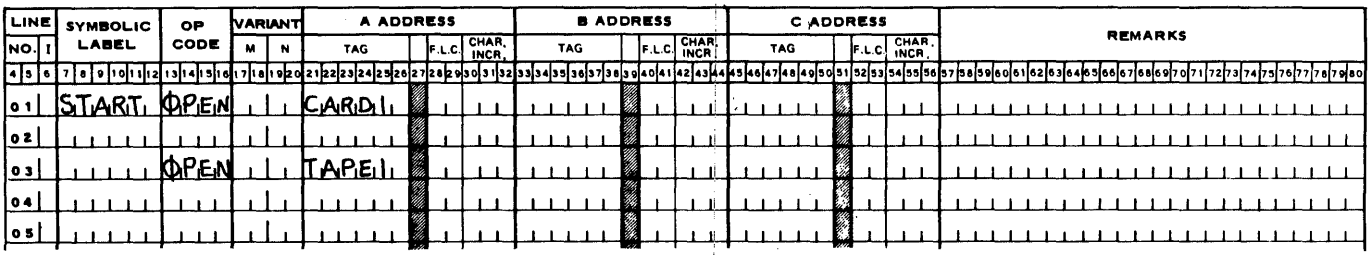


Figure 8-15. File OPEN with the Reading of First Record

The parameters are converted to:

a. %1 - file-name.
b. %2 - BEG disk file ADRS.
c. %3 - END disk file ADRS
d. %4 - file-name with CI of 2B.

Figure 8-16 illustrates an example of OPEN symbolic code emitted from the Macro Library at the time of assembly.

```
                    FILE OPEN MACRO

HDG  2010                FILE OPEN MACRO
HDG    10################################################################
TFR     3*              B@OPN          ×OVRNM       LINK TO EXEC CONTROLLER
TFR     2*              B@08           ×OVRWA
TCB    27*             10×OVRCL        ×OVRTS          TO CALL FILE
SAD3    *              30%4                               OPEN ROUTINE
CST     6%1
GPMK
CST     7%2
CST    13%3
ALC   0
HDG    10*****************************************************************
END    20                                                *** END OF MACRO ***
```

Figure 8-16. OPEN Listing

PF/C (PROGRAMMATIC FUNCTION CALL).
This two-card macro sets up the linkage to call a function of MCP II for execution. The A Address contains the first 12 parameter positions, B Address the second 12 parameter positions, and the third 12 parameter positions are entered in the C Address Field (figure 8-17).

LINE NO.	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01		P,F/C			PRDK1			0001100	00	00	029992	40		CARD1
02														CARD2
03														
04														

Figure 8-17. Operating System Function to Print Disk from Segment 1,000 thru 2,999

If a second card is required, the A Address Field contains the last 11 parameter positions, and the B and C Addresses are blank.

Figure 8-18 illustrates an example of PF/C symbolic code emitted from the Macro Library at the time of assembly.

```
                    PROGRAMMATIC FUNCTION CALL MACRO

HDG 2010          PROGRAMMATIC FUNCTION CALL MACRO
HDG   10##################################################################
TFR    2*           B@04          @113
TFR    3*           B*          70×ADRTB      ASET EXIT ADDRESS
TCB   54*          10×P F/C       ×F/CAR       MOVE F/C & PARAMETERS
CST    3≥≥≥
CST    4%1                                     NAME
CST   12%2                                     PARAMETERS
CST   12%3
CST   12%4
CST   11%5
TFR    2*           B@630         @113
HDG   10******************************************************************
END   20                                              *** END OF MACRO ***
```

Figure 8-18. PF/C Listing

POSN (POSITION).
This macro is used to position the appropriate I/O unit. The A Address contains the file name of the unit to be positioned, B Address the type of positioning required, and the C Address Field is blank (figure 8-19).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01			POSN			TP OUT			R						REWIND TAPE OUT AT EOF
02															
03															
04															

Figure 8-19. Appropriate I/O Positioning

Acceptable codes for positioning are as follows:

a. R - rewind tape.
b. B - backspace tape.
c. SS - single space printer.
d. DS - double space printer.
e. CN - skip to channel N after printing, where N is 1-@.

Figure 8-20 illustrates an example of POSN symbolic code emitted from the Macro Library at the time of assembly.

POSITION MAGNETIC TAPE FILE MACRO

```
HDG 2010         POSITION MAGNETIC TAPE FILE MACRO
HDG   10##################################################################
TFR   36%8                          *IOSEG       MOVE I/O SEGMENT
TFR    1*           B%6             *IOSEG     1TMODIFY M VARIANT
TFR    6*          1B               *IOSEG     30MODIFY RET & RETRY ADRS
BRU     *EXI/O       -/              *          10GO TO EXECUTE
HDG   10******************************************************************
END   20                                                *** END OF MACRO ***
```

Figure 8-20. POSN Listing

These parameters are converted to:

a. %1 - file-name.
b. %2 - not used.
c. %3 - end-of-output address.
d. %4 - file-name with CI of 2A.
e. %5 - space/skip variants.
f. %6 - position OP Code or variant.
g. %7 - file-name with CI of 60.
h. %8 - file-name with CI of 2B.

READ (READ).

The purpose of this macro is to make the next record available for processing. The A Address contains the file-name, B Address the error address (if B Address is blank a standard address is assumed), and the End-of-Input return address is entered in the C Address Field (figure 8-21).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01		IN	READ			CARD11						EOF			
02		TAPE11	READ			TAPEIN						EOT			
03															
04															

Figure 8-21. Read Next Record Available

The parameters are converted to:

a. %1 - file-name.
b. %2 - error-address. If blank, standard is inserted.
c. %3 - end-of-input address.
d. %4 - file-name with CI of 2A.

Figure 8-22 illustrates an example of READ symbolic code emitted from the Macro Library at the time of assembly.

```
                READ MACRO

HDG 2010                READ MACRO
HDG   10################################################################
TCB    8*          1A%1            %4          LNK TO RECORD ROUTINE
NOP    *        @2 10%2            %3            TO GET NEXT RECORD
HDG   10****************************************************************
END   20                                              *** END OF MACRO ***
```

Figure 8-22. Read Listing

RECD (RECORD DESCRIPTOR).
The purpose of this two-card macro is to describe the records to be processed. It works in conjunction with the File Descriptor Macro to select the proper I/O routines, depending on whether the records are blocked or unblocked.

The A Address on the first card contains the record name, B Address the record length (4 positions), and C Address the records per block (3 positions).

The A Address of the second card contains the I/O code to be used by the file. If the records are disk, the segment size is entered in the B Address and the segments per block in the C Address Field. The acceptable codes are 01-10 (figure 8-23).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01		CARD1	RECD			CARD			0080			001			
02						#0									
03															
04															

Figure 8-23. Record Description and Selection of I/O Routine

The parameters are converted to:

a. %1 - record name.

b. %2 - record length (also used as M VAR/Record length).

c. %3 - block length (RECD length x RECDS/BLK).

d. %4 - block length plus excess disk area (excess DSK Area = SEG/BLK x SEG SZE).

e. %5 - number of characters for last TFR (RECD SZE divided by 120. Zero remainder set to 120) four positions.

f. %6 - record size less character size of last TFR.

g. %7 - number of characters for last ADM (preceded by #) four positions total.

h. %8 - number of segments per block.

i. %9 - last two positions of record length if unit records or last two positions of last TFR are blocked records.

This macro will automatically insert the proper routines to effectively handle the blocking and unblocking of records. The routines are provided as part of the Advanced Assembler II Macros.

ROVR (RELATIVE OVERLAY).
This macro assigns codes to the auto-load output listing (columns 67-71). The codes are used to relocate the program at object time. The A Address contains the overlay name (three significant positions), B Address the number of program blocks in the overlay, and the C Address Field is blank (figure 8-24).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	N	A ADDRESS TAG	F.L.C.	CHAR. INCR.	B ADDRESS TAG	F.L.C.	CHAR. INCR.	C ADDRESS TAG	F.L.C.	CHAR. INCR.	REMARKS
01			ROVR			IDEN			XX						SET OVERLAY FOR REL ADR
02															
03															
04															
05			LOVR			IDEN			BRADDR						CALL OVERLAY
06															
07			THE	ABOVE	INSTRUCTION	WILL	LOAD	THE	OVERLAY	#IDEN#					
08			AND	BRANCH	TO	LOCATION	BRADDR.	IF	THE	B	FIELD	IS			
09			BLANK,	A	BRANCH	TO	THE	OVERLAY	WILL	OCCUR.					
10															
11															

Figure 8-24. Auto-Load Output Codes

The parameters are converted to:

a. %1 - overlay name.
b. %2 - number of blocks.
c. %3 - overlay base address.

Figure 8-25 illustrates an example of ROVR symbolic code from the Macro Library at the time of assembly.

```
                    RELATIVE OVERLAY MACRO

OVR      %3
HDG 2010              RELATIVE OVERLAY MACRO
HDG   10##############################################################
CST    6+≥OVR≥                                        OVERLAY
CST    4%1
CST    2%2                                                  LABEL
HDG   10**************************************************************
END   20                                                *** END OF MACRO ***
```

Figure 8-25. ROVR Listing

STOP (END RUN).

This macro ends processing, removes a program from the programs in the Program Table, returns the I/O units to the Systems I/O Table, and updates the Available Memory Table. Control is returned to the Executive Routine, and the A, B, and C Address Fields are blank (figure 8-26).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01			STOP												
02															
03															

Figure 8-26. Termination of Processing

Figure 8-27 illustrates an example of STOP symbolic code emitted from the Macro Library at the time of assembly.

```
                    STOP RUN MACRO

HDG 2010                STOP MACRO
HDG   10##############################################################
TFR    3*             B@EOP          ×OVRNM       LINK TO EXEC CONTROLLER
TFR    2*             B@15           ×OVRWA         TO CALL THE END OF
TCB   24×BEGLB         ×OVRCL         ×OVRTS      C    PROGRAM ROUTINE
HDG   10**************************************************************
END   20                                                *** END OF MACRO ***
```

Figure 8-27. Stop Run Macro

WRIT (WRITE).

The purpose of this macro is to provide a means of moving records to an output area. If blocked records are written, an appropriate blocking sub-routine is automatically provided, but if unblocked records are used they will be written on the appropriate output device. This is a two-card macro (figure 8-28). The first card contains the file name in the A Address Field, B Address the error address (a standard address is assumed if this field is blank), and the C Address contains the End-of-File Branch Address entry (a standard address is assumed if this field is blank).

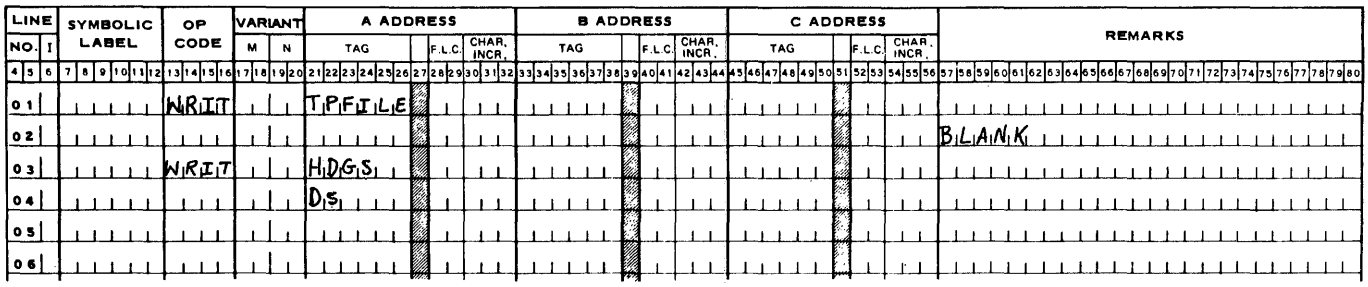


Figure 8-28. Heading Write Followed by Double Space on Line Printer

A second card contains the printer spacing (printer files) in the A Address Field. The B and C Address Fields are blank.

Acceptable codes are:

a. SS - single space after printing.
b. DS - double space after printing.
c. NS - no space after printing.
d. CN - skip to channel N after printing, where N is 1-@.

The following parameters are converted to:

a. %1 - file-name.
b. %2 - error ADRS. If blank standard is used.
c. %3 - end of output. If blank standard is used.

d. %4 - file-name with CI of 2A.

e. %5 - space/skip variants.

f. %6 - printer of code-A.

g. %7 - file-name with CI of 60.

Figures 8-29 and 8-30 illustrate examples of the WRIT symbolic code emitted from the Macro Library at the time of assembly.

```
                    WRITE OR POSITION PRINTER FILE MACRO

HDG 2010        WRITE OR POSITION PRINTER FILE MACRO
HDG   10########################################################################
TFR    3*           20              %7          SET PRINT OR SPACE OPCDE
TCB    8*           1A%1            %4          LINK TO RECORD ROUTINE
CST    1%6                                        TO WRITE OR
CST    2%5                                          POSITION
SAD3    *           10%2            %3             PRINTER FILE
HDG   10************************************************************************
END   20                                          *** END OF MACRO ***
```

Figure 8-29. Write or Position Printer File Macro Listing

```
                    WRITE MACRO

HDG 2010              WRITE MACRO
HDG   10################################################################################
TCB    8*           1A%1            %4          LINK TO RECORD ROUTINE
NOP     *           10%2            %3            TO WRITE RECORD
HDG   10********************************************************************************
END   20                                          *** END OF MACRO ***
```

Figure 8-30. Write Macro Listing

ZIP (STOP ONE PROGRAM - START ANOTHER).

The purpose of this macro is to provide the capability to programmatically stop and/or start a program. The A Address entry for the End Run/Zip to next program card, contains the identity of the program to be called (columns 21 thru 25). The B and C Address Fields are blank (figure 8-31).

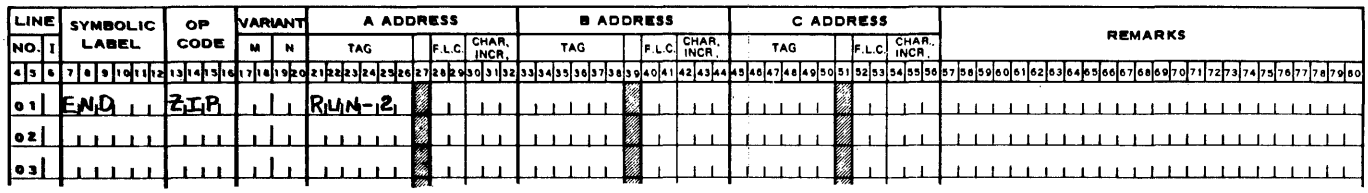


Figure 8-31. Program/Start Zip Stop

Figure 8-32 illustrates an example of ZIP symbolic code emitted from the Macro Library at the time of assembly.

```
                    ZIP MACRO -- STOP & CALL ANOTHER PROGRAM

HOG 2020              ZIP MACRO -- STOP & CALL ANOTHER PROGRAM
TFR    13*          40              *F/CAR       SET ZIP CALL
TFR     3*           B@EOP          *OVRNM       LINK TO EXEC CONTROLLER
TFR     2*           B@15           *OVRWA         TO CALL THE END OF
TCB    24*BEGLB       *OVRCL        *OVRTS     C     PROGRAM ROUTINE
CST     7ZIPPADD                                  ZIP CALL CONSTANT
CST     5%1                                       ZIP PROGRAM IDENTITY
GPMK
CST    11                                         TO EVEN OUT MACRO
EQU     *                                         SET FOR FALL THRU ADRS
END 2020                                            *** END OF MACRO ***
```

Figure 8-32. ZIP Listing

The program points within a macro entry do not interfere with the program points used by the programmer; because program points are always local to the library level in which they occur.

A macro instruction can refer to other macro instructions (e.g., a macro instruction inside a macro instruction, inside a macro instruction, within the main program) and also the CALL pseudo entry. The only pseudo operation not allowed inside a macro routine is OVR. A programmer need not be concerned about the length of included macro instructions for entry increment purposes, or if any of the included sub-routines contain program points.

The calling strings of the included macro routine may contain references to the calling string of the next higher level macro instruction (i.e., percent references are permissible in the calling string of a macro routine).

LIBRARY ROUTINES (MACRO AND CALL ROUTINES).

Library routines are written the same as other sections of coding, except it allows program control to return to the parameters of the calling string (line or lines of code that call the macro). The additional type of addressing (called Percent Addressing) is provided for this purpose.

Percent Addressing is coded as % followed by a number from 0 through 9. This coding corresponds to the parameters reflected in the A, B, and C Address Fields of the calling string. For example, the unit number in the Macro Tape Read entry would be referred to as %3. A %0 entry refers to the address for the line of code following the macro entry. Percent Addresses may be included in any or all of the following positions:

a. High-order position of the symbolic label field. Percent Addressing will use the first six characters from the calling string and insert them as a symbolic label.

b. M or N Variants. In either position, Percent Addressing will take the first two characters from the calling string and insert them into the variant field of the symbolic command.

c. High-order position of an address field. The full 12-character symbolic address from the calling string will be inserted into the address field.

d. Entry increment field (positions directly following the sign position). The first two characters from the calling string will replace the percent reference.

A second or third percent address may be written directly following the first. In essence this construct says that if the left-most Percent Address Field is blank use the one to the immediate right.

Macro routines are to be used by the programmer as though they were a single special purpose instruction. The number of lines to be

inserted by the macro routine counts for only as many entries as it takes the programmer to declare it (usually one). An example of this is the READ Macro. The programmer may consider this as a special command with three address fields. The fact that it provides for error checking in addition to reading a tape is unimportant (as far as coding is concerned). The programmer can treat it as one line of entry, or just as he would an actual machine language instruction.

LIBRARY MACRO REQUIREMENTS. The coding of macro routines is the same as regular coding with the following differences:

a. Absolute symbolic labels (as opposed to program points) should never be used, because it causes duplicate labels whenever a specific macro is used more than once in a program.

b. When writing a macro routine, it must be possible to make use of the parameters furnished by the programmer when calling for the routine.

In order to accomplish item b, an address composed of a % followed by a number from 0-9 is used. %1 through %9 refers to the nine possible parameters that may be used by the person requesting a macro insertion. The %0 entry is the address assigned to the entry directly following the macro routine in the calling program. The % references may be used in many places within a macro, and in all cases it will result in the appropriate number of characters to replace the % reference. The locations where these references may appear are as follows:

a. Symbolic label - the first six characters from the appropriate parameter field becomes the symbolic address.

b. M and N variants - the first two characters of the appropriate parameter are used.

c. A, B, and C Addresses - the complete 12-character parameter is used.

NOTE

Refer to the Advanced Assembler II Reference Manual (1042769) for additional information concerning the CALL or Macro routines.

SECTION 9
B 500 COBOL COMPILER

GENERAL.

MCP II has the capability of storing the COBOL Compiler on disk and making it callable as a MCP II Function.

MCP II also contains the COBOL Compiler maintenance and start-up programs that are available to the user as function calls.

By coding OP-SYSTEM under the OBJECT-COMPUTER paragraph, the source program is compiled at base machine location 800, and an execution under MCP II control is achieved. If MULTIPROGRAMMING is specified under the OBJECT-COMPUTER paragraph, the resultant object program will automatically contain float codes for multiprogramming.

COBOL COMPILATION FOR OPERATING SYSTEM.

The following must be observed within the ENVIRONMENT DIVISION of the source program so that the object program can be added to the program library (figure 9-1).

```
LINE NO   A        B
4    6  7  8    11 12
01        ENVIRØNMENT DIVISIØN.
02        CØNFIGURATIØN SECTIØN.
03        SØURCE-CØMPUTER. B500.
04        ØBJECT-CØMPUTER. B500
05            ØP-SYSTEM
06            MEMØRY SIZE 19200 CHARACTERS
07            ASSIGN ØBJECT-PRØGRAM TØ LINE-PRINTER
08            ASSIGN ØBJECT-PRØGRAM TØ TAPE.
09
```

Figure 9-1. COBOL ENVIRONMENT DIVISION Example

The OP-SYSTEM statement will cause source program output to be compiled at base 800. If MULTIPROGRAMMING is specified the program will contain float codes for multiprogramming. The resultant Collector Tape may contain several object programs that will be subsequently loaded to the User Program Library.

COBOL programs must be loaded into the User Program Library from the Collector Tape using the CPAT Function. Specifications for CPAT are the same as LPAT and may be found in Section Four of this manual.

After the COBOL program is loaded to the User Program Library and a program delete (DPDL) is executed, the output from the DPDL Function (back-up library tape) must be reloaded using LPAT.

CPAT is only used to initially load COBOL object programs from a Collector Tape.

The COPR Function of MCP II is used to load the B 500 COBOL Compiler to disk. The COBOL Binary Card Image Tape must be mounted on TSU #1 with scratch tape on TSU #2.

The format of this function is:

```
≥ ≥ ≥ C  O  P  R  a  b  b  b  c  d  e  f
1 - - -  -  -  7  8  9  -  11 12 13 14 15
```

where:

a. a - disk segment size
   - 1 = 480-character segments
   - 2 = 240-character segments
   - 3 = 96-character segments

b. bbb - base disk address where the compiler is to be loaded onto disk. The address must be expressed in thousands. An electronic unit zero is assumed and cannot be changed.

   Example:

   015 indicates a disk address of 0015000.

c. c - MCP II indicator
   - 1 = object program to run with MCP II.
   - 0 = object program is not to run with MCP II.

d. d - line printer #1 size
   - 1 = 132-print positions.
   - 8 = 120-print positions.

e. e - line printer #2 size
   - same as printer #1. Must be coded even though two line printers may not be available.

f. f - maintenance type
   - 0 = create base tape and load tape to disk.
   - 1 = load base or intermediate tape.

An example of a ≥ ≥ ≥ COPR entry is as follows:

```
≥ ≥ ≥ COPR20201110
```

This example will create a COBOL Base Tape, and load it to 240-character disk segments starting at 20,000. The object programs will run with MCP II.

If system memory is used, the compiler should be loaded no lower than segment 6,000. The resident compiler requires approximately 1400 segments of 240-character disk. The 2,000 segments immediately preceding the compiler, and the area from segment address 0000000 to N-1 (depending on the size of the source program) will be used as an I/O work area.

BLANK EXECUTIVE ROUTINE SAVE AREA ($\geq \geq \geq$ BLNK).
The BLNK Routine is an automatic function called by COPR to save the Executive Routine, while the COPR Function is being executed.

SET-UP COBOL COMPILER LOADER ($\geq \geq \geq$ SCCL).
This function is automatically called at the conclusion of the COPR Function to pass parameters from COPR to the CMPL Function.

It is necessary to reload the compiler with COPR whenever the Operating System is initially loaded to disk; otherwise, COBOL compilations will not be processed under MCP II Control.

COMPILE COBOL SOURCE PROGRAM FUNCTION (> > > CMPL).
This function will save the Executive Routine in the area called BLNK, pass any changes in printer size and/or the MCP II indicator to the COBOL Compiler, load the compiler into memory, and execute the compilation of a COBOL source program.

The operating instructions are as follows:

a. Mount a scratch tape on MTU-1.

b. Mount a scratch tape on MTU-2 for the object program Collector Tape.

c. Ready the line printer.

d. Enter CMPL via card reader 1 or the SPO.

The format for the CMPL Function is as follows:

```
> > > CMPL a  b  c  d
1 - - ---7 8  9  10 11
```

where:

a. a - MCP II indicator.
   - 1 - object program to run with the MCP II.
   - 0 - object program not to run with the MCP II.

b. b - Line Printer #1 size.
   - 1 - 132 print positions.
   - 8 - 120 print positions.

c. c - Line Printer #2 size. Same as printer #1.

d. d - Syntax only.
   - 1 - compile for syntax only.
   - Blank - compile with object code.

NOTE

If any of the parameters are changed, all the parameters must be re-entered.

The following message may be displayed on the SPO.

Message:

PARAMETERS NOT CORRECT

The parameters entered with the function call were not correct. A restart is required.

Figure 9-2 illustrates the position of assembler information.

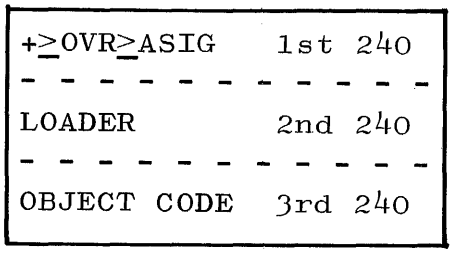


Figure 9-2. Assembler Program in the User Library

COBOL MULTIPROGRAMMING.

The COBOL programs are loaded to the User Program Library in an auto-load format. The first block of the program (an assigner) is read into machine location 400 with control transferring to machine location 400. The COBOL Assigner Routine will select information from the Executive, store the Executive in the reserve memory location, and read the second block of the program (loader) into machine location 000. The loader reads the object program block to the address as required, and then transfers the object code to main core at its new execution address.

An ASG block follows each overlay section. When the ASG block is recognized, control will transfer to the ASG Routine. The ASG Routine will test for subscript 61Axxx in the object program. If 61Axxx is found, the four position field (Axxx) is added to decimally, and a test for +BEG in the first word of the object program is performed. If no +BEG in the first word of the object program is performed, and if no +BEG is found, the object code is considered an overlay and will be written onto the disk at the location specified by the file limits

of the source statement. After the overlay has been written, control is returned to the loader to read the next block(s) of object code. If +BEG is found, the Executive will be restored and updated with I/O requirements, the next program start address etc., after which control will pass to the Executive (figure 9-3).

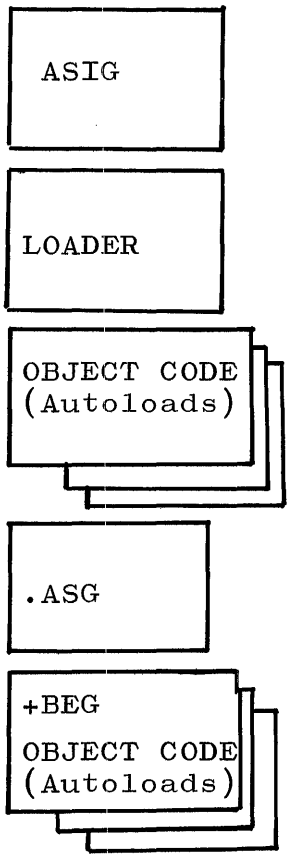


Figure 9-3. Segmented COBOL in the Disk Library

SECTION 10
PROGRAMMING SPECIFICATIONS

GENERAL.

This section emphasizes some MCP II programming techniques. Most of the examples use the Operating System Assembler (ASOP) format and a few COBOL techniques. The programmer can use either Basic or Auto-load formatted programs (provided they have the necessary linkage to the Executive Routine for execution).

In order to use MCP II in a multiprogramming environment, the following multiprogramming specifications must be used.

The Operating System execution of a relative user program is presented in the following format.

MULTIPROGRAMMING SPECIFICATIONS.

In order to effectively use MCP II as a multiprogramming system, the following programming specifications must be followed:

a. Programs may be written in ASOP (Advanced Assembler language), or B 500 COBOL.

b. All addressing must be symbolic except for the Set Location Counter and Overlay entries.

c. The use of SAD2 is not allowed.

d. SAD3 may be used for address tables in a normal address position of a word (OP, A, B, or C).

e. MCP II Multiprogramming Macros must be used for input/output.

f. If input/output operations are infrequent or non-existent in a program, the Multiprogramming Interrupt Macro should be placed within each execution cycle of the program.

g. The sequence of a user program is relatively unrestricted following the initial entries. The initial entries must be in the following sequence.

   1) The Begin Run Macro must be the first entry.

   2) All files must be described immediately following the Begin Run Macro.

   3) All records must be described immediately following the File Descriptor Macros.

   4) Constants should be placed after file and record descriptors.

h. Files must be OPENed prior to the execution of a READ or WRITE Macro.

i. If the initial positioning of the file is required, the Position Macro must be executed.

j. Files must be CLOSEd prior to the execution of the STOP-RUN Macro if the units are to be returned to the Systems I/O Table at end-of-job time.

k. After the execution of the file CLOSE Macro, files cannot be accessed by the execution of another file OPEN Macro.

INITIALIZING THE SYSTEM.

The following steps are used by MCP II to execute a multiprogramming program.

The program call is initiated via the card reader or SPO.

a. An interrupt occurs.

b. The Executive branches to machine location 260 and sets the SPO initialize flag (a one (1) in machine location 179), and branches to 050 in the Executive.

c. IFCC (Interrupting Function Call Check) is called to determine the following:

1) Type of input (Function CALL or user program CALL).

2) Checks to see if Save Memory is required.

3) Sets the Program/Function Switch at machine location 028 to P.

4) Checks machine location 318 to determine if the system is in a multiprogramming mode.

5) Transfers the disk address of the User Program Library from 11# to 144 in the Executive Routine.

EXECUTIVE.
The Executive reads the User Program Directory Library Record, and transfers control to the Directory.

CALL RECORD BLOCK.
The Directory Record transfers three words from the program to the function call input area (location @ 370).

The three words consist of:

a. Program disk address.
b. Program identification.
c. Program input/output requirements.
d. Program MODE (Relative COBOL segmented, etc.).
e. Program blocks.

Program IDENT is compared, and the MODE checked.

The input/output requirements are transferred to the multiprogramming CALL work area at machine location 270, and control is transferred to machine location 330 in the Executive.

MULTIPROGRAMMING CONTROL OVERLAY LOADER.

a. Calculates the relative address.

b. Adds the relative address to the base.

c. Branches to machine location 060 which loads the Memory Check Function.

MEMORY CHECK FUNCTION.

a. Tests the multiprogramming flag at machine location 318 for MODE.

b. Checks machine location 31@ for the number of programs in process.

c. Determines if enough memory is available to execute the program.

d. The linkage is set in the Executive Routine to call the Input/Output Check Routine.

INPUT/OUTPUT CHECK ROUTINE.

a. Stores the I/O Table located at machine location 290 to machine location 400.

b. Program I/O requirements are checked against the Systems I/O Table for available units.

c. Assigns the I/O units to the program and then removes the I/O units from the Systems I/O Table.

d. Linkage for loading the Memory Assigner Routine is set, and the linkage accomplishes the following:

1) Transfers the Revised System I/O Table from machine location 400 to 290.

2) Transfers Revised Program Table to Executive Program Table at machine location 280.

3) Transfers the ADM counter from machine location 304 to 356.

4) Transfers the beginning disk address from machine location 309 to 306.

5) Increments the program counter at machine location 31@.

6) Branches to machine location 060 to load the Assigner Routine.

ASSIGNER ROUTINE.

a. Tests the ADM counter for 00. (00 identifies the first multiprogramming program in the mix.)

b. Sets the transfer to ADDRESS.

c. ADM's beginning address.

d. Transfers the first 10 words of the program into memory.

e. Decreases the program block counter at machine location 396.

f. Performs steps b thru e, until all the program blocks are loaded, and the linkage for loading the Date Check Function.

DATE CHECK FUNCTION.

a. Tests for relative overlay, if the relative overlay recalls the multiprogramming loader.

b. Tests for DATE. (Second word of object program.)

c. Checks program type at machine location 380 for an "R."

d. Transfers the first two words of the program to 35#.

e. Tests for TODAY's and REPORT dates.

f. Prints BEGIN RUN message on the SPO.

g. Sets linkage for loading Input/Output File Declaration Function.

INPUT/OUTPUT FILE DECLARATION FUNCTION.
This function stores the following items in the I/O Control Segment.

a. Interrogate, OP CODE, M-VAR.

b. Table position length.

c. Program I/O Table position.

d. I/O command (OP CODE & M-VAR).

e. Sets the linkage for loading the second I/O File Declaration Function.

INPUT/OUTPUT FILE DECLARATION #2 FUNCTION.

a. Tests for error flag (2) at machine location 382, and sets a discontinue if equal. The error flag will be set by IOFD.

b. Assigns the printer or tape to the file requesting the line printer with tape backup.

c. Prints the file name and the type of I/O assigned on the SPO.

d. Sets up the user program return address linkage table (machine location 049).

e. Branches to the Executive which will load the multiprogramming controller.

f. Control will transfer to the user program to begin execution.

FILE OPEN FUNCTION (OPNF).

a. The OPNF Function will store the file name and disk file address for the open message.

b. Sets the address of the I/O Control Segment.

c. Determines the type of I/O device.

d. Assigns the I/O unit number (inserted into the I/O Control Segment by the I/O File Declaration Routine).

e. Deletes the I/O unit from the Program I/O Table.

f. Restores the updated I/O Control Segment and Program I/O Table.

g. Prints the file OPEN message on the SPO.

h. Sets the linkage to recall the Multiprogramming Controller.

NOTE

When an end-of-file condition occurs, the user program will set the linkage (Close Macros) to call the File Close Function.

FILE CLOSE FUNCTION.

a. Saves the file name.
b. Tests the type of file.
c. Restores the unit to the Program I/O Table.
d. Removes the unit from the I/O Control Segment.
e. Sets the programs return address.
f. Prints the file CLOSE message on the SPO.
g. Sets the linkage and recalls the Multiprogramming Controller.

END-OF-PROGRAM FUNCTION.

a. Turns the Call Record Switch on, and transfers A "#" to machine location 015.

b. Returns the I/O units (from the Program I/O Table) to the Systems I/O Table.

c. Eliminates the program from the Return Linkage Table.

d. Restores memory space to the system.

e. Prints the End-of-Program message on the SPO.

f. Reduces the program count.

g. Tests for ZIP.

h. Tests for more than one program in the mix.

i. Tests program TANK.

j. Recalls Multiprogramming Controller.

MCP II CAPABILITIES.

a. The capability of customizing the Disk Operating System to utilize it to its fullest intent.

b. Maintains a User Program Library containing Basic, Advanced, or COBOL programs, and the capability of loading a program with minimum effort.

c. Assigning of dates through the Operating System.

d. The ability to call program overlays.

e. The automatic handling of End-of-Job or discontinue routines for return to the Executive Controller.

f. Allows serial jobs to be interrupted so that the execution of other programs can be accomplished, and automatically restores and restarts the program that was interrupted.

g. The capability to programmatically call another program or function.

MULTIPROGRAMMING GENERAL SYSTEM FEATURES.

a. Multiprogramming of a maximum of three user programs.

b. MCP II Macros to automatically handle I/O.

c. Discontinue of one or all programs in the mix.

d. Available core query and/or what programs are presently in the mix and their core requirements.

e. Automatic assignment of peripheral units and control of the I/O Table.

f. Automatic relocation of programs at the time of execution.

CUSTOMIZING THE OPERATING SYSTEM.

This feature is accomplished with the Delete (DELF) and Add (ADDR) Functions: Both functions are on the Tape Operating System. The DELF Function provides the capability of deleting any function(s) that are not required, and the ADDR Function provides a means of adding functions to the Disk Operating System. These functions are used to free library space and to minimize the time needed to load a program and repack the User Program Library.

PROGRAM LIBRARY.

In order to load a user program into the User Program Library, a program PADD record must be furnished each object program. This record is automatically provided when the Begin Run Macro or OP-SYSTEM is used. For other types of programs the user can insert a SLC at machine location 750 followed by the PADD record specification

described as constants. This entry will create a PADD Record during assembly. The Object Program must be assigned with the first word residing at machine location 800. The first two words at machine location 800 and 810 are reserved for the Operating System. Machine location 800 must contain the following BEGIN Program Label.

a. Position 1-4 +BEG-positions.

b. Positions 5-9 Program identity.

c. Positions 10-12 Address of first instruction to be executed in object program.

The second reserved word at machine location 810 is used for Date Assignment Codes. This field is blank if date assignments are not used. The following data represents the necessary coding for the date assignment.

a. (position 1) +.

b. (position 2) D if either Today's or Report is required.

c. (position 3) Y if Today's date is required.

d. (position 4) Y if Report date is required.

e. (position 5) Today's date type code or blank.

f. (position 6) Report date type code or blank.

g. (positions 7-9) the address in the user program used to store today's date.

h. (positions 10-12) the address in the user program used to store report date.

The following are acceptable types of date format codes.

Type Code	Format	Type of Date
A	Alpha-month DD, YYYY	Month-day-year
B	DD Alpha-month YYYY	Day-month-year
J	Alpha-month DD, YYYY	Abbreviated month-day-year
K	DD Alpha-month YYYY	Day-abbreviated month-year
1	MM-DD-YY	Month-day-year
2	DD-MM-YY	Day-month-year
3	YYDDD	Year-Julian day
4	DDDYY	Julian day-year

The dates supplied to the user-specified area are left-justified. The following are several examples of date formatted output.

A January 31, 1970 (alpha: month, day, year)
B 01 April 1970 (alpha: day, month, year)
J SEPT 05 1970 (abbreviated alpha: month, day, year)
K 10 JUN 1970 (abbreviated alpha: day, month, year)
1 01-15-70 (numeric: month, day, year)
2 31-03-70 (numeric: day, month, year)
3 70150 (numeric: year, Julian day)
4 36570 (numeric: Julian day, year)

Figure 10-1 illustrates the date assignment format.

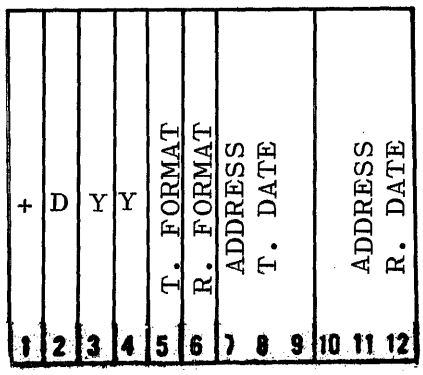


Figure 10-1. Date Assignment Format

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01			SLC			@750									
02			CST		7	≥≥≥PADD									FUNCTION CALL ID
03			CST		5	IDENT									PROGRAM IDENTITY
04			CST		1	M									TYPE PROGRAM
05			CST		1	W									BEGIN-RUN HALT
06			CST		16										RESERVED
07			CST		202										MAIN CORE BLOCK COUNT
08			CST		40002										TOTAL BLOCK COUNT
09			CST		24										
10			CST		12	+BEGIDENT	8	20							BEGIN RECORD
11			CST		12	+DYY	1	1	95096	0					DATE RECORD
12		START													START OF PROGRAM
13															

Figure 10-2. Program Beginning

Figure 10-2 illustrates the symbolic entries required to start an Assembler program.

OVERLAY CALLS.

MCP II has provisions for locating and loading user program overlays. The overlays are located by the names specified within the overlay label (see figure 10-3). A program may call any overlay within the User Program Library, but a multiprogramming program may not call an overlay that does not contain relative address codes.

After the Executive receives the parameters to call an overlay, it searches the Overlay Table to locate the correct disk address segment. The overlay is loaded one block at a time, starting at the address specified in the overlay label. After the completion of the load operation, control transfers to the first word of the overlay.

When an overlay is called, MCP II makes no provisions to save memory before loading, or to restore memory after the overlay has been completed. If these functions are required, the user must make the necessary provisions to call them.

The first overlay instruction must be an overlay label using the following format:

a. (positions 1-6) +≥OVR≥.

b. (positions 7-10) unique overlay name (positions 7-9 must be unique).

c. (positions 11-12) number of blocks in the overlay (each block contains 480 characters).

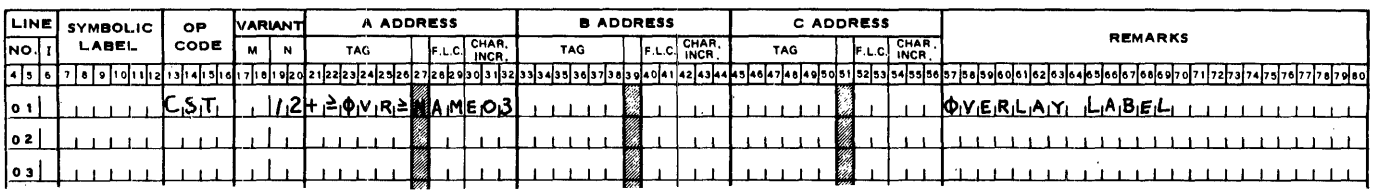
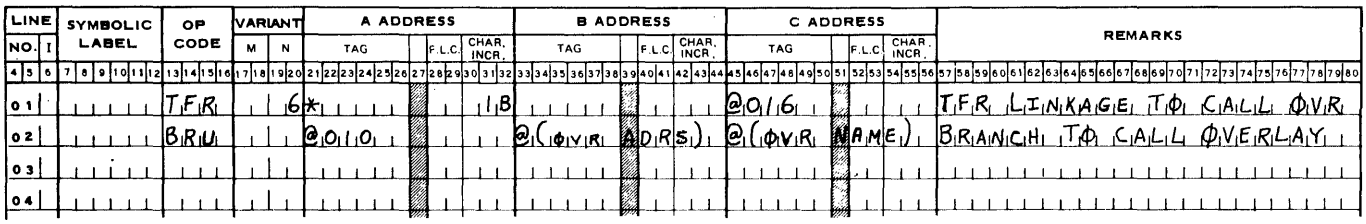


Figure 10-3. Specific Overlay Identification

NON-MULTIPROGRAMMING OVERLAYS. The linkage must be supplied to the Executive to programmatically call the desired overlay. This is accomplished by transferring the name of the overlay into machine location 019, and the address of where the overlay is to be loaded into machine location 016 (see figure 10-4).



LINE NO.	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01		TFR		6*			18				@016			TFR LINKAGE TO CALL OVR
02		BRU			@010			@(OVR ADRS)			@(OVR NAME)			BRANCH TO CALL OVERLAY
03														
04														

Figure 10-4. Non-Multiprogramming Overlay Linkage

After passing the required information, the user program will branch to location 010.

MULTIPROGRAMMING OVERLAY CALLS. Programs awaiting execution in the multiprogramming mode may call those overlays that have been assembled with relative address codes. The macro call for this setting is LDRO. This call requires that the overlay ident and block count (parameter entries) have been established by the ROVR Macro. For additional information refer to the Advanced Assembler II Reference Manual (1042769).

COBOL SEGMENTATION.

COBOL program overlays can be obtained by using segmentation. Overlays are useful whenever memory is exceeded during compilation, or to limit the size of main core. Segmentation is assigning priority numbers to section names. The priority number determines whether the section is resident or an overlayable section. For additional information refer to the COBOL Reference Manual (1045226).

DISCONTINUE (Non-Multiprogramming).

The discontinue of a non-multiprogramming program because of error conditions determined programmatically, can be accomplished by storing

a B-bit in machine location 005 and exiting to machine location 000. The Executive Controller automatically restores this bit after the program is discontinued (see figure 10-5).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01		ERROR	TFR		1	*+1			BCJ			@005			STORE DISCONTINUE BIT
02			BRU			@000									BRANCH TO EXECUTIVE
03															

Figure 10-5. Non-Multiprogramming Discontinue

DISCONTINUE (Multiprogramming).

To programmatically discontinue a multiprogramming program, it is necessary to pass the parameters to the disk Executive and call the Discontinue Function (see figure 10-6).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01			TCB		13	*+1			10@020			@356			TFR DISC FIC AND BRANCH
02			CST		13	>>>DISC		IDENT	←						PARAMETERS FOR DISC
03															

Figure 10-6. Multiprogramming Discontinue

END-OF-JOB.

The End-of-Job Routine is called after the completion of each user program, and to return control to the Operating System (see figure figure 10-7).

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01		EOJ	BRU			@000									BRANCH TO EXECUTIVE
02															
03															

Figure 10-7. Branch to Executive Controller

STOP RUN.
The COBOL STOP-RUN construct generates a branch to machine location 000 and returns control to the disk Executive.

DATA COMMUNICATION INTERRUPTS.
Data Communication Interrupts are somewhat restricted by the specific requirements that must be met. The user must provide two one-block overlays residing in the User Program Library. The overlays are used to control the inquiries for determining I/O replies. The names and location are reserved, and each overlay must be assembled at machine location 400. With the input ready conditions, the DFI (reserved overlay name) overlay is used to process data communication input messages on input ready conditions.

The overlays must provide the readying of input messages by processing or programmatically calling the program to process the input message, and for transmitting additional replies for multiple buffer load messages.

After detecting a data communication interrupt the Operating System performs only the following functions:

a. Interrogates all the terminals for an input or output ready condition.

b. Upon detection of a ready condition, the executive will store the terminal unit number at machine location 108 and call either the DFI or DFO overlay, and transfer control to that overlay.

If an additional program is brought into memory the DFI or DFO overlay is destroyed. Therefore if the message and/or any additional information from DFI or DFO is required for the execution of the program, the data must be stored below the area that will be used by the program to be called.

For example if the inquiry program is eight blocks long, the information to be retained can be stored in the area between 0X0 and the End-

of-Memory will automatically be saved before loading the inquiry program (unless the overlay was entered from the End-of-Job Routine). After the processing of the inquiry has been completed, the user may branch to machine location 030. The saved contents will be restored automatically, and control will return to the point originally specified by the interrupt linkage.

PROGRAMMED INTERRUPTS NON-MULTIPROGRAMMING. At some point(s) during processing the programmer may want to interrupt a non-multiprogramming program and have the Operating System interrogate the supervisory printer, data communications terminals, and/or card reader for an inquiry or function call (see figure 10-8). The following information must be furnished to the OP/System.

a. The address within the program where control should be returned after the interrupt has been completed. This address must be stored at machine location 043.

b. Branch to machine location 030.

LINE NO.	I	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01			TFR		3	*		B	RETURN			@043			TRANSFER RETURN ADDRESS
02			BR	4		@030									BRETURN FOR INTERRUPT
03		RETURN													TEST
04															
05															

Figure 10-8. Return Address Branch for Interrupt Test

MULTIPROGRAMMING INTERRUPTS. Multiprogramming Interrupts are supplied to a user program by Macro routine(s), and after each I/O operation. For those programs that require very few I/O operations, the Interrupt Macro (M/PI) should be used. The Interrupt Macro will store the necessary program return linkage, and release control to the Executive.

COBOL INTERRUPTS. The INTERRUPT Verb generates the linkage to the Operating System for a test of the input requests. If an input request has been made, the Executive will call the Save/Restore Memory

Function (if required), call the function or program requested, and release control to the requested function or program. At End-of-Job the Executive Controller will restore memory (if it had been saved) and control will return to the program issuing the interrupt. If an input request is not detected, control returns to the instruction immediately following the interrupt.

Maximum multiprogramming benefits can be derived through the use of this verb whenever a few I/O routines are used by the program.

PROGRAMMATIC FUNCTION CALL.
A user program can call for the execution of any function within the Operating System.

If a program calls a function, the following linkage to the Operating System must be furnished by the programmer.

a. The address from the program where control will return after the function has been executed. The return address must be stored in machine location 043. If the user does not want to return to this program after execution of the function, the word END must be stored at machine location 043.

   If required, the disk Executive will automatically save memory on the reserved area of disk.

b. When the tape Executive has control, and the user wants control returned to his program, the Save and Memory Unit Code must be stored at location 045.

   The Codes are as follows:

   1) 1 through 5 - tape unit 1 through 5 where memory will be written.

   2) # - cards.

c. Three tape marks (≥ ≥ ≥) must be stored at location 356.

d. The four-digit function ID must be stored at location 359.

e. The parameters required by the function must be saved starting at machine location 361.

f. A branch to machine location 020 must be executed.

For example, assume that the program wants to clear an area of disk programmatically and return. The following linkage within the program will clear an area of 240-character disk segments from 0010000 through 0010999 with the CLEAR character "A"(figure 10-9).

LINE NO.	SYMBOLIC LABEL	OP CODE	VARIANT M	VARIANT N	A ADDRESS TAG	A ADDRESS F.L.C.	A ADDRESS CHAR. INCR.	B ADDRESS TAG	B ADDRESS F.L.C.	B ADDRESS CHAR. INCR.	C ADDRESS TAG	C ADDRESS F.L.C.	C ADDRESS CHAR. INCR.	REMARKS
01		TFR		3	*			BRETURN			@043			STORE RETURN ADDRESS
02		TCB		26	*			1@0020			@356			F/C AND BR TO EXEC
03		CST		26	≥≥≥CLDK			001000000010			9992401			PARAMETERS
04	RETURN													CONTINUE PROCESSING
05														

Figure 10-9. Stores Return Address

COBOL FUNCTION or PROGRAM CALL.

The COBOL verb ZIP will programmatically call another program or function. The execution of a program or function is performed by ZIPing to a data-name. The data-name must contain the program or function call parameters. After the completion of the program or function, control will return to the next instruction in the sequence. Reference COBOL Manual (1045226) for additional information related to ZIP.

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: B 500 SYSTEMS
MASTER CONTROL PROGRAM II
Reference Manual

FORM: 1057205
DATE: 2-72

CHECK TYPE OF SUGGESTION:

☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

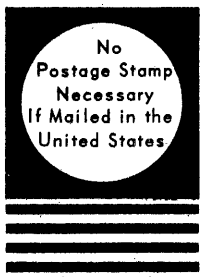
FROM: NAME _____
TITLE _____
COMPANY _____
ADDRESS _____
_____

DATE _____

cut along dotted line

STAPLE

FOLD DOWN SECOND FOLD DOWN



BUSINESS REPLY MAIL
First Class Permit No. 817, Detroit, Mich. 48232

Burroughs Corporation
6071 Second Avenue
Detroit, Michigan 48232

attn: Sales Technical Services
Systems Documentation

FOLD UP FIRST FOLD UP

1057205 2-72 Printed in U.S. America