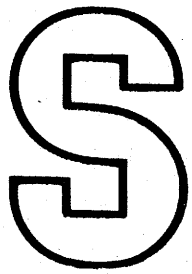
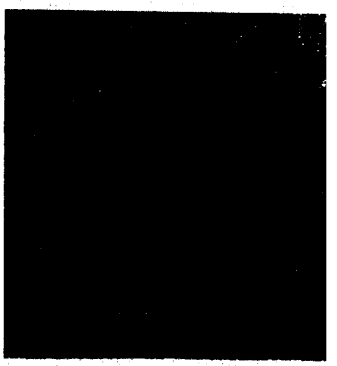
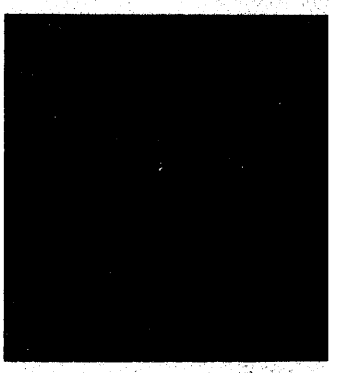
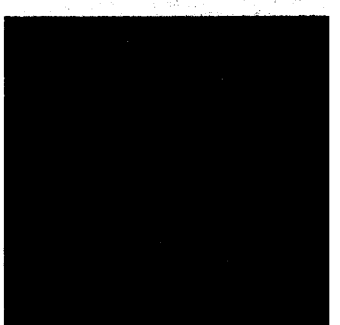






IBM



Reference Manual

IBM 7030 Data Processing System

Master Control Program

IBM

Reference Manual

IBM 7030 Data Processing System

Master Control Program

MAJOR REVISION (June 1964)
This manual; Form C22-6678-1 obsoletes Form C22-6678-0 and Technical Newsletter, Additions and Corrections to the MCP Reference Manual, N22-0085 and should be reviewed in its entirety.

Los Alamos personnel worked closely with IBM in planning this Master Control Program.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, Dept. D83, PO Box 390, Poughkeepsie, N.Y. 12602

CONTENTS

The Master Control Program (MCP) for the IBM 7030 Data Processing System is described in this manual with the assumption that the user has a knowledge of the IBM 7030 and the 7030 assembly program, STRAP II.

The general characteristics of MCP are:

Automatic Operation
Processor Supervision
Symbolic and System I/O

Automatic Operation

As an automatic operator, MCP loads problem programs in a continuous sequence requiring specific control cards which contain instructions and descriptions of the problem program in order to schedule a queue of jobs.

MCP accepts commands from the operator to alter the mode of operation, to change the normal sequence of jobs to be run, or to change the system I/O configuration.

Instructions issued by MCP, enable the operator to set up jobs before they are run and to service I/O units after a problem program has started operation. The identification of jobs and the phase of execution are also communicated to the operator for logging purposes.

MCP controls the interrupt system to ensure its own continuity. After all MCP interrupts are handled, the problem program interrupts are routed to the problem program with the correct machine status. If the problem program interrupts are unexpected, MCP automatically issues an indication to the problem program, dumps the machine status, and proceeds to the next problem program, thus eliminating operator intervention for these error situations.

Processor Supervision

By using a "type of problem" card, MCP sets up and controls the sequence of processors that are to operate on the problem program. For the processor programs, MCP provides a common communication region by means of which MCP can communicate with processors and processors can communicate among themselves. The information provided in this region consists of identification of the chain of processors, the type of run, reject indicators, and a description of the I/O requirements of the processors.

MCP also maintains a system disk that contains the processors themselves and a library of subroutines. Subroutines, in whole or in part, may be fetched by the processors. The fetching mechanism incorporates a buffer and a dictionary of names of the material on the disk.

The system disk also includes programs that are not subroutines or true processors but are problem programs acting as processors. To eliminate the handling of large, often used problem program card decks, the problem program may be placed on the system disk. The "type of problem" card becomes the only card handled in place of the entire binary deck.

Symbolic and System Input-Output

The configuration of I/O devices varies from installation to installation and at times, within the same installation. MCP assumes the responsibility of maintaining a record of the status and availability of all I/O devices attached to the exchange. MCP can then assign I/O devices to problem programs and prevent conflicting requirements between them.

The I/O requirements for any problem program may change from run to run. To provide for the greatest flexibility, the problem program describes its I/O requirements by means of Symbolic I/O Definitions (IOD). In the source program, an IOD defines the symbol to be used by the problem program in any MCP calling sequence. In the object program, the IODs furnish the I/O requirements in advance of their use so that absolute channels and units can be assigned and made ready. Before execution, the IODs are expanded into tables by means of which MCP controls the corresponding units. The symbol naming the IOD is equated to a reference number by the compiler. In turn, this number locates the appropriate IOD table to be used during execution.

Symbolic I/O permits programming without absolute references to channels and units. It implies that maintenance of the status of I/O units and the issuance of absolute hardware instructions to the I/O units is a function of MCP. I/O requirements must be given to MCP in symbolic form in both the source and object program.

In addition to assigning tape units symbolically, MCP includes the ability to check the requested tape reel against the reel which is actually mounted by the operator. When a working tape to be used at a later date as a special tape, the label is recorded in a "reel history" included at the end of the problem program output and the operator is informed that the tape is to be put in the library.

MCP allows the problem program to use system as well as symbolic I/O. Under system I/O, all

the necessary control words, mode, density, interrupts, etc., are handled by MCP. The programmer need concern himself only with providing MCP with locations and sizes via a calling sequence. The system I/O includes input, output, disk fetching of prestored routines, and console typewriter facilities.

Minimum I/O and Core Storage for the 7030

The minimum input-output and storage requirements for the IBM 7030 Data Processing System under MCP are:

An IBM 1401 Data Processing System (Tape-Oriented), used for off-line operation.

A minimum of three IBM 7302 Core Storage units.

An IBM 7612 Disk Synchronizer with at least one disk unit used to store system programs and as auxiliary storage for the problem program.

An IBM 7619 Exchange with the minimum eight basic exchange channels is used as follows:

1. An IBM 7152 Operator's Console is used by the operator to issue and receive commands controlling MCP. This unit may also be used by the programmer to control his own program.
2. An IBM 7503 Card Reader is used by the system input subroutine to read in problem programs and MCP commands while MCP is in either the on-line overlapped or bypass mode of operation. While MCP is in the off-line overlapped mode, the card reader may be used by the problem program. (See "Normal Running Modes" for a discussion of MCP modes.)
3. At least four IBM 729 IV Magnetic Tape Units with one or more tape control units are used as follows:
   a. The system output tape contains information to be printed and/or punched off-line by the tape-oriented IBM 1401 System. This tape cannot be controlled or used directly by the problem program.
   b. The alternate system output tape is used to eliminate waiting while the other system output tape is being unloaded.
   c. The system read tape, used in an overlapped mode, contains problem programs about to be run.
   d. The system write tape is used with the card reader in the on-line overlapped mode to prepare a tape for subsequent reading. This same tape unit may contain the system scan tape during operation in the off-line overlapped mode.

Note: For optimal performance, the write-scan tape and the read tape should be on different channels, and both output tapes should be on a third channel. Except for these requirements, all other magnetic tape units may be used by the problem program.

All other channels on the basic exchange are not required by MCP or by the system programs.

The system, including the off-line functions, can operate in a limited way when these minimum requirements are not completely satisfied.

1. The system can operate while the tape-oriented IBM 1401 System is temporarily inoperative.
2. The system can operate in the off-line overlapped mode without the card reader.
3. The system can operate with one output tape while waiting for reloading.
4. The system can operate in the bypass mode without the input tapes.
5. The system can operate in either overlapped mode with one input tape unit while waiting for input prescan and rewind time.

Minimum IBM 1401 System

An IBM 1401 Data Processing System (Tape-Oriented) will perform all peripheral operations associated with the IBM 7030 System. These operations are:

Card-to-Tape Tape-to-Printer System Output
Tape-to-Punch System Input

The System Input and System Output operations are performed by 1401 programs supplied with the 7030 programming system.

The operations require an off-line system with the following minimum components:

1. IBM 1401 Model C3 Processing Unit with Column Binary Feature.
2. IBM 1402 Model N1 Card Read Punch.
3. IBM 1403 Model 2 Printer.
4. IBM 729 II or IV Magnetic Tape Units (at least two).

Alternatively, an IBM 1401 Model E3 System may be used with IBM 7330 Magnetic Tape Units.

THE INPUT DECK AND COMPILING

Input through MCP is defined as that read by MCP's system input program. Every problem program must be entered through MCP and must provide definition cards that identify and describe it to MCP. Beyond this requirement, the problem program may either read its own input or request MCP to read it through the system input program.

Definition Cards

Definition cards are processed only by MCP or by a processor program; they are never given to the problem program. They are classified by a unique punch, B, in column 1. Definition cards have four fields:

Column	Field
1	B
2-9	Name
10-72(62)	Statement. JOB, TYPE and LIM cards can be punched to column 72; IOD and REEL cards can be punched only to column 62 because the processor punches information in columns 63-72.
73-80	Program identification and sequencing. Information in this field does not affect the operation of the program in any way.

Commas separate the statement field into subfields that are subject to a right-to-left drop-out.

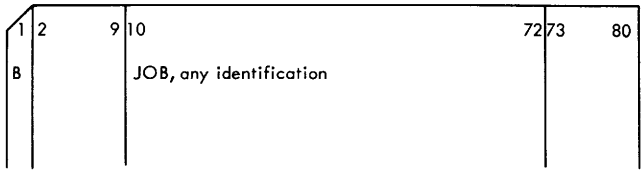
There are five types of definition cards:

JOB	LIM	REEL
TYPE	IOD	

One JOB card and one TYPE card must be at the beginning of every problem program in that order. If other definition cards are required, they must follow these cards. They must also precede any other input from the problem program, except T cards. T cards may be mixed with LIM, IOD and REEL cards.

JOB Card

The JOB card is always the initial card in an input deck. It serves as a logical separation between jobs; its appearance signals the beginning of a new job. The format of the JOB card is:



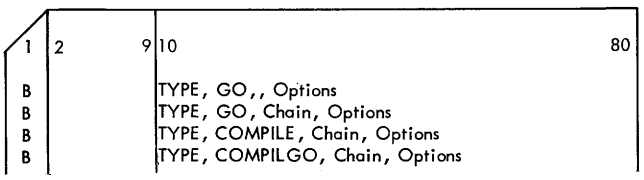
The characters JOB, must appear in columns 10-13.

The first eight characters of the identifying information, normally the name of the job, are printed out on the typewriter when the job starts. They are also written on the output tape. At the option of the installation, an accounting program may also make use of the identifying information supplied on JOB cards.

MCP always supplies a reproduction of the JOB card on the output tape to separate and identify the punch output.

TYPE Card

The TYPE card is the second card of the problem program input deck. It defines the actions that MCP must take in the total job processing. If the problem program requests a compilation as part of the job process, the TYPE card must name the processor chain; it may also indicate special output options. The format of the TYPE card may be any of the following:



Chain indicates the name of a list of processors that must operate on this problem program. Chain may include installation chains of processors as well as the following:

FORTRAN (TYPE, GO, FORTRAN)
FORTRAN (TYPE, COMPILGO, FORTRAN)
SMAC
STRAP

Options is a group of fields on the TYPE card following the chain field. The group has a maximum of 15 fields which need not be in any particular order within the group. Six of these are standard output options while nine are installation options which may be added to the system. Each option field causes MCP to set an appropriate indicator in the communication region. The problem program and processors can then test these communication bits for their own purpose. The six standard output options are:

NRL: STRAP name reference listing included on output

SEQ: STRAP adds a sequence number to each symbolic card

NOLIST: Processor suppresses listing

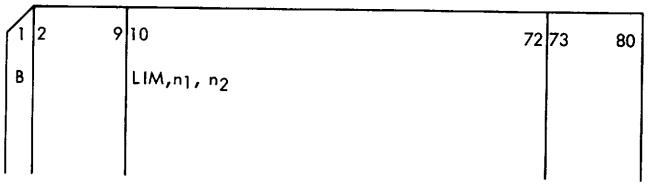
NOPUNCH: Processor suppresses binary deck punching

TITLE: A TITLE is added to each page of GO phase output

PBINARY: The entire binary deck including library subprograms will be punched by BSS.

Limit Card

This card states the core storage requirements for a problem program class that does not require dynamic storage allocation. The STRAP II chain can produce this class of problem program. The LIM card is input with the problem program only for GO type jobs and must follow the TYPE card. The LIM card is part of the output produced by the processor chain. The format is:



n_1 and n_2 are six-digit octal integers (with leading zeros if necessary).

n_1 is the full word (18 bit) address that contains the problem program's lowest bit address. n_1 must be $\geq 41_8$.

n_2 is the full word (18 bit) address immediately following the problem program's highest bit address. Enough storage must remain above n_2 to accommodate tables set up by MCP for this problem program plus MCP itself. (The size of these tables is discussed under "IOD Card.")

At load time, n_1 becomes the lower boundary and n_2 the upper boundary of unprotected storage. The tables that the problem program requires will be above n_2 in protected storage.

If the problem program consists of more than one load, the limits must be set so that the lower limit includes the lowest limit of any of the loads, and the upper limit includes the highest limit of any of the loads.

IOD Card

The configuration of I/O devices can vary not only from installation to installation but also, at times, within a single installation. MCP assumes the responsibility of maintaining an up-to-date record of the status and availability of all I/O devices attached to the exchange. The system can then assign I/O devices to problem programs and prevent conflicting assignments of these devices. Moreover, the I/O requirements for any problem program may change from one run to another. To provide greatest flexibility, the problem program describes its I/O requirements by means of symbolic I/O definitions, the IOD cards.

An IOD card describes symbolically an I/O requirement, or logical I/O unit, by giving parameters that specify the type of I/O device to be used; how the device is to be operated; and the exit to be taken upon I/O interruption. Several IOD statements (logical units) can share one physical unit. The position of the IOD cards in an input deck is shown for all cases in the section, "Preparation of the Input Deck".

The processors reproduce IOD cards as they are submitted and include them in the punched output. The processors punch two additional fields in each duplicate IOD card: the I/O reference number field (column 63-66) and the absolute exit field (columns 67-72). Thus, when writing an IOD statement, the programmer can use only columns 1-62.

The problem program names each IOD card with a symbol and thereafter refers to the particular logical I/O unit by using this symbol as a parameter in a pseudo I/O operation.

The IOD card is divided into five fields:

Class: Column 1 contains the single character "B" to classify the card as a definition card provided by the problem program.

Name: Among definition cards, the name field is used only on IOD cards. Columns 2-9 contain a problem program symbol (used in compilations) not more than eight characters long that names the logical I/O unit. This symbol is called the IODNAME. Each IOD must have a unique name. It is a description of a logical unit; each physical unit

is symbolized either by a channel name if a single-unit channel is involved, or by channel and unit names in the case of a multi-unit channel. In the latter case, channel name is a request for grouping physical units. There may be many IOD cards for each physical unit and their order is important if they are indexed by the problem program.

Statement: The statement field of an IOD card contains subfields that vary with the type of I/O unit described. The general format of the statement field is:

IOD, type, exit, channel,....

in which:

Type must be one of the following symbols that define I/O equipment:

PRINTER	READER	DISK	TAPE
CONSOLE	PUNCH	TRACK	

Exit is a problem program symbol (used at compilation) that specifies the location of the I/O table of exits to be used when an interrupt occurs for the logical unit. The symbol may not contain more than eight characters.

Channel is a programmer symbol (it is not used for compilation) that corresponds to an absolute channel number which will be assigned by MCP. The same channel symbol on several IOD cards of the same type of equipment is a request to assign them to the same channel. A different channel symbol on IOD cards of the same type is a request to assign them to different channels. MCP will honor such requests as the I/O configuration and its current usage allow. The symbol may not contain more than eight characters. A null symbol indicates that any channel of that type may be used.

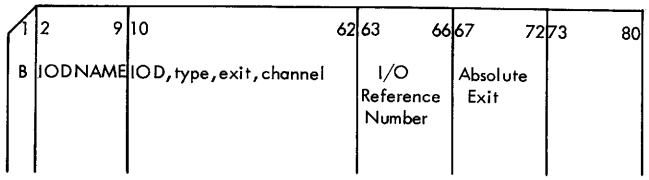
I/O Reference Number: Columns 63-66 are left blank by the problem program. On the output IOD cards, STRAP II converts the IODNAME to an absolute number and punches this four-digit octal number in this field. The first IOD card encountered has the number 1 punched in this field, the second IOD card has the number 2 punched here, and so on.

Absolute Exit: Columns 67-72 are left blank by the problem program. On the output IOD cards, STRAP II punches the six-digit octal equivalent of the evaluated symbol for the first word address of the I/O table of exits in this field.

The name field and the exit subfield are used only on compilations. Upon running the problem program, the absolute values in the I/O reference number and absolute exit subfields are used, respectively, in place of the symbols used for compilation.

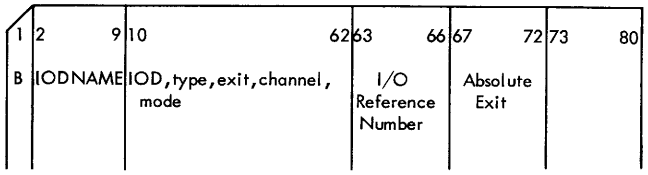
There are four classes of IOD cards. An IOD card is classified by the type of I/O unit to which the symbolic file of information is to be assigned.

PRINTER OR CONSOLE IOD CARD:



where type is PRINTER or CONSOLE.

READER OR PUNCH IOD CARD:

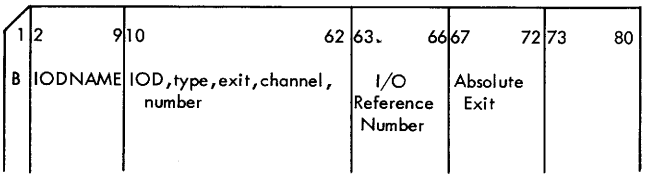


where:

type is READER or PUNCH;

mode is ECC or NOECC. If mode is null, NOECC is assumed.

DISK OR TRACK IOD CARD:



where:

type is either DISK or TRACK;

number is a decimal integer indicating:

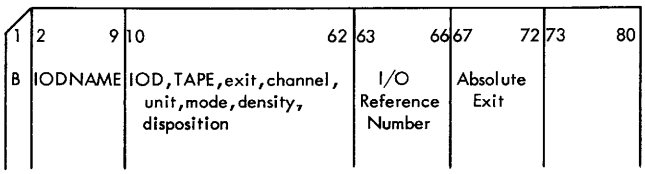
a. Number of arcs if type is DISK, or

b. Number of tracks if type is TRACK. MCP reserves the requested multiple of eight arcs, the first reserved arc being the first arc of a track. This permits the programmer to perform an autoaccess elimination operation, but his control word must meet the requirements of the 7030 as stated in the Automatic Access Elimination section of the Reference Manual, IBM 7030 Data Processing System, Form A22-6530-2.

For both DISK and TRACK, a null number field is a request for all the disk storage available to problem programs. Only one such request per problem program will be honored for an installation with only one disk channel.

When both DISK and TRACK IOD cards are used, they should be grouped by type to optimize allocation of storage on the disk.

TAPE IOD CARD:



where:

unit is a programmer symbol, limited to eight characters, which corresponds to an absolute unit number which will be assigned by MCP. The same unit symbol on several tape IOD cards unconditionally associates them with the same tape unit. A different unit symbol on tape IOD cards unconditionally associates them with different tape units. A tape unit will be assigned for each unique unit symbol, including the null symbol (many null unit fields are treated as a single, unique unit symbol). In turn, tape units will be grouped by the channel symbol but only conditionally. Where both the channel and unit fields are null, there will be unit separation.

The mode field on the IOD card must be one of the following:

ODD	Odd parity, no ECC
EVEN	Even parity
ECC	Odd parity, ECC
Null	See "Reel Labeling"

The density field on the IOD must contain one of the following:

HD	High density
LD	Low density
Null	See "Reel Labeling"

Disposition indicates what is to be done with the tape last mounted when the job is terminated. It must be one of the following:

NSAVE	Do not dismount in any case
CSAVE	Dismount only if job is complete
ISAVE	Dismount only if job is incomplete
SAVE	Dismount in any case
PSAVE	Do not dismount; tape to be passed
NULL	Equivalent to NSAVE

MCP normally unloads those problem program tapes which cannot be immediately reused as scratch tapes. If PSAVE disposition had been specified in the problem program's tape IOD, the end of job procedure would not have unloaded the tape but would have simply rewound it to be used by a later problem program. In the bypass system input operating mode (to be discussed under "Normal Running Mode"), PSAVE must be the tape IOD disposition for tape passing to occur.

In either the on-line or off-line mode operation, MCP prescans the IOD cards and preassigns the required tape reels. MCP is therefore in a position to accomplish tape passing among those jobs which have been preassigned without using PSAVE dispositions. The number of preassigned jobs may be as low as one job. Therefore, to improve the possibility of tape passing between non-contiguous jobs, the problem program should specify a PSAVE disposition.

For all dispositions except PSAVE, a file protected tape will be dismounted. In PSAVE usage, a file protected tape can not be passed to a job that requires a non-protected tape and dismounting will occur.

The programmer must allow adequate space for tables. The amount of space needed is a function of the number of IOD cards in the program, the number of I/O units required, and the number of tape reels required. If:

K = Largest I/O reference number
I = Number of IOD cards
U = Number of units
R = Number of reels, including scratch tapes

the number of full words that must be reserved for tables above the upper limit of the problem program can be computed using the equation:

$K + 7I + 9U + 2R + 1$

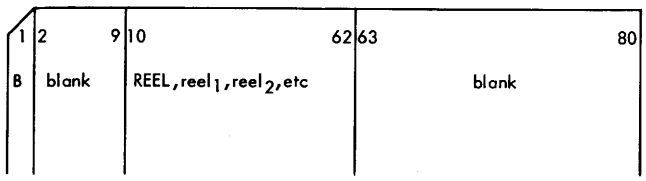
The letter R represents not only the number of reels designated by the REEL cards, but any scratch tapes mounted in response to sequential unload pseudo-operations. Although no scratch tape need be specified, requests for tapes beyond those specified on REEL cards will be honored by mounting scratch tapes. The identity of such tapes will be placed in a pool along with those designated by the REEL cards.

If the computed number of full words is not available, the programmer should not attempt to run his program. If he does, MCP will remove his job from the computer as soon as the available space is exhausted.

Reel Card

The REEL card contains the identification of tapes to be mounted. A REEL card must immediately follow the tape IOD card to which it refers, unless there is

more than one IOD for the same unit and channel. In this latter case, the REEL card must immediately follow one of the IOD cards for that unit and channel. A REEL card may also follow another REEL card that refers to the same unit and channel. A REEL card is punched in the format:



where:

REEL represents a pseudo-operation code that identifies a card which lists reel numbers of tapes associated with a tape IOD statement. $Reel_i$ are subfields that identify tapes in the order in which they are to be mounted. Each $reel_i$ represents a symbol eight characters in length. The first three characters are not part of the reel identification, but specify whether the tape is labeled or unlabeled and whether the tape is protected (ring out) or not protected. If the entire subfield is not null, the $reel_i$ may be:

PLBXXXXX	Protected, labeled
PULXXXXX	Protected, unlabeled
NLBXXXXX	Unprotected, labeled
NULXXXXX	Unprotected, unlabeled

The remaining five characters, XXXXX, are the same as those shown externally on the tape container. If it is a labeled tape, they are the same as those on the (magnetic) reel label file.

If the entire subfield is null, any labeled, unprotected tape will be used. Therefore, it is impossible to specify an unlabeled scratch tape. If all tapes to be mounted on the same physical unit are to be labeled and unprotected, no REEL card is necessary.

REEL HISTORY: A reel history is added by MCP to the end of the problem program output. For every physical tape unit assigned to the problem program, N + 1 lines of output will appear where N is the number of tape reels that have operated on this tape unit. The first line of each N + 1 group contains the actual channel and unit number assigned to the tape request as well as the IOD number (in octal) that last symbolically referenced this unit.

Each subsequent line of the N + 1 group contains the following tape reel information:

1. Reel name
2. UK interrupt count
3. UK-EOP interrupt count
4. Read pseudo-operation count
5. Write pseudo-operation count
6. Control pseudo-operation count
7. Actual tape disposition

REEL LABELING: When a tape is required by the problem program, it is identified by a REEL card. This identification is given to the operator by MCP in mounting instructions typed on the console typewriter. The operator matches this identification with that appearing externally on the tape and mounts the required tape. If the tape is labeled, MCP verifies that the tape requested is the tape mounted.

A labeled tape contains a magnetic identification that corresponds to the external identification on the tape. The magnetic identification is contained in the reel label. The reel label is written as the first record on the tape and is followed by a tape mark. A reel label consists of one (tape) BCD record of at most 15 words and at least 66 bits (11 six-bit bytes) as follows:

0-29	Reel name
30-35	Density
36-41	Mode

with the remaining bits of the record unused.

Reel Name: The reel name consists of five characters that correspond to the exterior reel identification. If less than five nonblank characters are used on the REEL card, rightmost blanks are provided when it is decoded so that five characters are verified.

Density: The density, as coded in the reel label, is that in which the remainder of the tape, including the tape mark following the label, is written. It must be either:

H $(111000)_2$	High, or
L $(100011)_2$	Low

Mode: The mode, as coded in the reel label, is that of the current IOD used when the label is written. It must be one of the following:

O $(100110)_2$	Odd parity, no ECC
C $(110011)_2$	Odd parity, ECC
E $(110101)_2$	Even parity

Label Tape Mark: The reel label record is followed by a tape mark which is the problem program's logical load point. The first tape mark is written in the density as coded in the reel label record. The reel label, including the first tape mark, is rewritten on-line only if all the following conditions are met:

1. The tape is labeled.
2. The first problem program operation that moves the tape is either a write or write end of file.

3. The density or the mode in the label is different from that on the IOD used in the first operation. The density or mode stated in the IOD is used when rewriting the label.

Because the information in the reel label is not used by the problem program, the tape mark following this initial file is used to simulate the beginning-of-tape metallic strip. If the programmer attempts to backspace over this end-of-file mark to the real metallic strip, his action is prohibited by MCP. The tape will be positioned at the end of the reel label and a branch will be made to the error return in the I/O table of exits. (See "I/O Interrupts.") Repeated attempts to backspace over the reel identification will be met with the same response and signal from MCP.

The mode and density options offered by the system are:

Label Mode and Density: The mode and density in which the label is written on the tape.

Information Mode and Density: The mode and density appearing in the label and in which the file mark following the label and at least the first information record are written. The information density is the density in which the rest of the tape is written. The mode used to write a tape may be altered after the first record. Thus, data following the first record can be written in a mode different from that appearing in the label.

The two sets of parameters for label and information mode and density can be altered by compilation. All MCP systems will be delivered with the following modes and densities:

Label -- High density, even parity

Information -- High density, odd parity

Individual installations may elect to alter these modes and densities, in which case they may be incompatible with the modes and densities of other installations.

Installation Mode and Density: The mode and density established by an installation for either or both the label and information modes and densities. These modes and densities may or may not be different from the modes and densities set upon delivery of the MCP system to the installation.

In the earlier description of the mode and density fields on IOD cards, the null case was referred to this section. For the mode field, the following is true:

$READ operation, labeled tape: Use mode specified in the label (information mode).

$READ operation, unlabeled tape. Use installation information mode.

$WRITE or $WEF, labeled or unlabeled tape: Use installation information mode.

In discussing the null case for the density field, it must be realized that the density is set for the entire tape by the first operation. Therefore, the following rules apply only to the first operation:

$READ operation, labeled tape: Use density specified in tape label.

$READ operation, unlabeled tape: Use installation information density.

$WRITE or $WEF, labeled or unlabeled tape: Use installation information density.

On all $READ or $WRITE requests after the first, the density request is ignored, and the tape is operated upon in the density established by the first operation.

Note that if a $READ operation is given with a density specified by the IOD card, the density for the operation performed becomes the density for the entire tape. This density may not agree with the information density, in which case errors will occur. It is advisable to use a null density and mode on the IOD used for reading the first record, or to use the same IOD with which the first information record was written.

Whenever a tape reel is to be written in a density other than the current active density, the rewind and change density pseudo-operations must be issued. The next write pseudo-operation issued to that tape will cause a new label and file mark to be written prior to writing the requested record.

Preparation of the Input Deck

The structure of input decks may be divided into two general classes: (1) those decks which require compilation by one of the language processors before the job can be loaded and executed, and (2) those decks which are output from a previous compilation and can be loaded and executed without compilation.

Jobs to be Compiled

For jobs to be compiled, the programmer may request that the deck be handled by MCP as a compilation only (execution is not to be attempted) by using the COMPILE option on the type card:

B TYPE, COMPILE, Chain, Options

The programmer may request that the deck be loaded and executed at the completion of compilation by using the COMPILGO option on the type card:

B TYPE, COMPILGO, Chain, Options

If STRAP is the name of the processor chain, the deck should have the following structure:

```
B   JOB, identification
B   TYPE, COMPILE (COMPILGO), STRAP
B   IOD (if any)
B   REEL (if any)
    SYMBOLIC CARDS in STRAP language
    PROGRAM DATA (if any) (if option is
    COMPILGO)
```

If FORTRAN is the name of the processor chain, the deck should have the following structure:

```
B   JOB, identification
B   TYPE, COMPILE (COMPILGO), FORTRAN
T   SUBTYPE, FIOD (if any IOD/REEL)
B   IOD (if any)
B   REEL (if any)
    END
T   SUBTYPE, FORTRAN
    SYMBOLIC CARDS
T   SUBTYPE, STRAP
    SYMBOLIC CARDS
T   SUBTYPE, BIN
    PRECOMPILED BINARY CARDS
T   SUBTYPE DATA
    PROGRAM DATA (if any) (if option is
    COMPILGO)
```

Notes: 1. The T SUBTYPE card indicates to the language processor, the language in which the following subprogram is written. A more complete description of T SUBTYPE cards and more examples of input decks may be found in the IBM 7030 FORTRAN IV Reference Manual, Form C22-6751, chapter 12.

2. The preceding example of a sample job deck illustrates that the job may include subprograms in a mixture of symbolic languages and previously compiled binary subprograms.

Jobs to be Executed

A job is ready to be loaded and executed if all subprograms in the deck have been previously compiled. The binary cards which make up the input deck must contain only card classes (See "Card Classes.") acceptable to the MCP loader.

If the job does not require dynamic storage allocation provided by the BSS processor, the input deck to MCP should have the following structure:

```
B   JOB, identification
B   TYPE, GO
B   LIM, A, B
B   IOD (if any)
B   REEL (if any)
    BINARY CARDS (acceptable to the MCP
    loader, including any correction, patch,
    or dump cards)
    BRANCH CARD (absolute)
    PROGRAM DATA (if any)
```

If the job was compiled with the FORTRAN chain of processors, the input deck to MCP should have the following structure:

```
B       JOB, identification
B       TYPE, GO, FORTRAN, TITLE
TITLE   PROGRESS REPORT
B       IOD (if any)
B       REEL (if any)
B       TYPE, GO, FORTRAN (optional)
T       SUBTYPE, BIN (optional)
        BINARY CARDS (acceptable to MCP
        loader, including any correction,
        patch, or dump cards for the
        subprogram)
B       TYPE, GO, FORTRAN (optional)
        PRECOMPILED BINARY CARDS
        FORTRAN BRANCH CARD
        PROGRAM DATA (if any)
```

Notes: 1. If TITLE is specified as a TYPE card option, the TITLE card must immediately follow the TYPE card. Otherwise, the problem program will be ended with a TYPE card error.

2. Any B TYPE or T SUBTYPE cards which appear after the first binary card will be printed. Otherwise, they will have no effect on the input deck.

3. The reading of program data which follows the branch card must be initiated by the problem program.

Card Classes

The MCP loader handles 18 classes of cards. Individual card formats and their functions may be found in the IBM 7030 Loader and BSS Processor Data Processing System Bulletin, C28-6379. The names of the cards are listed below:

Card Name	Column 1 Punches
1. Absolute origin	(7, 8, 9)
2. Absolute flow	(7, 9)
3. Absolute branch	(6, 7, 9)
4. Absolute correction (C)	(12, 3)
5. Absolute patch (P)	(11, 7)
6. Absolute dump (D)	(12, 4)
7. Relocatable data	(6, 7, 8, 9)

8.	Relocatable instruction	(5, 7, 9)
9.	FORTRAN program	(5, 6, 7, 9)
10.	Common definition	(5, 7, 8, 9)
11.	FORTRAN Branch	(5, 6, 7, 8, 9)
12.	Relocatable correction (K)	(11, 2)
13.	Relocatable patch (A)	(12, 1)
14.	Relocatable dump (Z)	(0, 9)
15.	T	(0, 3)
16.	Super T	(0, 2, 3)
17.	Loader adjustment (O)	(11, 6)
18.	B	(12, 2)

PROGRAMMING USES

A problem program may issue a request to MCP to perform a specific function such as stack I/O interrupts, write an end-of-file mark, dump problem program core storage, and so on. These requests must be issued as pseudo-operations in the form of macro-instructions, or calling sequences. The macro-instructions take the general form:

MOP, parameter 1, parameter 2, etc.

where:

MOP represents the STRAP II mnemonic for the requested operation, prefixed by the character M. Any parameters required, including null parameters, are written in a specified order separated by commas.

The macro-instructions are expanded to calling sequences by the macro-generator (SMAC), in the general form:

B, $MCP
, $OP

followed, in consecutive half-words, by the specified parameters.

The calling sequences replace the macro-instructions, and are subsequently assembled by STRAP II along with the other symbolic statements. If SMAC is not in the compiling chain, pseudo-operations must be specified as calling sequences in the STRAP II form.

The first instruction in all pseudo-operation calling sequences is a B, $MCP, where $MCP is a STRAP II system symbol assembled as address 32. MCP always protects location 32., therefore, the attempted branch causes an instruction fetch (IF) interrupt. The interrupt handling routine within MCP examines the rest of the calling sequence to determine the requested pseudo-operation and to initiate the appropriate action. Thus, the programmer is cautioned that his program must be enabled, and that the IF mask bit must be one.

Symbolic I/O Operations

Operations on I/O units are initiated through pseudo-operations expressed either as macro-instructions or as calling sequences. MCP receives all pseudo-operations as calling sequence linkages.

MCP operates on logical units that are grouped on physical units by means of the problem program's IOD cards. Before problem program execution, MCP assigns an absolute channel and unit to each IOD. This information is also typed on the console typewriter, as a message to the operator, if operator intervention is required.

As part of this assignment, MCP stores the parameters appearing on the IOD cards in special tables. IODNAME is then used to form the necessary addresses to retrieve the parameters from these tables. IODNAME is a problem program symbol which appears in the name field of the appropriate IOD card, and refers to the requested logical unit via that IOD card. (See "IOD CARD".) Thus, MCP relieves the programmer of any concern over absolute units and channels.

The general form for I/O pseudo-operation macro-instructions is:

MOP, IODNAME (I)

where:

MOP represents the STRAP II mnemonic for the I/O operations prefixed by the character M.

I represents any index register, excluding index register zero, ($0).

The I/O macro-instructions are expanded to calling sequences in the general form:

B, $MCP
, $OP
, IODNAME (I)

The I/O pseudo-operation is expressed as a system symbol -- the STRAP II mnemonic prefixed by a $. IODNAME will be assembled by STRAP II as an integer, the I/O Reference Number. If a B, $MCP execution is attempted by the problem program, control will be given to a subroutine in MCP via the IF interrupt. IODNAME is used to locate the table of parameters specified on the appropriate IOD card. Thus, the operation write end of file might be written in the macro-instruction form:

MWEF, PPSYMB ($4)

which would be expanded to the calling sequence:

B, $MCP
, $WEF
, PPSYMB ($4)

where:

PPSYMB ($4) is the effective IOD reference number.

Symbolic I/O pseudo-operations are separated into two types:

1. Machine I/O pseudo-operations
2. Support I/O pseudo-operations

Machine I/O Pseudo-Operations

MCP, upon receiving control from a problem program calling sequence, issues the necessary hardware I/O instructions required to perform the requested operation. When the exchange has accepted the instruction and initiated the operation, MCP returns control to the problem program at the first instruction following the calling sequence. When the exchange signals the completion of the operation, MCP processes the interrupt and communicates it to the problem program. (See "I/O Interrupts.") This action preserves the asynchronous nature of I/O operations in the 7030 System.

If the requested channel is busy because another operation is already in progress on that channel, the instruction counter is frozen until the channel is free. When both the channel and unit are free, set-up of the requested operation is begun. This set-up may include:

1. Locating a unit on a multiple-unit channel.
2. Setting the mode and density as requested on the IOD card, or tape label.
3. Saving and checking the control word or arc number, if one is given.

In summary, it may be said that MCP supplies the necessary 7030 instructions to perform the requested I/O operation.

The read, write, and copy control word operations require the use of control words. The location of the first control word used by the operation should be specified symbolically in the macro-instruction as:

MOP, IODNAME (I), CTLWD (I)

where:

CTLWD is the problem program symbol for the first control word location, and I is any index register, excluding $0. (I indicates the use of an index register and not the register itself so that I for IODNAME (I) is any register and I for CTLWD (I) may be the same or any other register excluding $0.)

The control word chain indicated by CTLWD (I) must be in the problem program's core storage when the operation is started, and should not be altered until the operation is completed.

The macro-instruction MOP, IODNAME (I), CTLWD (I) is expanded to a calling sequence of the general form:

B, $MCP
, $OP
, IODNAME(I)
, CTLWD(I)
(return)

Not all I/O operations available in the 7030 instruction set may be actuated by means of macro-instructions, but the intent of these pseudo-operations, as used by the problem program, can be accomplished. For example, a locate tape pseudo-operation will result in an EPGK interrupt to the problem program, but MCP automatically locates the problem program tape for other tape usage pseudo-operations. A complete list of machine I/O pseudo-operations and the resultant interrupts are given in Appendix B.

I/O INTERRUPTS: The I/O table of exits, whose location is specified symbolically on the IOD card, is used by MCP to communicate I/O interrupts to the problem program. Each table is six full words in length. The first two words provide storage into which MCP can transmit information about the interrupt. The remaining words must contain instructions to handle the interrupt.

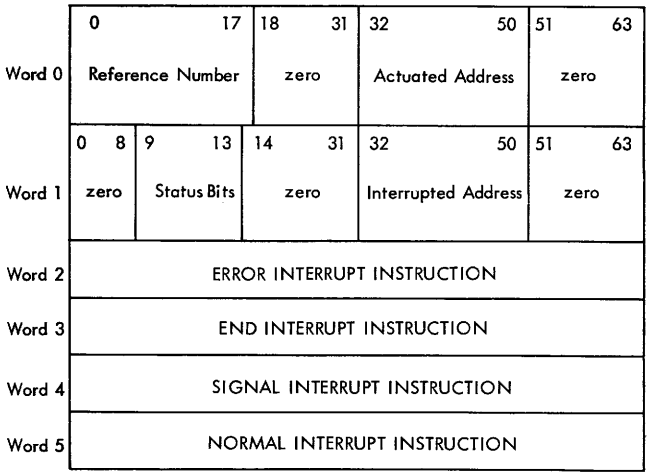
Every IOD has a table of exits associated with it; several IOD cards may, however, share a table of exits. Thus, if a program has N IOD cards, there may be as many as N or as few as one I/O tables of exits.

The programmer must construct each I/O table of exits himself. Each table must begin at a full word address. A simple method of constructing a table is to assemble the instruction

EXIT DRZ(N), 2

followed by one full-word or two half-word instructions for each of the four interrupt groups. EXIT is the name of the first word of the table that is currently associated with an IOD.

The format of each table is:



The unused bits of the first two words are reset to zero each time the table is used by MCP. The I/O reference number is that of the logical unit for which the interrupt occurred. The actuated address is the location of the macro-instruction (actually the location of the first instruction of the calling sequence) used to actuate the unit. The status bits contain all indicators turned on at the time of the interrupt, including the high-order indicator that directly caused the interrupt. The interrupted address is the location of the next instruction to be executed at the time of the interrupt. It is given for programmer reference only; MCP does not use it on return.

MCP gives control to the error interrupt instruction when the high-order status bit is either EPGK (.9) or UK (.10); to the end interrupt instruction when the high-order bit is EE (.11); to the signal interrupt instruction when CS (.13) is on alone; or to the normal interrupt instruction when the high-order bit is EOP (.12) with or without CS.

Normally, the last four words of an I/O table of exits will contain branch instructions that provide entry into fix-up routines. It is the responsibility of the fix-up routines to determine the combination of status bits that are on in order to correctly evaluate the interrupt and to provide appropriate corrective action. (Appendix B contains summaries of I/O status indicators with their interpretations for different types of I/O units.) Occasionally, some of these last four words will be one-word fix-up routines themselves.

I/O operations with suppress end of operation (SEOP) specified, function the same as non-SEOPed operations, except that interrupts are not issued for an end of operation (EOP) condition.

I/O OPERATING MODES: There are three modes of running in conjunction with I/O operations; only two are under problem program control.

Auto-Stacked Mode: The problem program is placed in the auto-stacked mode whenever an I/O interrupt occurs for that problem program during mainstream and control is passed to an I/O table of exits. This is known as an I/O fix-up.

The auto-stacked mode is a pseudo-disabled mode; the problem program is effectively disabled while MCP proceeds enabled. All additional problem program I/O interrupts are automatically stacked in an interrupt queue as they occur. This is done in the same manner as the exchange would stack them when in the disabled mode, except that MCP may still process MCP interrupts immediately.

Once the programmer has processed the initial I/O interrupt, there are several options for handling any remaining stacked I/O interrupts.

1. He may disregard all stacked I/O interrupts for a logical unit by issuing the Release pseudo-operation. (See Appendix B.)
2. He may selectively process the stacked interrupts, on a per unit basis, by issuing the Wait pseudo-operation. (See "Support I/O Pseudo-Operations.")
3. He may process the stacked interrupts on a first-come first-served basis, by issuing the Return pseudo-operation. (See "Support I/O Pseudo-Operations.")

The Return pseudo-operation is the only means of removing the problem program from the auto-stacked mode.

The problem program's lower registers, 3-31, are saved in protected storage upon entry into the auto-stacked mode. Whenever control is given to an I/O table of exits, the indicator register is zero; the mask register is zero except for the IF bit; index register 15 is zero; and the remaining lower registers contain insignificant information. Thus, when going from mainstream to auto-stacked mode, the problem programmer cannot depend upon his mainstream lower registers until he has returned to the mainstream mode via the Return pseudo-operation. The converse is also true. In the mainstream mode, he cannot depend upon the lower registers that he set up in the auto-stacked mode. If the programmer requires the contents of his mainstream registers while operating in the auto-stacked mode, a Fetch Lower Registers pseudo-operation may be issued. The registers will be placed into the program specified buffer and the auto-stacked fixup may then load any registers that it requires from the appropriate location within the buffer. (See "Fetch Lower Registers Pseudo-Operation.")

Mainstream Stack I/O (SIO) Mode: The second I/O operating mode causes all I/O interrupts to be stacked in a queue, as they occur, while the problem program is executing instructions in mainstream. The interrupts are not released to the problem program until requested by a Release I/O pseudo-operation or a Wait pseudo-operation.

Mainstream Mode: The third mode is the normal operating mode, where an I/O interrupt is released to the problem program immediately upon receipt, invoking the auto-stacked mode.

Support I/O Pseudo-Operations

In order to further facilitate machine I/O processing for the programmer, ten support I/O pseudo-operations are available, which do not require hardware I/O instructions, (except in the case of the Free tape drive pseudo-operation) and which, in

themselves, do not cause any I/O interrupts to be generated or communicated to the problem program.

Wait Pseudo-Operation ($WAIT): $WAIT is specified by the following macro-instruction:

MWAIT, IODNAME (I)

which is expanded to the calling sequence

B, $MCP
, $WAIT
, IODNAME (I)
(return)

It is used to freeze the instruction counter until an I/O interrupt occurs on a particular logical unit. The interpretation of $WAIT depends upon the status of the logical unit specified by IODNAME(I) and the mode in which the problem program is operating.

In the most common usage, an I/O operation is in progress on the logical unit specified by the $WAIT. $WAIT directs MCP to freeze the instruction counter until any I/O interrupt occurs on that logical unit. MCP immediately releases the interrupt to the problem program, a fix-up is executed, and control is returned to the mainstream to continue processing.

If an I/O interrupt is already stacked for the logical unit named by the $WAIT, the interrupt is immediately released to the problem program. If there is neither an I/O operation in progress nor an interrupt stacked on the logical unit specified, $WAIT is treated as a no-operation.

The effect of a $WAIT is also determined by the mode in which the problem program is operating. If the problem program is in neither the auto-stacked nor the stack I/O mode, other interrupts are released as they occur, but control is always returned to the $WAIT. If the problem program is in either the auto-stacked or the stack I/O mode, no other interrupts are released and the action of $WAIT does not alter the mode in effect. When the awaited interrupt is stacked on the specified logical unit, it is released by $WAIT regardless of the mode of the problem program, but no other stacked interrupts are released at this time.

If $WAIT is effected in the auto-stacked mode, a second fix-up is executed to process the awaited interrupt. If this fix-up is terminated by a return pseudo-operation and no other I/O interrupts are stacked, control is returned to the mainstream, not to the instruction after the auto-stacked $WAIT. If other interrupts are stacked, they are released and processed and control is then returned to the mainstream. A chart summarizing the action of $WAIT follows.

	Logical Unit		
Problem Program Mode	Operating. Awaited interrupt not stacked	Awaited interrupt already stacked	Not operating. Awaited interrupt not stacked
Mainstream	Hold mainstream IC until awaited interrupt occurs. Service others as they occur	Impossible	NOP
Mainstream SIO Mode	Hold mainstream IC until awaited interrupt occurs. Stack others	Release awaited interupt immediately	NOP
I/O Fix-up AS Mode	Hold fix-up IC until awaited interrupt occurs. Stack all others	Release awaited interrupt immediately	NOP

Return Pseudo-Operation ($RET): $RET is used to take the problem program out of the current I/O fix-up and, if no other interrupts require service, out of the auto-stacked mode. The macro-instruction

MRET

which is expanded to the calling sequence

B, $MCP
, $RET

indicates to MCP that the current I/O interrupt has been processed. Any other I/O interrupts that have been held in queue are released to the problem program, one at a time, in the order in which they were received and stacked by MCP. If there were no other interrupts in the queue, or if the queue has been exhausted, the lower registers are restored to their mainstream status, the problem program is taken out of the auto-stacked mode; and control is returned to the address in mainstream where the interrupt occurred.

$RET is the only way in which the problem program can be taken out of the auto-stacked mode, and the only way to have the lower registers restored by MCP. If $RET is used when the auto-stacked mode is not in effect, it is treated as a no-operation.

Stack I/O Pseudo-Operation ($SIO): $SIO is requested via the macro-instruction

MSIO

which is expanded to the calling sequence

```
B, $MCP
, $SIO
(return)
```

$SIO places the problem program in the stack I/O (SIO) mode, which causes all I/O interrupts to be stacked in a queue while the problem program is executing instructions in the mainstream. The stacked I/O interrupts are not released until requested by the problem program.

Release I/O Pseudo-Operation ($RIO): $RIO is the inverse of $SIO.

```
MRIO
```

is the macro-instruction and is expanded to the calling sequence

```
B, $MCP
, $RIO
(return)
```

This pseudo-operation tells MCP to begin unstacking, on a first in-first out basis, any I/O interrupts held in the stacked queue and to change the mode from mainstream SIO to normal mainstream. If no interrupts were stacked, the mode is simply changed. If there were stacked interrupts, they are now released, one at a time. As soon as the first I/O interrupt is released, and the problem program enters an I/O fix-up, the auto-stacked mode is invoked. After all I/O interrupts have been processed, control is returned to the location following $RIO in mainstream, provided the I/O fix-up (in auto-stacked mode) has issued a $RET. When $RIO is given in the mainstream, unstacking begins immediately. If $RIO is given in an I/O fix-up, the auto-stacked mode has priority and $RIO will not take effect until a $RET has been issued.

Change I/O Table of Exits Pseudo-Operation ($CHEX): $CHEX permits the programmer to change the table of exits address for a particular logical unit. The macro-instruction

```
MCHEX, IODNAME(I), NEXIT(I)
```

is expanded to the calling sequence

```
B, $MCP
, $CHEX
, IODNAME (I)
, NEXIT (I)
(return)
```

where:

NEXIT(I) is the problem program symbol locating a new table of exits to be associated with the IOD statement as specified by IODNAME (I). The problem program is given control at the return word in the linkage when the operation is completed.

Alter Tape IOD Disposition Pseudo-Operation ($ATID): $ATID allows the problem programmer to change the disposition of a tape reel during execution. The macro-instruction

```
MATID, IODNAME(I), NDISP
```

is expanded to the calling sequence

```
B, $MCP
, $ATID
, IODNAME(I)
, NDISP
(return)
```

where:

IODNAME (I) must specify a tape IOD statement, and NDISP represents the new disposition for the tape reels used in association with IODNAME (I). The NDISP must be specified in a full-word address form, selecting one of the following disposition codes:

```
NSAVE = 0.0          ISAVE = 2.0
CSAVE = 1.0           SAVE = 3.0
          PSAVE = 4.0
```

Fetch Lower Registers Pseudo-Operation ($FELR): $FELR and store lower registers pseudo-operations may be used by the programmer to examine the contents of the mainstream lower registers and to alter them if he chooses. $FELR is expressed in the macro-instruction

```
MFELR, FWA.(I)
```

and is expanded to the calling sequence

```
B, $MCP
, $FELR
, FWA.(I)
(return)
```

where:

FWA.(I) specifies the first word (18 bit) address of a 31 word buffer into which the contents of the problem program's lower registers will be placed. The fetched lower registers describe the status of the mainstream problem program. The problem program buffer will have the following contents after execution of $FELR:

FWA.(I)(.00-.17)	IOD Reference Number
FWA.(I)+1.	Boundary Register
FWA.(I)+2.	Maintenance Bits
FWA.(I)+3.	Channel Address
FWA.(I)+4.	Other CPU Bits
FWA.(I)+5.	Left Zeros Count and All Ones Count
FWA.(I)+6.	Left Half of Accumulator
FWA.(I)+7.	Right Half of Accumulator
FWA.(I)+8.	Accumulator Sign Byte
FWA.(I)+9.	Indicator register
FWA.(I)+10.	Mask register
FWA.(I)+11.	Remainder register

FWA. (I) + 12.	Factor register
FWA. (I) + 13.	Transit register
FWA. (I) + 14.-	
FWA. (I)+29.	Index Registers 0-15
FWA. (I) + 30.	Instruction Counter
FWA. (I) + 30. 32	Base Address of I/O Location Tables

An IOD reference number will appear in FWA. (I) if the current I/O fix-up is for a unit other than the one that the programmer is still expecting in his mainstream non-SIO $WAIT. The reference number will be the $WAIT reference number, otherwise FWA. (I) will be zero.

Store Lower Registers Pseudo-Operation ($STLR): $STLR allows the user to replace mainstream lower registers with the contents of a 31 word problem program buffer. $STLR may be used only in the auto-stacked mode. The macro-instruction

MSTLR, FWA. (I)

is expanded to the calling sequence

B, $MCP
, $STLR
, FWA. (I)
(return)

where:

FWA. (I) specifies the first word (18 bit) address of the 31 word buffer which will be used to replace the lower registers saved by MCP.

The following are restrictions on the use of $STLR:

1. Word 3, boundary control, is not changed.
2. The time signal (TS) and the execute exception (EXE) indicators are not changed.
3. The instruction fetch (IF) mask bit is forced to one.

The user's lower registers from the maintenance bits to index register 15, as well as the instruction counter, will be taken from the buffer in the same order as described under $FELR. The IOD Reference Number, Boundary Register and Base Address of the I/O Location Tables are not stored.

I/O Definition Pseudo-Operation ($IODEF): $IODEF allows the problem program to obtain in an IQS statement, the absolute channel and unit number associated with an IOD statement. The macro-instruction

MIODEF, IODNAME (I), ADDR. (I)

is expanded to the calling sequence

B, $MCP
, $IODEF
, IODNAME (I)
, ADDR. (I)
(return)

where:

ADDR. (I) is a full word (18 bit) address of the problem program storage location into which the full word IQS statement is placed. The IQS statement is in the following form:

CHXX UNY

Free Tape Drive Pseudo-Operation ($FREE): $FREE expressed by the macro-instruction

MFREE, IODNAME (I)

is expanded to the calling sequence

B, $MCP
, $FREE
, IODNAME (I)
(return)

$FREE indicates to MCP that this problem program no longer requires the tape unit specified by the IODNAME (I). MCP will unassign the unit from the problem program and give control to the return word in the calling sequence. After a $FREE is issued, the problem program can no longer use the tape drive specified by IODNAME (I).

A list of all the support I/O pseudo-operations is contained in Appendix A.

System I/O Pseudo-Operations

System Card Read Pseudo-Operation ($SCR): MCP provides a pseudo-operation that transmits to the programmer a specified number of card images from an input buffer maintained by MCP. The programmer specifies the number of cards to be read and the address of the first word of the core storage block into which they are to be read. If it is necessary to fill the buffer, MCP constructs the necessary I/O hardware instructions, initiates the reading operation, and handles any I/O interrupts that may occur. MCP returns control to the problem program when the transmission is terminated.

$SCR is a request to transmit a number of cards from the input buffer to a block of core storage within the problem program. For the purposes of $SCR, each card is assumed to be a column binary card image of 15 full words, 80 twelve-bit bytes. Each byte is a map of a card column where bit 0 corresponds to row 12, bit 1 to row 11 etc., and bit 11 to row 9. A one bit corresponds to a punch and a zero bit corresponds to no punch. Only complete cards may be transmitted to consecutive multiples of 15 words in the user's core storage area. $SCR is specified by a macro-instruction of the form:

MSCR, FWA. (I), N. (I), END

which is expanded to the calling sequence

B, MCP
, $SCR

, FWA. (I)
, N. (I)
, 0. (described as "M." following)
B, END' END RETURN
(normal return)

where:

FWA. (I) is the first word (18 bit) address of the buffer into which the cards are to be transmitted.

N. (I) is the number of cards to be transmitted. If there are at least N. (I) cards remaining to be transmitted to this job, 15N. (I) words are transmitted and the normal return is given.

END represents a problem program symbol for the location where the programmer wishes control returned if less than N. (I) cards are transmitted.

The macro-generator reserves a half-word for M., which will contain the number of cards that were actually transmitted if there were fewer than N. (I) cards remaining. If this situation occurs, the end return is given to the half-word preceding the normal return, where a branch instruction has been constructed by the macro-generator, using the programmer symbol END as the address.

The primary function of the MCP output program is to provide a standard method of output for both printing and punching. Output requests are made through two pseudo-operations rather than through I/O commands by the user. Buffering is used to output the data with a minimum of I/O interruption. The output program prepares a tape for processing on the IBM 1401 Data Processing System.

System Print Pseudo-Operation ($SPR): $SPR is a request to move a number of lines to the print buffer maintained by the output program. Each line consists of 17 full words, 136 eight-bit bytes. The first byte of each line must be one of the following characters, which control spacing before printing the line:

Function	Binary Representation	8-Bit BCD Code
Restore	00000001	1
Single Space	00100000	- (Minus)
Double Space	00100001	J
Triple Space	00100010	K
Quadruple Space	00100011	L

The 132 bytes which follow, must be 8-bit BCD (A8) characters to be printed. The remaining 3 bytes are used for padding and may contain any configuration of bits. Only complete lines may be moved from consecutive multiples of 17 full words in the user's core storage area. $SPR is specified through the macro-instruction

MSPR, FWA. (I), N. (I)

which is expanded to the calling sequence

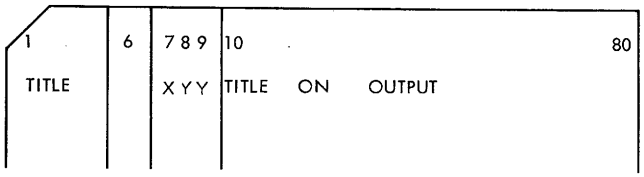
B, $MCP
, $SPR
, FWA. (I)
, N. (I)
(return)

where:

FWA. (I) is the first word address from which the lines are moved.

N. (I) is the number of lines to be moved. Information is not altered in the user's core storage.

In addition to performing the $SPR function, the Output program in MCP can insert a TITLE, a page number, and start a new page in the GO phase of a problem program as a TYPE card option. If TITLE is specified as an output option, the TYPE card must be followed immediately by a TITLE card of the following format:



Where X may be:

1 = insert a title if a restore character is encountered in a $SPR line. Substitute a double space character in the output buffer but not in the problem program's core storage for the $SPR line.

2 = insert a title and start a new page for every YY lines of problem program output (including the title line of output). If YY is blank, 55 will be used. A double space character will be used as in 1.

blank = insert a title for either a restore character (as in 1) in the problem program $SPR output or for every YY lines (as in 2).

System Punch Pseudo-Operation ($SPU): $SPU is a request to transmit a number of cards to the punch buffer maintained by the output program. Each card is a column binary card image of 15 full words, 80 twelve-bit bytes. Each byte is a map of a card column where bit 0 corresponds to row 12, bit 1 to row 11, etc., and bit 11 to row 9. A one bit corresponds to a punch and a zero bit corresponds to no punch. Only complete cards may be transmitted

from consecutive multiples of 15 full words in the user's core storage area. $SPU is written as a macro-instruction in the form

MSPU, FWA.(I), N.(I)

which is expanded to the calling sequence

B,$MCP
,$SPU
,FWA.(I)
,N.(I)
(return)

where:

FWA.(I) is the first word (18 bit) address from which the cards are to be transmitted. N.(I) is the number of cards to be transmitted. Information is not altered in the user's core storage.

Fetch Pseudo-Operation ($FETCH): $FETCH provides access to named system material on the disk. The requested material does not have to start at an arc boundary, and the amount of material to be transmitted is not limited to multiples of arcs. It eliminates the need for the programmer to provide core storage for an entire arc. The disk storage unit attached to the 7030 System is divided into logical sections by MCP. One section, the permanent read only storage area (PROSA), cannot be read or written by the problem program, but may be obtained via $FETCH. System material, such as mathematical subroutines, tables, and similar items, is maintained by MCP as a system library in PROSA.

PROSA is divided into two levels of storage, areas within types. An area normally contains a subroutine, or a set of tables, and several such areas are grouped into types. Each type-area has a unique name consisting of eight (tape) BCD characters in the following format:

TTAAAAAA

where:

TT, the first two characters, identify the type.
AAAAAA, the remaining six characters, identify the area within the type.

The composite name, type-area, must be unique. However, the same area name may be used with different type names. Type-areas are ordered on the disk in BCD code order.

A type-area is the largest logical unit of PROSA that can be fetched with one operation. $FETCH is a request to transmit a type-area, or part of a type-area, into the problem program core storage. The macro-instruction:

MFETCH, TTAAAAAA, RELFWA.(I), FWA.(I),
, N.(I), ERROR, END

is expanded to the calling sequence:

B,$MCP
,$FETCH
(AX)DD(BU,48,6),TTAAAAAAX
,RELFWA.(I)
,FWA.(I)
,N.(I)
,0. (described as "M." following)
B,ERROR' Error return
B,END' End return
(normal return)

where:

TTAAAAAA is an eight-character name of a specific entry in PROSA.
RELFWA.(I) specifies the first word within the type-area to be fetched, relative to the starting address of the type-area. The effective RELFWA.(I) must be an integer. RELFWA.(I) should be zero if fetching is to begin with the first word of the type-area, one if fetching is to begin with the second word of the type-area, and so on.
FWA.(I) specifies the first word (18 bit) address of the block of core storage into which the requested library material is to be transmitted.
N.(I) specifies the number of full words requested. If N.(I) is zero, fetch will read in all of the specified type-area from RELFWA.(I) to the end of the type-area.
ERROR is the address of a fix-up which the programmer wishes to execute if the fetch attempts to violate protected storage or when a nonexistent type-area is requested.
END is the address of a routine that the programmer wishes to execute in case the type-area is exhausted before the requested number of full words can be transmitted.
M. is a half word reserved by the macro-generator for the case when $FETCH cannot transmit N.(I) full words or when N.(I) = 0. In either event, M. will then contain the number of full words that were actually transmitted. The error return slot in the calling sequence consists of a branch instruction constructed by the macro-generator whose address is that specified by the programmer symbol ERROR in the macro-instruction. Similarly, the end return slot consists of a branch instruction whose address is the programmer symbol END.

Comment Pseudo-Operation ($COMM): $COMM is a request from the problem program to transmit a message from its core storage to MCP, to be output to the operator via the console typewriter. $COMM is specified in the macro-instruction

MCOMM, FWA.(I), N.(I)

which is expanded to the calling sequence

```
B,$MCP
,$COMM
,FWA.(I)
,N.(I)
(return)
```

where:

FWA.(I) is the first word address of a message to be typed.

N.(I) is the number of words in the message. The message to be typed must be a multiple of full words and cannot exceed ten. A message consists of IQS characters to be typed and need not include words for carriage control. Words for console display and end bytes must not be included.

Termination Pseudo-Operations

There are three pseudo-operations recognized by MCP that allow the problem program to terminate an entire job or a section of a job.

Resume Load Pseudo-Operation ($RESLD): $RESLD can be used by the programmer to load and execute his program in sections that may or may not overlap in core storage. To begin the loading of a new section, the problem program specifies the macro-instruction

```
MRESLD
```

which is expanded to the calling sequence

```
B,$MCP
,$RESLD
```

The loading is resumed with the next binary card, obtained from the same source as that of the previous load operation (system input or disk).

The loading is terminated when a branch card is encountered. Control is then given to the address specified on the branch card, where execution of that section begins.

The programmer must use caution when issuing a $RESLD in an I/O fix-up. The pseudo-operation does not remove the problem program from the auto-stacked mode and the next section of the job will be given control in the auto-stacked mode. In this case, the first Return pseudo-operation issued by this section would attempt to give control to a location within the first section of the job, which may no longer be in core storage.

End of Job Pseudo-Operation ($EOJ): $EOJ is the normal method used by the programmer to terminate his job. It is specified in the macro-instruction

```
MEOJ
```

which is expanded to the calling sequence

```
B,$MCP
,$EOJ
```

$EOJ signals the end of the current job to MCP, and under no circumstances will control be returned to this problem program. As a result of a $EOJ, MCP does the following:

1. Cancels the auto-stacked or stack I/O mode if either is in effect, and discards all stacked interrupts.
2. Treats the current job's I/O in the following manner:
   a. A release is issued if a channel is being read.
   b. A release is issued to a channel being written under control of a time delay (designated by the installation).
   c. Control operations are completed normally since the four tape control functions: Space Record, Space File, Backspace Record, and Backspace File, can only be released by operator intervention, and the rest of the control operations do not effect the status of the machine when allowed to terminate normally.
3. Unloads the tapes for the completed job under a priority scheme (discussed under tape dispositions in the "IOD Card" section), and informs the operator when it is necessary to save a tape.
4. Unassigns all of this problem program's I/O units, making them available for the next job to be run.
5. Completes any problem program output, and writes a tape mark on the systems output tape to indicate job output separation.
6. Prepares to process the next job.

Abnormal End of Job Pseudo-Operation ($ABEOJ): $ABEOJ is provided to permit the programmer to request the job terminating facilities of MCP when some abnormal condition is detected. $ABEOJ also provides additional, automatic facilities for checking purposes that are not provided under $EOJ. $ABEOJ may be expressed in the macro-instruction

```
MABEOJ
```

which is expanded to the calling sequence

```
B,$MCP
,$ABEOJ
```

The actions of MCP for $EOJ all apply to a $ABEOJ condition (except under $ABEX control), and in addition, an error dump will be taken of the problem program's core storage and of the associated I/O tables, if any. (See "Check-out Pseudo-Operations.")

Check-Out Pseudo-Operations

MCP provides four pseudo-operations that are useful to the programmer for debugging purposes, or for logging information automatically when an abnormal condition arises.

Problem Program Exit from Abnormal End of Job Pseudo-Operation ($ABEX): $ABEX allows the problem program to regain control in the event that MCP terminates the job for a problem program or machine error. The macro-instruction

MABEX

is expanded to the calling sequence

B,$MCP
,$ABEX
,FWA.(I)
(return)

where:

FWA.(I) is the first word (18 bit) address of a 32 word table, that includes a return slot, into which the following information will be placed:

FWA.(I)+0.00-0.17 contains the associated IOD reference number, if the problem program was terminated while in a mainstream wait status, for an I/O interruption that was not released to the problem program.

FWA.(I)+0.25 contains a one if the problem program was interrupted during an I/O fixup (the auto-stacked mode).

FWA.(I)+0.26 contains a one if the problem program was interrupted while in the stack I/O mode (SIO).

FWA.(I)+0.28-0.45 contains the error code indicating the cause of termination (described later).

FWA.(I)+0.46-0.63 contains the I/O interrupt queue count of the number of non-SEOPed I/O operations actuated by the problem program; the interrupts for these operations were not released to the problem program.

FWA.(I)+1.0-30.0 contains the problem program's lower registers, in the format of the $FELR buffer. (See "Fetch Lower Registers Pseudo-Operation.")

FWA.(I)+31.0 is the address where control will be returned to the problem program.

Note: This last word, FWA.(I)+31. must be set-up by the problem program.

At ABEOJ time, MCP checks to see if the current job had previously issued a $ABEX. If so, and if the abnormal job termination was neither a result of the problem program issuing a $ABEOJ nor a result of an improper use of $RESLD, MCP does the following: sets up the $ABEX table; clears the interrupt queue count and the interrupt queue of all previously stacked problem program interrupts; removes the problem program from any combination of auto-stacked mode, suppressed I/O mode, or wait status; clears the refill field of $15; clears the Indicator Register; resets the Mask Register to the MCP mask; and returns control to the problem program at location FWA.(I)+31. in the mainstream mode.

Each time a problem program receives control via $ABEX, a new $ABEX must be issued to regain control from the next abnormal job termination. Each $ABEX is in effect for one abnormal job termination only.

MCP will return control to the problem program N times per job, if the job is terminated due to a problem program error or a machine error. (N is specified by the installation.) However, control will be returned only once per job, when the operator terminates the job, regardless of the problem program's issuance of a new $ABEX.

The following is an example of the output, on the system output tape, from a job termination with $ABEX in effect:

1. A message indicating the cause of the abnormal job termination. Examples:
   a. ABNORMAL EOJ REQUESTED BY OPERATOR. (Error Code 0).
   b. JOB ENDED -- ERROR TYPE YY (Where YY is one of the error codes 1-64).
   c. XXXX INTERRUPT AT LOCATION 'location'. (Where XXXX is one of the indicators 0-3, 5-8, 15-17 or 19 which are represented by $ABEX error codes 65-76)
   d. A descriptive diagnostic message. (Error codes 77-80). (See Appendix E for error code descriptions)
2. The ABEOJ dump indicating the status of the machine at the time of job termination. (The dump is taken before the current $ABEX table is set by MCP). The problem program I/O will have been released at this time.
3. The message indicating the address where control was returned within the problem program (FWA.(I)+31.). The message is:

   YOU HAVE BEEN GIVEN CONTROL VIA $ABEX, AT LOCATION 'location'.

Hold Pseudo-Operation ($HOLD): $HOLD allows the programmer to halt his operations at the point where $HOLD occurs. It may be specified in the macro-instruction

MHOLD

which is expanded to the calling sequence

B,$MCP
,$HOLD

When $HOLD is encountered, the corresponding hold binary light (above binary switch 16), on the console is turned on. The contents of the instruction counter are displayed, in the octal format, in the right half of the console numeric display. (See "Checkout Facilities" for console options with a $HOLD.)

The only means of continuing the problem program after a $HOLD has taken effect, is via the operator's console in the following manner:

1. Set the start Key (Binary Key 48)
2. Press the Enter Key
3. When the Enter light comes on, press the End Key.

Dump Pseudo-Operation ($DUMP): $DUMP provides a means of performing breakpoint dumps of the problem program's core storage. The dump(s) may be taken in one of four formats, described indirectly by a parameter. The macro-instruction

MDUMP, ADDR(I)

is expanded to the calling sequence

```
B, $MCP
, $DUMP
, ADDR(I)
(return)
```

where:

ADDR (I) is the (19 bit) address of the first word in a string of dump requests, as illustrated by the following example.

```
ADDR    , A. (I)
        , B. (I)
        , FORMAT (I)
        .........
        , C. (I)
        , D. (I)
        NOP, FORMAT (I)
```

where:

A. (I) is the first location to be dumped

B. (I) is the last location to be dumped

and FORMAT must be one of the following:

0.0	Octal hex with mnemonics, panel
0.32	Octal hex with mnemonics, no panel
1.	Floating point with panel
1.32	Floating point without panel
2.	Index word with panel
2.32	Index word without panel
15.	Octal hex with panel, no mnemonics
15.32	Octal hex without panel, no mnemonics

More than one dump request may be specified with each $DUMP issued. The additional parameters immediately follow the first dump request parameters (C. (I), D. (I) in preceding example), until the requests are terminated by a NOP, FORMAT (I). The specified dump information will be written in the problem program file on the system output tape by MCP.

If an error is detected in a dump request, only the panel will be dumped. Possible errors are:

1. Illegal format
2. A. (I) ≥ B. (I)

Control is returned to the $DUMP calling sequence return slot after all requests have been processed.

Error Dump Pseudo-Operation ($EDUMP): An error dump is automatically executed by MCP whenever an abnormal end of job occurs. A pseudo-operation need not be supplied by the programmer. The error dump will include the entire problem program if no $EDUMP has been issued. The format of the dump is fixed as octal-hex with the panel, no mnemonics.

If the programmer wishes to change the error dump limits or format, he may issue a $EDUMP as a macro-instruction

MEDUMP, FWA. (I)

which is expanded to the calling sequence

```
B, $MCP
, $EDUMP
, FWA. (I)
(return)
```

where:

FWA. (I) is the first word (19 bit) address of a string of dump requests similar to that used by $DUMP. The dump descriptors at FWA. (I) replace those originally set up by MCP based on the problem program limits, or any descriptors previously set up by another $EDUMP.

A maximum of six dump requests is permitted for $EDUMP. If there is an error in the calling sequence, the $EDUMP will be ignored, and the previously set conditions will remain in effect.

$EDUMP does not cause a dump to be executed; it only sets parameters for the dump to be executed at ABEOJ time. If $EDUMP is accepted, and the parameters are set, control is returned to the return slot in the calling sequence.

Figure 1 illustrates and explains a typical panel dump as well as lines from a typical core storage dump in various possible formats.

The panel dump indicates the contents of the lower registers. A description of the registers can be found in the 7030 Data Processing System Reference Manual (A22-6530-2, pages 34-38). An identification of the registers from the abbreviations in the panel dump and a description of the format and interpretation of each follows. The registers are dumped in octal format unless otherwise specified.

LC	The Location Counter where the dump was taken.
IT	The value of the Interval Timer at the time of the dump.
UB	Upper Boundary Register.
LB	Lower Boundary Register.
LZC	Left Zeros Counter.
AOC	All Ones Counter.
FT	Factor Register.
AC	Accumulator. When interpreting the contents of the accumulator, ignore the first bit in the left most octal digit. In the example, the left half of the accumu-

```
LC 000114.4    IT 1156166    UB 000125    LB 000044    LZC 003    AOC 000    FT 0300000000037777777760
AC 000 00000013 56024713 56200000 00000000 00100340     SB 0100+010    RM 0000000000000000000160
   IND 00000000000000000000 10010000100000000000000000000 0000000000000100    MR   +F003  +.12000000000000 TUV
  MASK                      01000000000000000000000000000                      TR-16 0000000000000070    RIO, NOT AUTO-STACKING MODE

    X0 000000.00+ 0 000000 000000   X1 000051.00+ 0 000000 000000   X2 000000.00+ 0 000007 000000   X3 053517.00+ 7 000005 030071
    X4 000000.00+ 3 000145 000000   X5 000000.00+ 0 000000 000000   X6 000000.00+ 0 000000 000000   X7 000000.00+ 0 000000 000000
    X8 000000.00+ 0 000000 000000   X9 000000.00+ 0 000000 000000   XA 000010.00+ 0 000000 000000   XB 000011.00+ 0 000000 000000
    XC 000012.00+ 0 000000 000000   XD 000013.00+ 0 000000 000000   XE 000014.00+ 0 000000 000000   XF 000015.00+ 0 000000 000000

000044   + 001  +.10000000000000        + 002  +.10000000000000        + 003  +.10000000000000        +F003  +.12000000000000 TUV

000050   000000.00+ 0 567012 345671     053517.00+ 7 000005 030071     000000.00+ 0 567012 345671     006000.00+ 0 001777 777777

000054   000000.00 0B    560247.13 B9   053517.00 70    000120.60 39   000000.00 0B    560247.13 B9   006000.00 00    037777.77 FF

000060   053517.07 01    030071.07 03   000005.07 02    000023.31 80   003000.36 F0    000051.07 10   000053.00 80    000000.20 80
           LVI  ,3         LRI  ,3        LCI  ,3                       CM 1111         SX  ,3                         LFT   BU
```

FIGURE 1. DUMP EXAMPLE

lator contains the same number that is shown in locations 54 and 56 in the octal hex dump.

SB Accumulator Sign Byte Register. The first four bits represent the accumulator zone bits, the fifth bit is the accumulator sign, and the last three bits represent the accumulator flags.

RM Remainder Register.

IND Indicator Register. Each bit in the indicator register is represented. In Figure 1, indicators 20 (DF), 23 (PF), 28 (XPFP), and 61 (AE) were on at the time of the dump.

MR Multiplier Register. This register is dumped in floating point format.

MASK Mask Register. The portion of the mask register (indicators 20-47) that the programmer may set for those interrupts he wishes to handle. The IF bit (21) however, is always masked to 1 by MCP.

TR-16 Transit Register. This register is dumped in hexadecimal format.

RIO, NOT AUTO-STACKING MODE. This message may contain the word SIO instead of RIO and may say AUTO-STACKING MODE. (See "Support I/O Pseudo-Operations.")

X0 - XF Following the special registers, the index registers (0 - 15) are dumped in index word format.

Examples of the four dump formats are given after the panel dump.

Locations 44 - 47 were dumped in floating point format and illustrate respectively the numbers 1, 10, 100, and a number with exponent flag and the three data flags TUV.

Locations 50 - 53 were dumped in index word format. The number in word 52 was loaded into the left half of the accumulator and was still there when the dump was taken. This same number was stored into location 50.

Locations 54 - 57 contain the same information as locations 50 - 53 but were dumped in octal hex format.

Locations 60 - 63 were dumped in octal hex with mnemonics format.

Interrupt Classes

Interrupts are divided into the following three classes:

Class	Indicators
1. Error Interrupts	0-3, 5-8, 15-17, 19, 21
2. Input-Output Interrupts	9-13
3. Maskable Interrupts	4, 18, 20, 22-47

Error Interrupts

The handling of interrupts in Class 2 has been previously discussed. For interrupts in Classes 1 and 3, other than 0-3 and 5, if the error occurs at the problem program level, the identifying message is written on the output tape and $ABEOJ is issued.

Interrupts 0-3 and 5, designate equipment checks. It is not recommended to continue. However, if the interrupt is at the problem program level, the interrupt is identified and the operator is instructed to press the console Signal key if he wishes to continue. If the channel signal is given, a message is written on the output tape to identify the interrupt, $ABEOJ is given to the problem program, and the next job is run.

Any error interrupts at the MCP level (except 21, the IF interrupt) will result in a console message and a halt. The system must then be reinitialized.

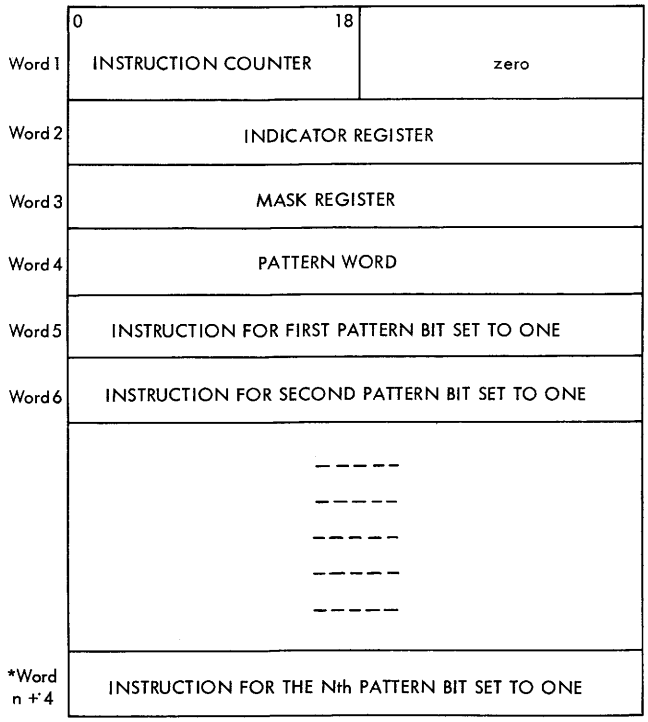
Maskable Interrupts: The problem program can exercise control over interrupts due to any problem program condition which causes the following indicators to be turned on:

4	Time Signal
18	Execute Exception
20	Data Fetch
22-34	Result Exceptions
35-38	Flaggings
39-40	Transits
41-47	Programs

Time signal and execute exception are permanently masked on, but to give the problem program more control, they are treated as pseudo-maskable. These, together with indicators 20-47, except for 21 (Instruction Fetch), are called the maskable interrupts.

Once a programmer indicates that he wishes to take a maskable interrupt (by setting the appropriate mask bit to one), he may then elect to use either a special fix-up routine that he provides within the problem program, or a standard fix-up routine provided by MCP. This choice is specified within a

problem program table of exits (PTOE), in the following format:



*N=The number of bits in the pattern set to one

The PTOE is a table provided by the programmer. The first three words of the table provide storage space in which MCP can save the instruction counter, indicator register, and mask register whenever the PTOE is used. The saved indicator register does not include the bit that caused the interrupt.

Pattern Word: The selection of a standard or special correction routine is made in the pattern word of the PTOE. In Word 4 of the PTOE, there is one bit which corresponds to each indicator in the maskable interrupt group. Each of these bits is in the same position of Word 4 as are the indicators in the indicator register. If the programmer wishes to use a special routine for a particular maskable interrupt, he sets to one, the bit in Word 4 that corresponds to that indicator. If he desires to use a standard routine, the appropriate bit in Word 4 is left as zero. If the programmer is willing to use only standard correction routines, no PTOE is required.

Word 5 of the PTOE contains the instruction related to the first bit in the pattern word (proceeding from left to right) that is set to one. This instruction will normally be a branch to the special routine provided by the programmer. Word 6 contains a branch to a special routine for the second Pattern bit set to one, and so on. One full word must be provided for each Pattern bit that is set to one.

Maskable Interrupt Linkages: If the programmer is supplying a PTOE, he must tell MCP where the table is located. This is done by placing the address of the first word of the PTOE in the refill field of Index Register 15 ($15) which is reserved for this purpose. The PTOE must begin at a full-word address, because the refill field is 18 bits long.

MCP initially sets the refill of $15 to zero. Therefore, whenever the problem program reaches a phase where the mask that is in effect permits maskable interrupts to be taken, and any special fix-ups are to be used, the initial location of the PTOE must be in the refill field of $15. If this field is zero, MCP assumes that no PTOE is provided.

Because the refill field of $15 is permanently unprotected, it may be addressed by any program at any time. The programmer can change the address of the PTOE several times during a single run in order to have different sets of special fix-ups available during different phases of the job. For example, if a programmer wishes to use a particular set of special fix-ups to handle any maskable interrupts that may occur while a special fix-up is already in progress, it is his responsibility to load the initial location of the new PTOE into the refill field of $15 at the appropriate time.

Standard Fix-Up Routines: Standard fix-up routines are provided by MCP to handle maskable interrupts when either the refill field of $15 is zero, or the corresponding problem program pattern bit is zero. The standard fix-up routines produce the following results for maskable interrupts:

Indicator Number	Symbol	Result
4	TS	No Operation
18	EXE	No Operation
20 and 22-47		A message is printed, through the output program giving the name of the interrupt and its location. If this is done more than 50 times, a $ABEOJ is given.

Note that MCP makes two permanently masked indicators appear to function as maskable indicators in the system -- indicator 4, Time Signal, (TS): and indicator 18, Execute Exception (EXE). Time signal causes a

completely asynchronous interrupt. If the programmer wishes to use time signal, he must turn on the appropriate pattern bit and also supply a special fix-up.

The occurrence of an execute exception interrupt can be anticipated more readily. It is the programmer's responsibility, therefore, to have a PTOE in effect that has the pattern bit for indicator 18 turned on whenever execute type instructions are being processed.

Restore After Maskable Pseudo-Operation ($RAM): At the conclusion of a special fix-up, the programmer may elect to restore the environment at the time of entry into the fix-up, or to have MCP perform the restoration. In the latter case, the programmer must provide the $RAM. The macro-instruction

```
MRAM
```

is expanded to the calling sequence

```
B,$MCP
,$RAM
```

When $RAM is encountered, MCP restores the indicator and mask registers from the PTOE, whose address is specified in the refill field of $15. MCP then executes a branch enabled to the address which was saved in the instruction counter slot of the same PTOE. The restored registers will be the same as they were when the maskable interrupt occurred, unless they were changed in the PTOE by the special fix-up routine. The lower registers, 3-31, except for the indicator and mask, will be in the same condition as when $RAM was given.

If the interval timer is to be used, the problem program must provide a special fix-up routine to handle the resultant TS interrupt and must use $RAM to exit from all maskable interrupt routines. Thus, $RAM serves to inform MCP when TS may be given safely to the problem program without destroying the condition of another current interrupt.

Fixup Pseudo-Operation ($FIXUP): $FIXUP is provided to facilitate the handling of maskable interrupts for indicators 20 and 22-47, excluding the pseudo-maskable indicators TS(4) and EXE(18). $FIXUP allows the programmer to insert a 64-bit word (one full word or two half word instructions) directly into the MCP interrupt table slot corresponding to the maskable indicator. The macro-instruction

```
MFIXUP, A(I), B(I)
```

is expanded to the calling sequence

```
B,$MCP
,$FIXUP
,A(I)
,B(I)
(return)
```

where:

A(I) may either be zero or may specify the first word (19 bit) address of a table of N + 1 words in problem program core storage. If A(I) is zero, it is ignored; if A(I) is a non-zero address, the following information will be stored into the table:

1. The original Pattern Word defined at B(I)
2. The current contents of each of the interrupt slots that are to be altered, unless they contain the standard MCP instruction, in which case, all zeros will be stored in place of the instruction.

B(I) specifies the first word (19 bit) address of a second table of N+1 words in problem program core storage. The table structure is as follows:

Word 1	PATTERN WORD (leftmost)
Word 2	Instruction for first pattern bit = one
Word 3	Instruction for second pattern bit = one
	-- -- --------

Word N + 1	Instruction for Nth pattern bit = one
N = the number of pattern bits set to one (20, 22-47)	

The Pattern Word specifies the maskable interrupt entries to be altered, by containing a one bit in the corresponding indicator position, e.g., to replace the zero divisor (ZD) interrupt table instruction, bit position 24 of the Pattern Word must contain a one.

For each pattern bit containing a one, there must be a 64 bit replacement word, in ascending order, in the table. This word may be set up as one of the following:

1. A full-word instruction, (e.g., VFL or SIC;B).
2. Two half-word instructions (e.g., two floating-point instructions or a half-word instruction followed by a NOP. However, only the first half-word instruction will be executed.

The following instructions are illegal replacements with $FIXUP and will cause job termination:

1. All I/O instructions.
2. All BD instructions.
3. All SIC; BD instructions.

Note: An MCP pseudo-operation will not cause job termination in $FIXUP, but will cause a TYPE 01 error when execution takes place in the interrupt table, since MCP depends upon the instruction counter to examine the calling sequence.

If the replacement word is all zeros, the standard MCP instruction will be restored into the interrupt table slot. This is the only method by which the

programmer can restore the original MCP interrupt table during execution. Control is passed to the return slot in the $FIXUP calling sequence when all replacements to and from the interrupt table have been completed.

Timing Operations

Reading the Time Clock Pseudo-Operation ($TIME): $TIME will read the time clock; adjust the reading by a predetermined calibration constant; convert the result into hours, minutes, and seconds; and transmit this information, in an edited form to a requested problem program location. The macro-instruction

```
MTIME, A. (I)
```

is expanded to the calling sequence

```
B, $MCP
, $TIME
, A. (I)
(return)
```

where:

A. (I) is the full word (18 bit) address to which the edited reading is transmitted. The edited reading will consist of eight IQS-coded characters and will occupy A. (I) in the format:

.00 - .15 Hours
.16 - .23 Colon
.24 - .39 Minutes
.40 - .47 Colon
.48 - .63 Seconds

Note that the following sequence of macro-instructions

```
MTIME, A. (I)
MCOMM, FWA. (I), 1.
```

will cause the current reading of the time clock to be typed on the console typewriter, assuming that A. (I) and FWA. (I) specify the same address.

Set Interval Timer Pseudo-Operation ($SIT): The interval timer measures elapsed time over relatively short intervals. It can be set to any time, and a TS interrupt occurs when the time period has ended. The value of the interval timer occupies bit positions 0-18 of word 1 in the 7030 special registers.

To simplify the problem of accessing the interval timer, which is permanently protected, MCP provides $SIT. Through use of $SIT, the programmer can transmit any 19-bit value from core storage to the interval timer. The macro-instruction

```
MSIT, A. (I)
```

is expanded to the calling sequence

```
B, $MCP
, $SIT
, A. (I)
(return)
```

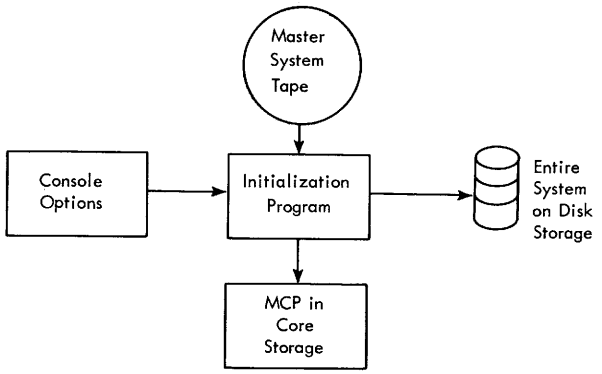
The 19-bit value at A. (I) (bits 0-18) is stored into the interval timer.

Note: To take advantage of the TS interrupt, the appropriate pattern bit must be on in the PTOE, and a special fix-up must be provided (See "Maskable Interrupts".) Control is passed to the return slot in the $SIT calling sequence, when the interval timer has been set and the timing begins immediately.

INITIALIZING THE SYSTEM

The 7030 Programming System must be initialized whenever the power has been turned on; the programming system or disk is suspected of being contaminated; or a machine error has made it impossible for MCP to continue.

At the time of initialization, the minimum machine configuration is assumed to be available. Basically, the initialization program reads the MCP portion of the master system tape into core storage and writes a copy of the entire tape on the disk in the PROSA area.



Input

The current MCP master system tape is file protected and mounted on any tape unit dialed to zero. Density for this unit is set to HIGH and the unit is made READY.

Procedure

1. Select initialization options as outlined in "Options" if an abnormal start of MCP is to be executed. Otherwise, binary keys 28, 29, 30, and 31 should be latched off and the operator should proceed immediately to step 2.
2. Set up the time in binary keys 0-23, and the date in binary keys 32-55. (See Figure 2.)
3. Press IPL key.
4. Press Signal key on the adapter unit for the channel on which the master system tape is mounted.
5. If the I/O status report option is desired, proceed as directed in step 3 under "I/O Status Report"; otherwise, omit this step and continue.
6. Mount the system tapes as directed by the mounting messages issued by the initialization program. (If key 28 is on, the IPL mounting messages are suppressed.)

Options

Any or all of the five options and their related parameters should be selected before pressing the IPL key.

Console Binary Key Latched On	Options
0-23	Time
32-55	Date
28	Suppress Messages
29	I/O Status Report
30	Rejected Job Count Report
31	Abnormal MCP Mode Report

Suppress Messages

If binary key 28 is latched on, all initialization program messages will be suppressed except the system tape "Date Verification Message" and the invalid Date Time message, if applicable. (See "Time and Date.")

I/O Status Report

To place a unit, or an entire channel, in the not available status, binary key 29 and the numeric switches are used, as explained in the following steps.

1. The unavailable device is identified in the numeric switches.

Numeric Switch Column	Contents
7 and 8	Channel number of the device.
15 and 16	Tape unit number. If the device is not a tape unit, these two columns are not used. If an entire tape channel is unavailable, put 08 in columns 15 and 16.

Note that the numbers on the numeric switch columns run from 1 to 16; there is no numeric switch column 0 as with the binary keys. (See Figure 2.) The contents of these switches will appear in the

numerical display for visual verification when the I/O Status option is used. A logging of the I/O status report will be given on the typewriter.

$ STATUS REPLY: CHXX ZZZ DELETED

where ZZZ is either UNY or ALL.

2. Return to step 3 of "Operation of the Initialization Program" to continue the initialization procedure. Note that since binary key 29 latched on indicates that at least one I/O status report is to be made, the initialization program must later inquire through a message whether there are additional I/O status reports to be made.

3. The following message indicates that the next I/O status report be made.

$ MAKE YOUR NEXT I/O STATUS REPORT.

4. Unlatch binary key 29 if there are no additional units to be made unavailable. If binary key 29 is left on, it will indicate that there is at least one more unavailable unit to be specified after the current one. Enter next channel and unit numbers in the numeric switches at this time.

5. Activate the sampling of the numeric switches and key 29 by pressing the console Signal key.

6. Return to step 6 of "Operation of the Initialization Program" if there are no more devices to be made unavailable. Otherwise, select the next I/O device to be made unavailable and return to step 3 above.

Rejected Job Count Report

In the off-line overlapped mode, key 30 when latched on allows MCP to start processing a problem program that is not the first job on the system input tape. The installation mode option (see "Abnormal MCP Mode Report") may be used to select the off-line overlapped mode if it is not the normal installation mode. When MCP is in control, the mode can be changed by the system commands provided for this purpose. (See "Commands.") The number of jobs to be rejected is inserted in columns 10 and 11 of the numeric switches.

Note: MCP will reject the number of jobs specified, but if the first off-line tape has less jobs than specified, MCP will reject as many jobs as is necessary from the subsequent off-line tapes to honor the count.

Abnormal MCP Mode Report

The installation system operating mode may be overridden at initialization time by latching on binary key 31 and specifying a mode code in numeric switch column 13. The installation's normal mode is established from the MCPP card (see "Updating the System") and this mode is selected if key 31 is off or if an invalid code is given.

Mode	Code
On-line overlapped	0
Off-line overlapped	1
Bypass	2
Mode from MCPP card	R, 3-9, blank

Time and Date

Binary keys 0-23 and 32-55 are used to enter the time and date. (See Figure 2.) Each number entered must be 9 or less to be valid. If the time and date entries are valid, they are saved by MCP, and the date and adjusted time are included on each problem program's output listing.

An invalid time or date entry will cause the following IPL message:

$ DATE/TIME ERROR. USE COMD, CLOCK

For the console response to a valid time and date setting, see the write-up for "COMD, CLOCK".

Messages

The following messages will be typed on the console typewriter:

Date Verification Message:

$DATE OF IPL TAPE XX MONTH YY DAY

where XX and YY are two-digit numbers. The dates should be visually verified by the operator to determine if the current master system tape has been used in the initialization.

I/O Assignment Messages:

$MCPIPL*MOUNT-REEL NUMBER 'reel'
CHANNEL XX UNIT Y

The preceding message specifies an I/O assignment for the system. Depending upon system requirements, other messages of this type may follow.

$READER IS INPUT SOURCE ON CHANNEL XX

In the on-line overlapped and bypass modes, a card reader is the system input source.

Error Messages

If an error condition is detected, the gong is sounded and a message stating the error condition is typed on the console typewriter. The error address is displayed in the accumulator lights on the maintenance console. (A complete list of error

messages for the initialization program is in Appendix D. under "Initialization Error Messages".) The instruction BD, $ will terminate the initialization error procedure.

RESTART - CONSOLE REINITIALIZATION

The Restart program permits the MCP system to renew itself, without recourse to the systems tape, if an error should occur while MCP is in core storage. Restart can write out the commentator and output buffers, execute a dump, and reinitialize MCP from the disk. Should the error be such that MCP cannot recover using Restart, MCP can be reinitialized from the disk. (See "Disk Initialization.")

Procedure

1. Latch on binary keys 12 and 27. This sets up a control word that is necessary when the console channel signal is given.
2. Select any combination of the four options, using binary keys 60, 61, 62, and 63.

Console Binary Key Latched On	Option
60	No check sum for disk IPL
61	Dump
62	Write output buffers
63	Write commentator buffer

For a detailed description of these options see "Restart Options."

3. Press the IPL key.
4. Press console Signal key.

The following two messages are always given when Restart begins:

$ READ FROM DISK OK.

This signifies that Restart bootstrap has successfully positioned Restart in core storage and given it control.

$ IPL OCCURRED DURING JOB "ppname" IN "system" MODE WITH IC = "location".

In this message, the condition of the system at the time of the Restart is given. The current job name, mode of operation, and instruction counter are included in the message.

5. Follow the directions printed by the various options selected in step 2. If keys 60-63 are not on, Restart goes directly to the Disk IPL program, checks PROSA for contamination, and renews the system.

Restart Options

Binary keys 60-63 are the four option keys. They should be set before the IPL and Signal.

Write Commentator Buffer

When binary key 63 is latched on, Restart will type the commentator buffer on the console typewriter.

Preceding the commentator output is the message:

$ BEGIN WRITE OF $COMM OUTPUT.

The terminating message is:

$ END WRITE OF $COMM OUTPUT.

If there is no output to be written within the buffer, this option is ignored and Restart proceeds to the next option.

Write Output Buffers

With binary key 62 latched on, the output program's print and punch buffers are written on the output tape. The first message with this option is:

$ BEGIN WRITE OF OUTPUT BUFFERS

If there is no output the next message is:

$ NO OUTPUT

The terminating message is always:

$ END BUFFER WRITE.

A number of conditions may exist concerning the output tape. If the tape is not available, a mounting message is given.

$ MNT AND RDY OUTPUT TP ON CHXX, UN0.

When the tape is mounted and made ready, the tape label will be spaced over and a message written on the tape to explain that it is a Restart output buffer write (OUTPUT AT IPL FOLLOWS. SOME OUTPUT MAY BE REPEATED...). Both print and punch buffers are written, two tape marks are written, and the tape is rewound and unloaded.

Should an EPGK condition occur, the message is:

$ PUT A RING IN TP NOW UNLOADING AND READY IT.

If EKJ or UK occurs, the output cannot be written and the message is:

$ XXX ON TP WRITE, OUTPUT SKIPPED.

XXX is either UK or EKJ.

Restart then gives the end buffer write message and continues.

Dump

Key 61 latched on, signifies a dump request. The message is:

$ OPTR PUSH CNSL CS FOR DUMP.

The dump limits and format are set in the numeric switches. The octal address of the lower limit is put in numeric switches, columns 1-8; the octal address of the upper limit is put in numeric switches, columns 9-15. Column 16 is reserved for the code that indicates the dump format as follows:

0 = octal hex with mnemonics
1 = floating point
2 = index word
3 = hexadecimal
R, 4-9 or blank = octal hex without mnemonics

The output device is selected by binary keys 44, 45, 46, 47 as shown by the following table:

Latched On	Output Unit Selected	
Key 44	Printer	
Key 45	CH 32, UN0	Unlabeled Tape
Key 46	CH 33, UN0	Unlabeled Tape
Key 47	CH 34, UN0	Unlabeled Tape

If keys 44-47 are all latched off, the dump is given on the printer.

Ready the tape or printer, set up the limits, and give channel signal from the console. Latch off key 61 after the channel signal is given for the last dump. The dump can be executed on several channels simultaneously. At the end of the last dump, a tape mark is written on all dump tapes, the tapes are rewound and unloaded and Restart continues. While the last dump is being given, the operator will have time to set up the IPL options (time, date, etc.).

Disk IPL

If key 60 is latched on, this means Restart will IPL from the disk without calculating the check sum of PROSA.
A message is given

$NO CHECK SUM

as a reminder that no check was made of the disk.

For a check sum calculation of PROSA in Restart, it is necessary only to keep key 60 off and have a type-area on the disk with the name ")))))))." (See explanation of Disk IPL with check sum, Procedure, 3. under "Disk Initialization.")

DISK INITIALIZATION

Note: In this section, the initials IPL represent the entire initial program loading procedure. A successful IPL involves placing MCP in core storage and writing a copy of the entire master system tape on the disk in the PROSA area.

The Disk IPL program can be used to reinitialize the system quickly, and eliminate the need for the master system tape once there has been a successful IPL.

Procedure

1. Press the IPL key.
2. Set up the IPL options as explained in "Initializing the System."
3. Put the binary deck of the Disk IPL program in the card reader.

The Disk IPL program has the ability to calculate a check sum of the disk PROSA area. The UPDATE30 program computes a check sum of the master system tape and places it in a type-area))))))., which is written on the new master source tape and is later included in PROSA. If there is no type-area))))))). on the disk, the Disk IPL program will reinitialize the system without checking the disk. A message will be printed:

$ NO CHECK SUM

The use of Disk IPL without the check sum is not recommended because the disk should be checked for contamination of PROSA.

4. Ready the card reader.

Error Situations

Disk IPL gives two error messages:

1. $ CHECKSUM INCORRECT
2. $ IPL THE TAPE

Both messages 1 and 2 are given if the check sum computed by Disk IPL does not agree with the one calculated by UPDATE 30 when the tape was made. This means that the disk is contaminated, so the program halts and the system must be rewritten on the disk by a tape IPL.

If any error interrupts occur, or the disk dictionary is incorrect, message 2 is given and the program halts. The system must be reinitialized by a tape IPL.

NORMAL RUNNING MODES

After successful initialization of the 7030 Programming System is completed, MCP is given control and begins to read in the first job to be run from the system input source. The MCP System Input Program is designed to:

1. Provide a standard method for buffering card input, regardless of the I/O device utilized by MCP.
2. Allow MCP to scan the input so that the jobs can be identified and controlled by type; the symbolic I/O requests can be assigned to absolute units, and system commands can be given by the operator via the card reader.

3. Minimize the handling of jobs by the operator.
4. Maximize the use of I/O units on-line.

Input is scanned to detect boundaries between jobs. When input is being read from tape, job boundaries are detected by tape marks. These tapes were originally prepared from card input, however, which did not provide such boundaries. Therefore, whenever cards are being read by the IBM 1401 for the off-line overlapped mode, or by the card reader when in either the on-line overlapped mode, or bypass mode, the following rule is used to detect job boundaries:

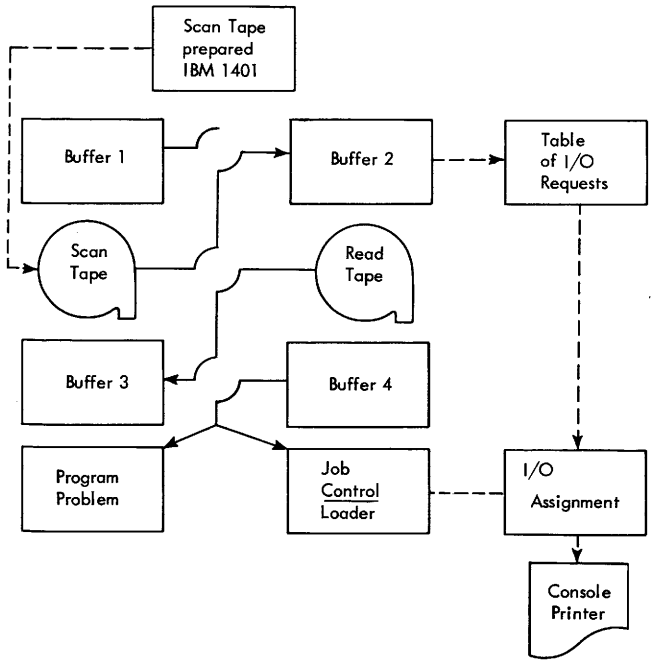
The first card of a job is that which is either a JOB card or a COMD card. A job extends up to the next JOB or COMD card.

The System Input Program can operate in one of three modes:

Off-line Overlapped
On-line Overlapped
Bypass

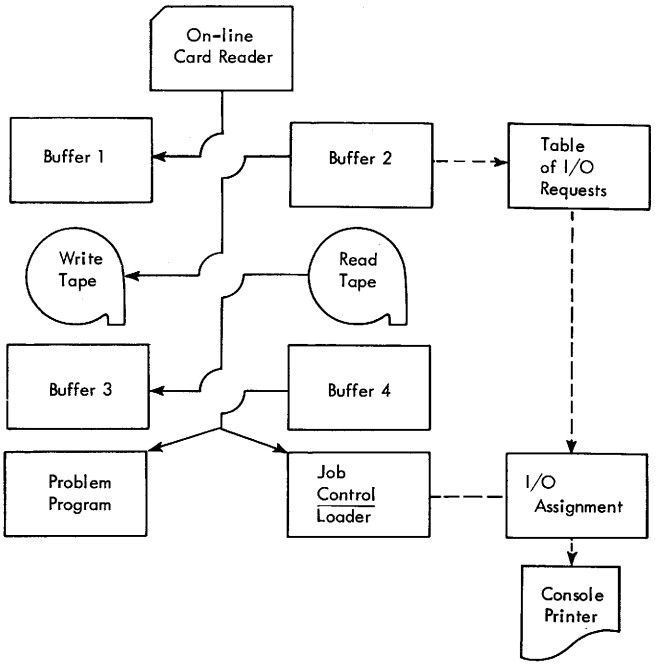
Off-line Overlapped

An input tape (SCAN tape) is prepared off-line by an IBM 1401. The scanning of this tape overlaps the running of jobs from a previously processed tape (READ tape). MCP allows a maximum of twenty jobs for each SCAN tape because of core storage limitations on the scanning tables.



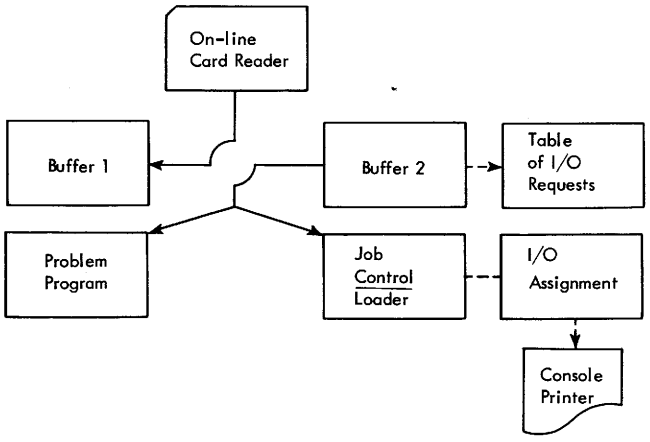
On-line Overlapped

An input tape (WRITE tape) is prepared on-line using a card reader. The scanning of the input and the writing of the tape overlaps the running of jobs from a previously processed tape (READ tape).



Bypass

Input from the card reader bypasses the overlapped tape, if any. Scanning of the input is done before running the job from the card reader buffer.



The mode of the input program affects the sequence in which the jobs are to be run and the sequence in which their I/O requirements will be assigned. However, programs can be written without regard to the mode in which the input program is operating: the method of giving cards to the problem program is the same for all modes.

After the problem program has been loaded by MCP, and at the point when control is transferred from MCP to the job, the programmer can assume that the following conditions will exist:

1. All registers from 4.0 through 31.63 are cleared to zero.
2. The 7030 is enabled.
3. The IF bit in the mask register (12.21) is set to one.
4. The boundary registers are set to the limits specified on the LIM card if the job was an absolute GO job or to the limits specified by the processor chain for this program.
5. No interrupts will be stacked and the job will be operating in the RIO mode.

CONSOLE USAGE

Under MCP, the console is a particularly active I/O device because of its flexibility of input. The operator can enter information without the delay of punching a card, generating a 1401 tape or, in general, preprocessing the input. For this reason, MCP includes routines to allow the operator to alter the current course of system or problem program action.

When the console is not being used as a system or symbolic I/O output device, the operator may use the "Check-Out" routine in MCP to assist the programmer in finding his errors. The operator may use the "Command" routine to alter system operation or he may use the console as a symbolic I/O problem program input device.

At any particular time, the entire facilities of the console must be available to either the problem program or to the system. This is achieved by distinguishing the manner in which the channel signal at the console is generated. A channel signal created by the Signal key (Figure 2) implies that console hardware belongs entirely to the programmer. If the Enter key is used to generate the channel signal, a typewritten message determines console ownership. If the mnemonic PP was typed, the console hardware and the following typewritten message belong to the problem program. In this case, MCP transfers control to the signal return slot in the problem program's I/O table of exits. For any other typewritten message, the information from the console belongs to MCP.

A null message, one in which the End key is depressed with no preceding typewritten message, places the console in the check-out mode. Any typed mnemonic other than PP would imply that the information from the console is to be handled by MCP. In any event, when the Enter key is used to generate a channel signal, the End key should not be depressed until the Enter light comes on. Otherwise, it will appear as if the channel signal were generated by the Signal key.

The rules for the typed mnemonics are:

1. Either upper or lower case characters may be used.
2. The carriage return, tabulate, and blank character codes appearing before the typed mnemonics are ignored.
3. The backspace character code is edited out of the message to be read.

Regardless of the cause of the channel signal from the console, the typewritten message can be no longer than ten words, or approximately one line. MCP always issues the hardware read that will accept a maximum of 13 words. When the subsequent pseudo-operation read occurs, MCP will transfer the console information as if a hardware read had just been issued. Depression of the Tabulate, Backspace and Carriage Return keys causes the appropriate character(s) to be sent as part of the ten-word message.

Check-Out

If the Enter-End key combination creates the channel signal, the console is read and the information is given to the MCP Check-Out program for interpretation. The End key should not be depressed until the Enter light comes on; otherwise it will seem as if the Signal key produced the channel signal interruption. When the console is in the check-out mode, the location of the next problem program instruction to be executed is displayed in the right half of the numeric display in octal.

Once the console has been placed in the check-out mode, several basic functions may be performed from the console. The functions use the binary keys, binary switches, binary lights, numeric switches and the numeric display.

Set Instruction Counter

The binary key, set instruction counter (16), starts or restarts the program from the console. This key enables the programmer to change the contents of the instruction counter which, arbitrarily alters the path of the program. When the key is active, the bit address is taken from the left bank of numeric switches; however, only the first 19 bits of the address will be used. This new address is also displayed in the right half of the numeric display when it is entered in the instruction counter.

Start

The binary key, start (48), is provided to begin the problem program from the console. When it is

sensed, it causes the problem program to start, beginning at the location currently in the instruction counter. If the set-instruction-counter binary key has not been used, the start key serves as a means of restarting the program from where it had been stopped. This key provides the only way of returning to the problem program once MCP has been placed in a check-out mode. Its action includes turning off the binary light hold (16).

Stop

Complementary to the problem of starting is the problem of stopping. One may randomly halt the execution of the problem program by depressing the binary key hold (47). When it is initiated, a corresponding binary light, labeled hold, is turned on and the contents of the next problem program instruction to be executed are displayed in octal in the right half of the numeric display. If both the start and hold binary keys are set, the binary light hold will be turned on and the problem program's instruction counter will be displayed; but the actions related to the start key will then occur. The primary function of the hold key is to randomly halt the problem program from the console without initiating any other action. If any other action were requested, then the hold key need not be set because it would be redundant. Stopping at a specific location can be accomplished by use of the pseudo-operation $HOLD. In any event, restarting is accomplished by means of the start key and the Enter key and End key combination.

Enter Information

The enter-from-binary-keys switch (17) enables the programmer to enter up to 64 bits into his program, beginning at the location specified in the left bank of numeric switches. The decimal number of bits, n, to be entered is placed right, justified in the right bank of numeric switches. Only the rightmost two columns of switches are sensed. If $64 < n < 99$, an error indication is given by means of binary light 18, to be labeled as the incorrect address, and no information will be entered. If $n = 0$, 64, or blank, a full word will be entered. Bits entered are taken from the binary keys in a left to right fashion.

Note: If $n = 103$, three bits of information will be entered with no error indication.

By the nature of the action requested, no further actions can be implemented through the binary keys. Therefore, the binary keys must be reset and a second channel signal must be generated. Meanwhile, the problem program will act as if a $HOLD had been issued.

Display Information

The display-on-binary-key-lights switch (16) enables the programmer to display 64 bits on the binary key lights, beginning at the bit address specified in the left bank of numeric switches. If any binary key is not in a neutral position when a display is requested, the action will not be carried out and binary light 17, to be labeled incompatible combination, will be turned on. If both the enter-from-binary-keys switch and the display-on-binary-key-lights switch are on at the same time, neither displaying nor entering may be accomplished. In this case too, the binary light incompatible combination will be turned on.

Dump

The programmer may request a dump from the console by setting the dump binary key (46). The range of the dump is specified in the numeric switches; the starting location is entered in the left bank of switches, and the last location to be dumped is placed in the right bank. If either address is partially blank; either address is invalid; either address is totally blank, or the left address is greater than the right address; then the binary light to be labeled incorrect address will be turned on and the dump will not be given.

The lower registers will always be included in the dump request unless the suppress lower registers binary switch is on. The formats are: octal hex, floating point, index word, or octal hex suppressing mnemonics.

To select one of the last three formats, the corresponding binary switch labeled:

FP FORMAT (13)
OHS FORMAT (14)
XW FORMAT (15)

must be selected. If none of the format switches is set, the format will be octal hex. If more than one format switch are set, the binary light labeled incompatible combination will be turned on and the dump will not be given.

EOJ

The binary key, end of job (EOJ) (58), enables the operator to end the current problem program without using the console typewriter. The operator should not have other check-out options indicated on the console because binary keys 58 and 59 will be interrogated first and action taken immediately. If the programmer wishes a check-out dump and EOJ, the operator should perform the dump function and then the EOJ function. Neither the hold nor display actions will occur for this console option.

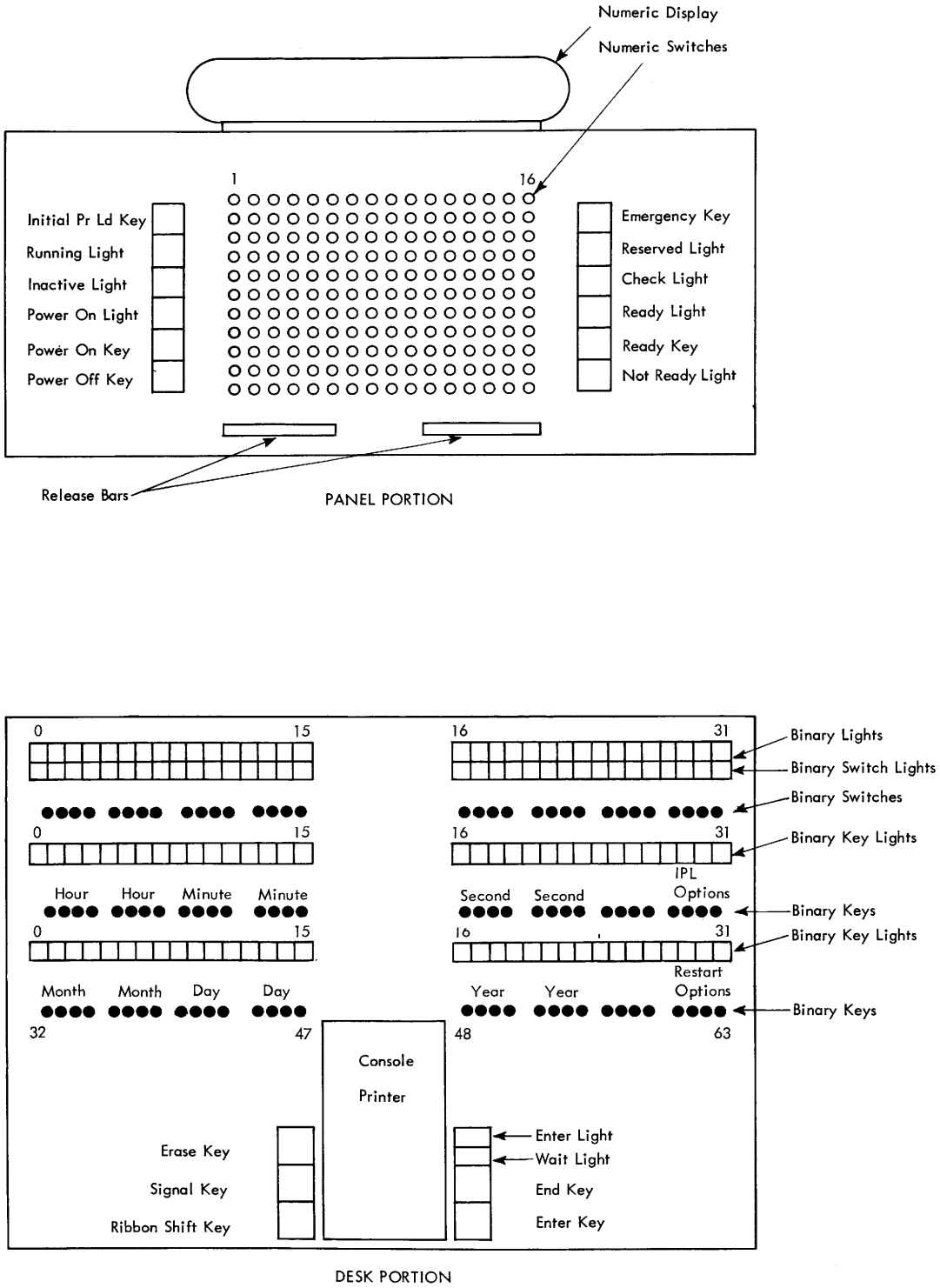


FIGURE 2. CONSOLE LAYOUT

IPL OPTIONS

Binary Keys:
0-23 Time
28 Suppress Messages
29 I/O Status Report (See "A" below)
30 Rejected Job Count (See "B" below)
31 Abnormal Mode Report (See "C" below)
32-55 Date

Numeric Switches:
A. Columns:
7-8 decimal channel number
15-16 decimal tape unit number
B. Columns:
10-11 number of jobs rejected
C. Column 13:
Mode Code: 0 = On-line overlapped
1 = Off-line overlapped
2 = Bypass

RESTART OPTIONS

Binary Keys:
12 and 27 Initial Control Word
44 Dump on Printer
45 Dump on CH32;UN0
46 Dump on CH33;UN0
47 Dump on CH34;UN0
60 No Check Sum
61 Dump (See Numeric Switches below)
62 Write Output Buffers
63 Write Commentator Buffer

Numeric Switches:
Columns:
1-8 Lower Dump Limit
9-15 Upper Dump Limit
16 Format:
0 = Octal Hex with Mnemonics
1 = Floating Point
2 = Index Word
3 = Hexadecimal
R, 4-9, blank = Octal Hex No Mnemonics

DEBUGGER OPTIONS

Binary Keys:
16 Set Instruction Counter
46 Dump (See Numeric Switches below)
47 Hold
48 Start
58 Eoj
59 Abeoj

Binary Switches:
11 Decimal Entry
12 Suppress Lower Register
13 Floating Point Dump
14 Octal Hex No Mnemonics Dump
15 Index Word Dump
16 Display On Binary Key Lights
17 Enter from Binary Keys

Binary Lights:
16 Hold
17 Incompatible Combination
18 Incorrect Address

Numeric Switches:
Columns:
1-8 Lower Dump Limit
9-16 Upper Dump Limit

FIGURE 3. CONSOLE USAGE SUMMARY

ABEOJ

The binary key, abnormal end of job (ABEOJ) (59), enables the operator to perform the ABEOJ function without using the console typewriter. The general information for EOJ will apply to ABEOJ.

Summary of Console Check-Out Facilities

A general rule applies to all console check-out facilities: whenever invalid settings or incompatible combinations of switches are encountered, the function will not be performed and an error indication will be given.

Numeric addresses entered via the numeric switches are always assumed to be in the octal radix. However, a decimal binary switch (11) has been provided which, when on, will cause address information being entered to be interpreted in the decimal radix. If entry of either 8 or 9 is attempted when the decimal switch is not active, no information will be entered and the binary light labeled incorrect address will be turned on.

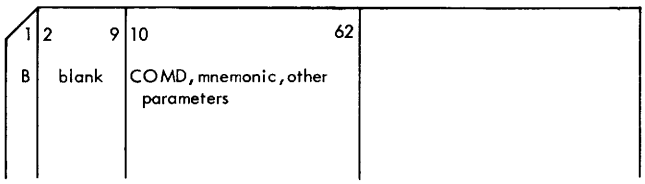
Figure 2 is a working drawing of the console. Figure 3 is a summary of the console functions and corresponding console settings.

Commands

System commands are used by the operator to direct MCP to perform specific tasks.

System commands may be entered through the operator's console or the system input source. In either case, the characters COMD, followed by a comma, must precede the mnemonic designation for the command.

There must be a B in column one of the command card if the source of the command is system input. The command card must be placed between jobs, never within a job. The command card is punched in the following format:



If the operator's console is the command source, press the Enter key on the console. When the Enter light comes on, type the information that would be punched in column 10-62 of a command card. No B is needed. When through typing, press the End key and wait for an acknowledgment of the command to be given on the console typewriter before attempting to enter another command.

The source of commands may make a great deal of difference in the time of execution of the command because of the asynchronism involved. Therefore, restrictions are placed on certain commands with regard to source. Following is a list of system commands and the sources to be used.

Commands That Change the System Operating Mode

The mode in which the system is operating remains in effect until another command is given to change it.

COMD, BYPASS: This command informs MCP that the next job is to be run in the bypass mode from the card reader as soon as the job currently running is completed. If the system is in the on-line overlapped mode, the command source must be the system card reader. If the system is operating in the off-line overlapped mode, the command source must be the operator's console.

COMD, ONLINE: This command informs MCP that the next job in the system card reader is to be written on the write tape as soon as the tape is available. The next job to be run comes from the read tape. If the system is in the bypass mode, the command source must be the system card reader. If the system is in the off-line mode, the command source is the console.

COMD, OFFLINE: This command informs MCP that the next job to be run will be read from the read tape. Regardless of the current operating mode, the command source must be the last card in the system card reader.

Commands That Affect The Current Input

COMD, REWIND: This command informs MCP, if in the on-line overlapped mode, to terminate and rewind the write tape. The command source must be the system card reader.

COMD, EOF: This command marks the end of the previous job in the system card reader. The command source must be the system card reader and the card must be the last one in the reader.

Commands That Affect The Current Job

COMD, REJECT: This command informs MCP, if in either of the overlapped modes, to reject the job immediately preceding the reject command. The command source is always the system input source.

COMD, EOJ: This command terminates the job being executed. The operator's console or system card reader may be the source of the command.

COMD, ABEOJ: This command terminates and dumps the current job. The command source may be the operator's console or the system card reader.

COMD, OUTPUT: This command informs MCP that the current job being run is to be the last job on the current system output tape. The output for the next job will be the first file on the new system output tape. Either command source is acceptable.

Other Commands

COMD, IOCHANGE, Channel, Unit, Code, Type: This command is used to alter the availability of I/O units and channels to the 7030 Programming System. This may result in reassignment of MCP units and/or termination of the presently operating problem program. Either command source is acceptable.

Channel is the decimal number of the channel.

Unit is the decimal number of the unit (0-7). If an entire channel is being altered, ALL should be used as the unit parameter.

Code may be one of the following:

ADD Make channel or unit available.

DELETE Make channel or unit unavailable. The command is effective if and when the channel or unit is not operating.

DELETM Make channel or unit unavailable only if unassigned at the present time. If the channel or unit is assigned, the command is ignored.

Type is used only if the Code is ADD. It must be one of the following:

READER PRINTER PUNCH

If the Type field is blank, it means no change in equipment type.

COMD, CLOCK, hours, minutes, seconds, mm/dd/yy: This command gives MCP the real time in the three 2-digit parameters shown, which are used to compute the calibration constant for the time clock. The date field also contains three 2-digit parameters separated by / and not by commas.

mm is the 2-digit month or day

dd is the 2-digit month or day

yy is the last two digits of the year.

The date and time are written on the output tape and on the console typewriter. The time can be entered without a date.

Response to the clock command is acknowledged by the following message.

$HH:MM:SS - TIME-HH:MM:SS DATE - mm/dd/yy

The time on the left is before calibration; the time on the right is the recalibrated time. If the time entered is zero, the time clock is not reset to zero but the present time clock setting is used; therefore, the time before and after is the same. This allows a date to be entered without changing the time clock calibration. The console, system input, or IPL program can be the source of this command.

COMD, COMMENT, maximum 40 character message: This command is used for communication between the operator and the programmer. The operator's console or system input may be the source. The comment will be written on the console typewriter and on the output tape preceding the job card.

OFF-LINE OPERATIONS

MCP is designed to operate with labelled input and output tapes. The normal 1401 programs, to process input and output, expect a label file. If MCP is altered to operate unlabelled, then unlabelled versions of the 1401 programs must be used and all comments about labels in the following section can be ignored.

Output Tape Construction

The output tape is written by the output program to be processed by an off-line IBM 1401 Data Processing System.

The tape is written at high density with odd parity in the NO-ECC mode. Odd parity is used to accommodate both column binary cards and BCD lines. The tape contains a label file followed by a separate file for each problem program. The last file on the tape is followed by a double tape mark and a trailer record.

Problem Program File

Each file contains at least one print record and one punch record. Within the same file, these different records are distinguished by their first character.

Print Record: A print record is written as a continuous string of six-bit characters. The first character is the identifying character P. Each successive string of 133 characters contains a form control character and 132 characters to be printed as a line. The last line in the record is followed by padding characters because of the relationship of the 64-bit word and the six-bit frame.

A 12-line print record is written with a word count of 150, which produces 1600 six-bit characters:

1	P $(100111)_2$
2	Form control, line 1
3	Print position 1, line 1
...	...
134	Print position 132, line 1
135	Form control, line 2
...	...
1597	Print position 132, line 12
1598	Padding character
1599	Padding character
1600	Padding character

The form control character for each line must be one of the following:

$(000001)_2$	Restore form
$(100000)_2$	Single space
$(100001)_2$	Double space
$(100010)_2$	Triple space
$(100011)_2$	Quadruple space

These space-before-printing characters, except for the single space, are used as d-characters for the carriage control instruction on the IBM 1401. The single space character is ignored, because the IBM 1403 Printer always spaces after each line is printed.

The padding character used is the BCD blank $(010000)_2$. Unused portions of the last print record, if any, are filled with blank lines.

Punch Record: A punch record is written as a continuous string of six-bit characters. The first character is the identifying character C. Each succeeding pair of characters corresponds to a card column so that successive strings of 160 characters correspond to 80 columns to be punched. The last card in the record is followed by padding characters.

A ten-card punch record is written with a word count of 151, which produces 1611 six-bit characters:

1	C $(110011)_2$
2	Rows 12-3, column 1, card 1
3	Rows 4-9, column 1, card 1
...	...
161	Rows 4-9, column 80, card 1
162	Rows 12-3, column 1, card 1
...	...
1601	Rows 4-9, column 80, card 10
1602	Padding character
...	...
1611	Padding character

The padding character used is the unpunched column $(000000)_2$. Unused portions of the last punch record, if any, are filled with padding characters.

Trailer Record: The trailer record indicates whether the last file on the output tape was terminated physically and logically, or physically, only. In the latter case, the last file will be continued on the next output tape. The trailer record

consists of 11 characters of which only the first is used:

$(000000)_2$ physical and logical end
$(000001)_2$ physical end only

Processing Output Tapes

Mount the tape on unit 2 and set density to high. Load the punch with blank 7030 binary cards. Set printer for 132-character print-out. Set sense switches and I/O switch off.

Load the output program with blank cards following. These blank cards should be of a different color or corner cut from those on the punch side. These cards will be used to separate jobs on the punch side.

Error comments and operator instructions will be printed on the right side of the page.

The label file is skipped by the output program before any printing or punching is started.

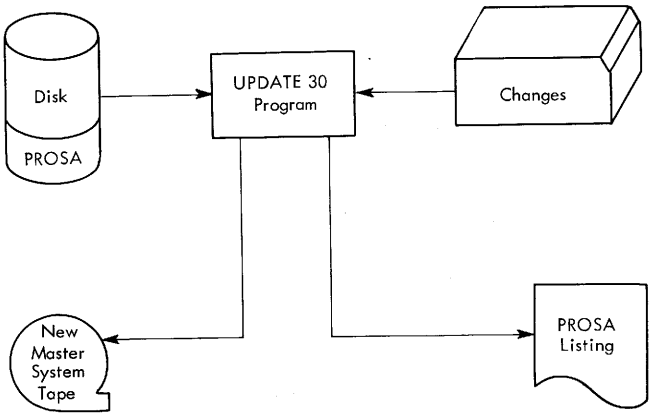
Format of the Input Tape

The input tape, written by the IBM 1401 or by the system input program when in the on-line overlapped mode, is a labeled tape (the label is the first file). The tape is multifile, with each file corresponding to a job and with a double tape mark terminating the last job. Each file consists of one or more records. Each record, except the last one, consists of 17 column binary card images (160 characters each). The last record of each file may contain 17 or fewer card images.

Job boundaries are determined by a card with a B in column 1 and either a COMD or a JOB, in columns 10-13. All COMD cards become the first cards in the next job's file. Any COMD cards on the end of an input deck will be placed in a file of their own, followed by two file marks.

UPDATING THE SYSTEM

A program called UPDATE 30 is available to facilitate the incorporation of changes into the current system located in the PROSA area of the disk. For a complete writeup of the UPDATE 30 program, see page 215 of the MCP Programming Systems Analysis Guide (C22-6728), or the bulletin published on the UPDATE 30 program (C22-6718).



Initial Preparation of the MCP Deck for Updating

Upon receipt of the MCP assembly output tape, the following directions should be followed to prepare the MCP deck for the initial update run.

A. List and punch the MCP assembly tape. The deck after punching will contain these types of cards in the following order:

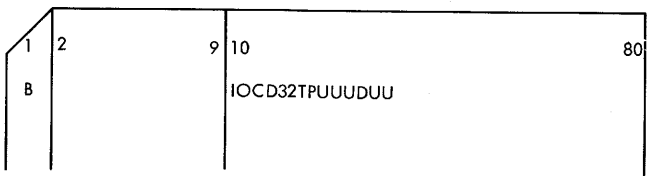
1. JOB card
2. TYPE card
3. LIM card
4. IOD cards
5. HED card MCP Bootstrap Type Area
6. Binary cards with PUNID 11E11BSP
7. HED card IPL Type Area
8. Binary cards with PUNID 11C11IPL
9. HED card Restart Type Area
10. Binary cards with PUNID 33RESTRT
11. HED card MCP Mainleg Type Area
12. Binary cards with PUNID 11D11MCP
13. HED card MCP Loader Type Area
14. Binary cards with PUNID 22LOADER
15. HED card MCP Dump Type Area
16. Binary cards with PUNID 22$DUMP
17. HED card MCP System Commands Type Area
18. Binary cards with PUNID 22SCOMD
19. HED card MCP Job Control Type Area
20. Binary cards with PUNID 22$EOJ
21. BRANCH card.

B. Interpret all B cards

C. Discard:
JOB card
TYPE card
LIM card

The four IOD cards that have instructions on them to "pull card before updating."
BRANCH card

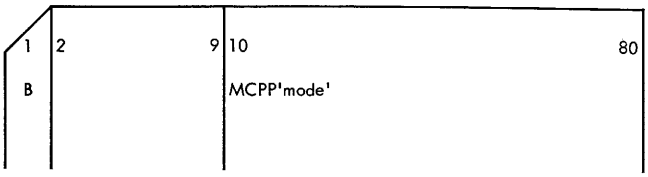
D. Each installation must make up IOCD cards. There must be an IOCD card for each channel used by the system. The IOCD card gives the channel number, the type of equipment attached to the channel, the availability of the channel, and the number of units attached to the channel and their status. The format of a typical IOCD card is as follows:



Punch a B in column 1, IOCD in columns 10-13, the channel number in columns 14 and 15, and the type of equipment attached to the channel in columns 16 and 17 using the appropriate codes: DK for disk, CN for console; PU for punch; PR for printer; RD for reader; TP for tape.

The initial state of the channel is given in column 18; punch a D for down or a U for up. The status of each unit on the channel is given, starting in column 19. Punch a D for every down unit and a U for every up unit on the channel. The order of the Ds and Us is important as columns 19 through 27 represent units 0-7, respectively. A single unit channel needs only a D or a U punched in column 18. The status of channels and units can be altered during the running of the MCP program by using the IOCHANGE command. Place the IOCD cards before the IOD cards.

E. The MCPP card must be made by the installation. The MCPP card determines the normal mode that the IPL program will leave in control after it has initialized the system. This mode can be overridden at IPL time (see "Abnormal MCP Mode Report") or after IPL by issuing a system command. The format of the MCPP card is as follows:



Where 'MODE' is one of the following: BYPASS, ONLINE, OFFLINE. Place the MCPP card after the IOD cards.

F. Place the IOCD cards, IOD cards, and MCPP card in that order, after the IPL binary deck.

G. Insert any C and P cards after the binary deck of the Type Area to which they pertain. It is advised that C cards be placed before P cards in order to eliminate possible errors should a P card reference the same location as a C card. Any corrections to the IPL program should be placed after the MCPP card. It is not recommended to use P cards in IPL as they are loaded above the IPL upper limit where the IOCD and IOD cards are located.

H. Be certain that each Type Area of MCP is preceded by the proper HED card.

I. The deck is now ready for updating the system with the UPDATE 30 program. On subsequent updates, once the deck is prepared, it is only necessary to update those Type Areas that have changes.

APPENDIX A. MCP PSEUDO-OPERATION CODES

The following list specifies mnemonic and absolute codes for MCP pseudo-operations that are available for use by the problem program. All have DDS of (BU, 24).

$RD	1.0
$RDS	1.01
$W	1.32
$WS	1.33
$CCW	2.0
$REL	2.32
$RELS	2.33
$LOC	3.0
$LOCS	3.01
$FC	3.32
$FCS	3.33
$TIF	4.0
$TIFS	4.01
$ERG	5.0
$ERGS	5.01
$SP	5.32
$SPS	5.33
$BSP	6.0
$BSPS	6.01
$SPFL	6.32
$SPFLS	6.33
$BSFL	7.0
$BSFLS	7.01
$WEF	7.32
$WEFS	7.33
$REW	8.0
$REWS	8.01
$UNLD	8.32
$UNLDS	8.33
$RLF	9.0
$RLFS	9.01
$RLN	9.32
$RLNS	9.33
$KLN	10.00
$KLNS	10.01
$FREE	10.32
$GONG	11.0
$GONGS	11.01
$WAIT	11.32
$CHEX	12.0
$IODEF	13.0
$HD	14.0
$HDS	14.01
$LD	14.32
$LDS	14.33
$EVEN	15.0
$EVENS	15.01
$ODD	15.32
$ODDS	15.33
$ECC	16.0
$ECCS	16.01
$NOECC	16.32
$NOECCS	16.33
$ATID	17.0
$SIO	32.0
$RIO	32.32
$RET	33.0
$RAM	33.32
$STRG	34.0
$FECRG	34.32
$TIME	35.0
$COMM	35.32
$SIT	36.0
$FIXUP	36.32
$STLR	37.0
$FELR	37.32
$ABEX	38.0
$DUMP	64.0
$EDUMP	64.32
$EOJ	65.0
$HOLD	65.32
$RESLD	66.0
$FETCH	66.32
$SPU	67.0
$SPR	67.32
$ABEOJ	68.0
$SCR	68.32

The I/O operations are listed in this appendix in both their macro-instructions and in the form of the expanded calling sequence. The status bits that may be turned on in the I/O table of exits, as a result of each operation, are also listed. With the exception of the pair EPGK and EOP, multiple bits may be turned on, and this can be interpreted as a combination of individual results. Multiple bits are listed which, by their special combination, identify a unique result. CS can be turned on independently of any operation. It may be caused by pressing the Signal key or by readying the unit.

Read

MRD or MRDS, IODNAME(I), CTLWD. (J)
B, $MCP
, $RD or $RDS
, IODNAME(I)
, CTLWD. (J)
(Return)

Note: A maximum of 13 words can be read when IODNAME(I) defines a console.

EPGK	(Punch or Printer) Invalid operation
EPGK	(Disk) Operation would exceed requested arcs
EPGK	(Tape) First operation to scratch tape
UK	(Disk, Tape, Console, or Reader) Unit error
UK, EOP	(Disk, Tape, Console, or Reader) Data error
EE	(Reader) Out of material or stacker full
EE	(Tape) Crossed tape mark
EOP	$RD completed
CS	Unit readied for Reader
None	$RDS completed

Write

MW or MWS, IODNAME(I), CTLWD. (J)
B, $MCP
, $W or $WS
, IODNAME(I)
, CTLWD. (J)
(Return)

EPGK	(Reader) Invalid operation
EPGK	(Disk) Operation would exceed requested arcs
EPGK	(Tape) Write on file protected tape
UK	(Disk, Tape, Console, Punch, or Printer) Unit error
UK, EOP	(Disk, Tape, Console, Punch, or Printer) Data error
EE	(Printer) Out of material
EE	(Tape) Crossed end mark
EE	(Punch) Out of material or stacker full
EOP	$W completed
CS	(Punch or Printer) Unit readied
None	$WS completed

Copy Control Word

MCCW, IODNAME(I), CTLWD. (J)

B, $MCP
, $CCW
, IODNAME(I)
, CTLWD. (J)
(Return)

Release

MREL or MRELS, IODNAME(I)
B, $MCP
, $REL or $RELS
, IODNAME(I)
(Return)

EOP	$REL completed
None	$RELS completed

Locate Arc

MLOC or MLOCS, IODNAME(I), ARC. (J)
B, $MCP
, $LOC or $LOCS
, IODNAME(I)
, ARC. (J)
(Return)

EPGK	(Tape, Console, Reader, Punch, or Printer) Invalid operation
EPGK	(Disk) The arc number, ARC. (J), exceeds the number requested for IODNAME(I).
CS	$LOC completed
None	$LOCS completed

Feed Card

MFC or MFCS, IODNAME(I)

B, $MCP
, $FC or $FCS
, IODNAME(I)
(Return)

EPGK	(Disk, Tape, Console, Reader, or Printer) Invalid operation
UK	Feed jam or punch failure
UK, EOP	Punching error in last card in stacker
EE	Out of material or stacker full
EOP	$FC completed
None	$FCS completed

Tape Indicator Off

MTIF or MTIFS, IODNAME(I)

B, $MCP
, $TIF or $TIFS
, IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch, or Printer) Invalid operation
UK	Repeated failure of operation
EOP	$TIF completed
None	$TIFS completed

Erase Long Gap

MERG or MERGS, IODNAME(I)
B, $MCP
, $ERG or $ERGS
, IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch, or Printer) Invalid operation
EOP	$ERG completed
None	$ERGS completed

Space Record

MSP or MSPS, IODNAME(I)
B, $MCP
, $SP or $SPS
, IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch, or Printer) Invalid operation
EPGK	(Tape) First operation on a scratch tape
UK	Repeated failure of operation or tape at physical end
EE, EOP	Operation completed by spacing across tape mark
EOP	$SP completed
None	$SPS completed

Backspace Record

MBSP or MBSPS, IODNAME(I)

B, $MCP
, $BSP or $BSPS
, IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch, or Printer) Invalid operation
EPGK	(Tape) Tape at load point
UK	Repeated failure of operation
EE, EOP	Operation completed by backspacing across tape mark
EOP	$BSP completed
None	$BSPS completed

Space File

MSPFL, IODNAME(I)
B, $MCP
, $SPFL
, IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch, or Printer) Invalid operation
EPGK	(Tape) First operation on a scratch tape
UK	Repeated failure of operation or tape at physical end
EE, EOP	Operation completed

Backspace File

MBSFL, IODNAME(I)
B, $MCP
, $BSFL
, IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch, or Printer) Invalid operation
EPGK	(Tape) Tape at load point
UK	Repeated failure of operation
EE, EOP	Operation completed

Write Tape Mark

MWEF or MWEFS, IODNAME(I)
B, $MCP
, $WEF or $WEFS
, IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch or Printer) Invalid operation
EPGK	(Tape) Write on file protected tape
UK	Repeated failure of operation
EE, EOP	Operation completed and tape crossed end mark
EOP	$WEF completed
None	$WEFS completed

Rewind

MREW or MREWS, IODNAME(I)
B, $MCP
, $REW or $REWS
, IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch or Printer) Invalid operation
UK	Repeated failure of operation
CS	$REW completed and new tape at load point
None	$REWS completed and tape at load point

Rewind and Unload

MUNLD or MUNLDS, IODNAME(I)
B,$MCP
,$UNLD or $UNLDS
,IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch, or Printer) Invalid operation
UK	Repeated failure of operation
CS	$UNLD completed and new tape at load point
None	$UNLDS completed and new tape at load point

Reserve Light Off

MRLF or MRLFS, IODNAME(I)
B, $MCP
, $RLF or $RLFS
, IODNAME(I)
(Return)

EPGK	(Disk or Tape) Invalid operation
UK	Repeated failure of the operation
EOP	$RLF completed
CS	(Reader, Punch, or Printer) Unit readied
None	$RLFS completed

Reserve Light On

MRLN or MRLNS, IODNAME(I)
B, $MCP
, $RLN or $RLNS
, IODNAME(I)
(Return)

EPGK	(Disk or Tape) Invalid operation
UK	Repeated failure of the operation
EOP	$RLN completed
CS	(Reader, Punch, or Printer) Unit readied
None	$RLNS completed

Check Light On

MKLN or MKLNS, IODNAME(I)
B, $MCP
, $KLN or $KLNS
, IODNAME(I)
(Return)

EPGK	(Disk or Tape) Invalid operation
UK	Repeated failure of the operation
EOP	$KLN completed
CS	(Reader, Punch, or Printer) Unit readied
None	$KLNS completed

Sound Gong

MGONG or MGONGS, IODNAME(I)
B, $MCP
, $GONG or $GONGS
, IODNAME(I)
(Return)

EPGK	(Disk, Tape, Reader, Punch, or Printer) Invalid operation
UK	Repeated failure of operation
EOP	$GONG completed
None	$GONGS completed

High Density

MHD or MHDS, IODNAME(I)
B,$MCP
,$HD or $HDS
,IODNAME(I)
(Return)

EPGK	(Disk, Console, Reader, Punch or Printer) Invalid operation
EPGK	(Tape) Tape is not at load point
EOP	$HD completed
NONE	$HDS completed

Low Density

```
MLD or MLDS, IODNAME(I)
     B,$MCP
     ,$LD or $LDS
     ,IODNAME(I)
     (Return)
   EPGK (Disk, Console, Reader, Punch or Printer)
        Invalid operation
   EPGK (Tape) Tape is not at load point
   EOP  $LD completed
   NONE $LDS completed
```

Even Parity, No ECC

```
MEVEN or MEVENS, IODNAME(I)
     B,$MCP
     ,$EVEN or $EVENS
     ,IODNAME(I)
     (Return)
   EPGK (Disk, Console, Reader, Punch or Printer)
        Invalid operation
   EOP  $EVEN completed
   NONE $EVENS completed
```

Odd Parity, No ECC

```
MODD or MODDS, IODNAME(I)
     B,$MCP
     ,$ODD or $ODDS
     ,IODNAME(I)
     (Return)
   EPGK (Disk, Console, Reader, Punch or Printer)
        Invalid operation
   EOP  $ODD completed
   NONE $ODDS completed
```

Error Checking and Correction

```
MECC or MECCS, IODNAME(I)
     B,$MCP
     ,$ECC or $ECCS
     ,IODNAME(I)
     (Return)
   EPGK (Disk, Console or Printer) Invalid operation
   EOP  $ECC completed
   NONE $ECCS completed
```

No Error Checking and Correction

```
MNOECC or MNOECCS, IODNAME(I)
     B,$MCP
     ,$NOECC or $NOECCS
     ,IODNAME(I)
     (Return)
   EPGK (Disk, Console, Printer or Tape) Invalid
        operation
   EOP  $NOECC completed
   NONE $NOECCS completed
```

The programming examples in this Appendix serve a two-fold purpose. First, they illustrate the appearance and position of MCP pseudo-operations intermixed with STRAP symbolic statements. Second, they give specific applications of the pseudo-operations to demonstrate their use to the programmer in the preparation of routines to be run under MCP.

```
B         JOB, ANYID
B         TYPE,COMPILGO,SMAC
T         SUBTYPE,A1
B SUEJEANIOD,TAPE,EXIT,,,ODD,HD
B         REEL,PLB12345
 INIT     LX,$1,BSTRY'          UK RETRY COUNTER
 READ     MRD,SUEJEAN,CW1'
          B,WAIT'
 START    LVI,$2,15.
 '  ************************************************
 '  ANY COMPUTATION TO BE DONE WOULD BE INSERTED HERE
 '  ************************************************
          BZB,EXIT+1.11,WAIT' TEST EE BIT
          MEOJ'               IF ON GIVE EOJ
 WAIT     MWAIT,SUEJEAN'
          BB,EXIT+1.9,EPGK'   TEST FOR EPGK
          BBZ,EXIT+1.10,UKFIX'TEST FOR UK
          R,$1'               RESET COUNTER
          SWAPI,1,CW1,CW2'    SWITCH BUFFERS
          MRD,SUEJEAN,CW1'    INITIATE READ
          B,START'
 UKFIX    BZB,EXIT+1.12,UNITBAD' NOT UK-EOP, UNIT FAILURE
          MBSP,SUEJEAN'       BACKSPACE AND WAIT
          MWAIT,SUEJEAN'
          BB,EXIT+1.9,EPGK'
          BBZ,EXIT+1.10,UNITBAD'
          CB,$1,READ'         TRY READ AGAIN
          MCOMM,MESG1,3.'     QUIT ON THIRD TRY
          MABEOJ'
 UNITBAD  MIODEF,SUEJEAN,MESG2+2.0'
          MCOMM,MESG2,4.'
          MABEOJ'
 EPGK     MCOMM,MESG3,5.'
          MABEOJ'
 CW1      CW,BUFFER1,N,0'     CW FOR BUFFER ONE
 CW2      CW,BUFFER2,N,0'     CW FOR BUFFER TWO
 BUFFER1  DR(BU),N'
 BUFFER2  DR(BU),N'
          CNOP'
 MESG 1   (IQS*)DD(BU), DATA ERROR UNCORRECTIBLE*'
          CNOP'
 MESG 2   (IQSX)DD(BU),****OPERATOR****CH UN IS DOWNX'
          CNOP'
 MESG3    (IQS*)DD(BU), THIS PROGRAM HAS BEEN CONTAMINATED.*'
 BSTRY    XW,,3,$'
 EXIT     DR(N),2'            TABLE OF EXITS
          MRET'
          MRET'
          MRET'
          MRET'
 N        SYN,1'
          MEND,INIT'
```

FIGURE 4. PROGRAMMING EXAMPLE 1: BUFFERED TAPE INPUT

```
      JOB,PPNAME
 TIME  14/27/18,  DATE 04/13/64,     VERSION     03/15/64
B        TYPE,COMPILGO,STRAP,NOPUNCH



 1*                                          B RON    IOD,TAPE,EXIT,,,ODD,HD,CSAVE
 2*          *                                        PRNS
 3* 000041.00      000040.10 00                START    B,$MCP'
 4* 000041.40      000104.40 80                         ,$SCR'                SYSTEM INPUT
 5* 000042.00      000055.01 80                         LVE,,CW+1.0'
 6* 000042.40      000001.00 80                         ,1.0'
 7* 000043.00      000000.00 80                         ,0'
 8* 000043.40      000051.10 00                         B,EOJ'                NO MORE CARDS
 9* 000044.00      000040.10 00                         B,$MCP'
10* 000044.40      000013.40 80                         ,$WAIT'
11* 000045.00      000001.00 80                         ,RON'                 TAKE INTERRUPTS
12* 000045.40      000054.00 80  000055.02 E0           SWAPI,1,CW,CW+1.0'    SWITCH BUFFERS
13* 000046.40      000040.10 00                         B,$MCP'
14* 000047.00      000001.41 80                         ,$WS'
15* 000047.40      000001.00 80                         ,RON'
16* 000050.00      000054.00 80                         ,CW'
17* 000050.40      000041.10 00                         B,START'
18* 000051.00      000040.10 00                EOJ      B,$MCP'
19* 000051.40      000007.41 80                         ,$WEFS'
20* 000052.00      000001.00 80                         ,RON'                 ASSUME SUCCESS ON $WEF
21* 000052.40      000040.10 00                         B,$MCP'
22* 000053.00      000101.00 80                         ,$EOJ'
23* 000054.00 *  000056.00+ 000 000017 000000  CW       CW,BUF1,15'
24* 000055.00    000075.00+ 000 000017 000000           CW,BUF2,15'
25* 000056.00 *     000017.00                BUF1     DR(N),15'
26* 000075.00       000017.00                BUF2     DR(N),15'
27* 000114.00       000002.00                EXIT     DR(N),2'
28* 000116.00       000122.10 00                      B,UK'
29* 000116.40       000000.30 00                      NOP'                  UK OR EPGK
30* 000117.00       000040.10 00                      B,$MCP'
31* 000117.40       000101.00 80                      ,$EOJ'                EE, TAPE FILLED
32* 000120.00       000040.10 00                      B,$MCP'
33* 000120.40       000104.00 80                      ,$ABEOJ'              CS NOT EXPECTED
34* 000121.00       000040.10 00                      B,$MCP'
35* 000121.40       000041.00 80                      ,$RET'                EOP
36* 000122.00       000115.11 80  000131.34 02  UK    BB,EXIT+1.09,EPGK'    TEST FOR UK OR EPGK
37* 000123.00       000040.10 00                      B,$MCP'
38* 000123.40       000006.01 80                      ,$BSPS'
39* 000124.00       000001.00 80                      ,RON'                 UK
40* 000124.40       000040.10 00                      B,$MCP'
41* 000125.00       000005.01 80                      ,$ERGS'
42* 000125.40       000040.10 00                      B,$MCP'
43* 000126.00       000001.40 80                      ,$W'                  $W EOP GIVES $RET AND
44* 000126.40       000001.00 80                      ,RON'                 CONTROL GOES TO MAINSTREAM



 1* 000127.00       000054.00 80                      ,CW'                  AT WAIT
 2* 000127.40       000040.10 00                      B,$MCP'
 3* 000130.00       000013.40 80                      ,$WAIT'
 4* 000130.40       000001.00 80                      ,RON'
 5* 000131.00 *     000040.10 00                EPGK     B,$MCP'
 6* 000131.40       000010.41 80                      ,$UNLDS'
 7* 000132.00       000001.00 80                      ,RON'                 REEL FILE PROTECTED
 8* 000132.40       000040.10 00                      B,$MCP'
 9* 000133.00       000015.00 80                      ,$IODEF'
10* 000133.40       000001.00 80                      ,RON'
11* 000134.00       000142.00 80                      ,MESG'                IDENTIFY THE UNIT
12* 000134.40       000040.10 00                      B,$MCP'
13* 000135.00       000043.40 80                      ,$COMM'
14* 000135.40       000142.00 80                      ,MESG'
15* 000136.00       000003.00 80                      ,3.0'                 TELL OPERATOR TO INSERT RING
16* 000136.40       000040.10 00                      B,$MCP'
17* 000137.00       000001.40 80                      ,$W'
18* 000137.40       000001.00 80                      ,RON'
19* 000140.00       000054.00 80                      ,CW'
20* 000140.40       000040.10 00                      B,$MCP'
21* 000141.00       000013.40 80                      ,$WAIT'
22* 000141.40       000001.00 80                      ,RON'
23* 000142.00 *     000001.00                MESG     DR(N),1'              CH XX UNY FILLED IN BY IODEF
24* 000143.00                                         (IQS*)DD(BU), REQUIRES A RING*'
25* 000145.00 *     000041.00                         END,START'
```

FIGURE 5. PROGRAMMING EXAMPLE 2: CARD-TO-TAPE ROUTINE

In Appendix D and E, many of the fields are to be filled in, such as job name, etc. These fields are indicated in two ways. A symbol may be inserted, which is to be explained thereafter, or a self-explanatory literal may be used, e.g., a literal such as 'ppname' indicates that the job name fills this field; 'reel' indicates that the tape reel name fills this field, etc.

Initialization Error Messages (IPL Program)

The Initialization error messages are all in the following form:

'location' - A TYPE IPLXX ERROR

where: 'location' is the address of the instruction +1. at which the error was detected.

XX is one of the following error codes:

00 An exchange failure (EKJ) was detected during the execution of a copy control word instruction.
01 The I/O test routine detected an error during the execution of an I/O operation, (EKJ, UNRJ or CBJ).
02 The space file operation was unsuccessful.
03 The read of the system tape was unsuccessful.
04 The disk locate was unsuccessful.
05 The disk write of PROSA was unsuccessful.
06 There were no IOCD cards detected.
07 An illegal equipment code was specified on an IOCD card.
08 An IOCD card was punched incorrectly.
09 A non IOCD or non IOD card was detected within the I/O Definition cards.
10 An illegal IOD card was detected.
11 The initialization program is unable to meet the I/O requirements defined, or is unable to assign the I/O units in the manner defined by the MCP IOD cards.
12 The I/O tables cannot be constructed properly with the information specified on the IOD cards.
13 The PROSA arc count is incorrect.
14 The MCP parameter input card is missing (MCPP card).
15 An incorrect operating mode was specified.
16 The system input assignments were not made correctly, or the system input IOD's were not defined for the overlapped mode.

Initialization Messages

These operator messages can only occur during initialization of MCP.

$ DATE OF IPL TAPE XX MONTH YY DAY.
The date specified reflects the date card used when generating this system tape.

$ STATUS REPLY: CHXX ZZZ DELETED.
Where ZZZ represents either UNY, or ALL which means the entire channel is deleted. This message is given in response to an I/O status report.

$ MAKE YOUR NEXT I/O STATUS REPORT.
This message directs the operator to specify any device that is currently not available to the system.

$ MCPIPL*- MOUNT REEL 'reel', ON CHANNEL XX, UNIT Y.
A number of messages of this type will be printed depending on the number of tapes required for system operation.

$READER IS INPUT SOURCE ON CHANNEL XX
The card reader is the system input source for either bypass or on-line system mode of operation.

Normal Running Messages

Once the system has been initialized, MCP receives control and issues a series of informative and action messages to keep the system running. The action messages are printed at the margin while the informative messages are shifted right for readability. The following represents a typical series of MCP issued messages while the system is running in the on-line mode.

$OPTR READY CHXX UNY.
In this example, the above message is a request for the operator to load and ready the card reader.

$OPTR SERVICE THE CARD READER.
The system expects more cards and will not reference the reader until the system gets the channel signal from the reader.

$ CHXX UNY-THE NEW IP READ TAPE.
The system has prescanned and rewound this tape. The first job on this tape will now be run.

$ JOB 'ppname' 'time' 'date'
This job will now be run.

$OPTR MOUNT THESE TAPES:
$ PP - 'ppname'
$ REEL 'reel' ON CHXX UNY 'protect'
$ PP REQS CHXX UNY 'reel'
The message PP REQS CHXX UNY 'reel' will be given if the desired tape (scratch or special name) is currently mounted because a previous job required the tape.

$ Begin Execution 'time'

$OPTR LOAD CHXX UNY WITH REEL 'reel' JOB 'ppname' 'protect'
The problem program or MCP has issued an I/O pseudo-operation and the tape must be loaded before the program can continue. In the next two messages, the SAVE field will be blank if the tape is not to be sent to the tape library.

$OPTR RPL CHXX UNY WITH RL 'reel' JOB 'ppname' 'save' 'protect'
The problem program or MCP has issued a $UNLD(S) and the indicated reel is to be loaded and readied.

$OPTR UNLD CHXX UNY JOB 'ppname' 'save'
The problem program is finished with this tape unit or MCP cannot assign this tape (left by a previous problem program) to the next problem program to be run.
If MCP issued the pseudo-operation that generated any of the three preceding messages, 'ppname' will be **MCP**.

Abnormal Running Messages

The following list represents problem situations of probable concern to the operator. In some cases, operator intervention is required; in other cases, the operator is informed about an unusual system response.

$INSERT RING ON CHXX UNY AND READY UNIT.
$SERVICE RD/FD CHK ON THE READER.
If the Feed Check light is on, there has been a card jam or a misfeed. Otherwise, a read check has occurred. To correct a read check: empty the hopper, press the Stop key, press the Unload key, place the last four stacker cards into the hopper, replace the cards that were in the hopper, and press Start.

"DATA" FAILURE ON CHXX UNY, nFILE MARKS ARE ON "PHYSICAL" END OF TAPE.
n = 0, 1 or 2
In the case of a unit failure, DATA will be replaced by the word UNIT. In the case of a failure prior to the physical end of tape (as indicated by the end of tape reflective strip), PHYSICAL will be replaced by the word LOGICAL. The output tape has encountered an uncorrectable UK situation. It may have been caused by a bad tape unit or bad tape reel while writing data, file marks or the end of tape trailer record.

$STRIP CHXX UNY BEFORE REUSING.
The beginning of the tape reel is worn and could not be written. Mount another reel in its place.

$CHXX UNY IS UNLOADING, INPUT FOUND END OF TAPE.

$PP XXXX IC = 'location' CNSL CS TO CONTINUE.
XXXX represents interrupts MK, IK, LJ, EK, and CPUS. The operator has the option at this point of giving the machine to the Customer Engineer or pressing the Signal key to $ABEOJ the job and continue.

$PP GIVEN CONTROL VIA $ABEX.
The operator has initiated a $ABEOJ using the console. The problem program has issued a $ABEX previously, so that the problem program will get control instead of being removed from the machine. A second operator initiated $ABEOJ will be necessary to remove the problem program from the machine.

$THE SCAN TABLES ARE FULL. THE PHASE I TAPE MAY BE STOPPED.
The I/O preassignment tables have been filled. Prescanning will resume as each problem program in execution phase terminates and creates space in the preassignment tables.

$NO JOB CARD IN THIS FILE.
A job has been lost because the first card of the file cannot be recognized as a JOB card.

Job Termination Messages

At the option of the installation, any unexpected problem program terminations can be typed on the console typewriter as well as on the output tape.

Refer to APPENDIX E for lists of the types of terminations. The exceptions to the lists in APPENDIX E are as follows:

1. Loader Rejects: Line 2 of the . LOD. REJ. CD. message will not be typed.
2. Operator Rejection messages will have already been typed out by the COMD package and will not be duplicated by the $EOJ procedure.

Command Responses

Command responses are operator messages given in response to a system command.

$ABEOJ TO PP. UNIT DOWN.
A request was made to delete a unit assigned to the problem program. The unit has been made unavailable, and the problem program terminated and dumped. An appropriate message is written on the system output tape.

$ABEOJ TO PP. MCP REQ UN.
A request was made to delete a unit or channel assigned to MCP. Sometimes, the configuration is such that in order for MCP to continue, a problem program unit must be reassigned to MCP. The problem program is terminated and

dumped with an appropriate message on the system output tape.

$NO REPLACEMENT FOR DELETED MCP UNIT.
A request was made to delete an MCP unit. The unit has been made unavailable but a replacement unit could not be found. The system will try to continue, but if too many tape units are taken from MCP or if key channels are deleted, a TYPE 82 error will result.

$ CHANGE MCP UNIT FROM XX-Y TO XX-Y.
An IOCHANGE command that deleted a unit or channel has been accepted.
The unit and channel number of the deleted unit, and the unit and channel number of the replacement unit for every unit deleted, is given for the operator's information. If the replacement unit was assigned to the problem program, the problem program is terminated and dumped, with appropriate messages given to the operator, and written on the system output tape.

$ 'command' COMMAND ACCEPTED.
This message is given when a command is accepted. Another command should not be entered from the console before this acceptance message or a rejection message for the current command is given.

$COMD REJ - ILEGL SITUATN.
A legal MCP command was given, but the situation under which it was given is invalid for the command: e.g., COMD, EOJ issued when a problem program is already terminated.

$COMD REJ - ILEGL.
A command was given that was not recognized as one of the MCP commands. Parameters were missing or incorrect, or the format was wrong.

$DISREGARD SYSTEM TAPE MOUNTING REQUESTS.
An off-line command has been received from the system input source. The tapes have been assigned for the off-line mode, but conditions cause the system to remain in the bypass mode; therefore, the request for off-line tapes must be ignored. This error can occur if the COMD card is not the last card in the card reader.

HH:MM:SS-TIME-HH:MM:SS DATE-mm/dd/yy
This message is issued in response to COMD, CLOCK which sets the time clock calibration constant. The time to the right has been computed using the time value just entered. The date is taken from the COMD. The date and adjusted time are included on each problem program's output.

Checkout Messages

Each time that one of the console functions is performed, an appropriate record will be made on the typewriter.

$ENT BK 'location' (O/D)
Enter from binary keys.

$DIS KL 'location' (O/D)
Display on binary key lights.

$SET IC 'location' (O/D)
Set Instruction Counter

$DUMP 'location 1' 'location 2' (O/D)
Dump.

The addresses typed out are taken from the numeric switches. Either the letter O or the letter D will appear in parentheses following the address, indicating whether the address is in octal or in decimal. This interpretation is determined by the setting of the decimal-binary switch.

Console Reject Messages

Under certain circumstances, console information will not be accepted. In such a case, a message will be typed and MCP will return to the location of the console interruption.

$EPGK ON CONSOLE.

$UK ON CONSOLE.

$NO PP CONSL IOD.
When the Signal key was depressed, the current problem program did not have a console IOD and, therefore, could not accept the channel signal.

$ MESSAGE ERASED.
This message is an indication that MCP has received the EE (caused by depressing the console Erase key) and has discarded the preceding typed information.

Note: The problem program is not involved in any of the above situations. The MCP conceptor routine handles all of these unusual console situations.

Error Messages

Error messages are given when an error condition occurs and MCP cannot continue.

$ SYSTEM ERROR. MCP WILL RE-IPL.
It is possible to execute an internal Restart should an error occur. If this is done, the gong is sounded once, the above message

printed, and the Restart program is automatically given control.

$PP XXXX. IC 'location' CNSL CS TO CONTINUE.

XXXX is either MK, IK, IJ, EK, or CPUS interrupt. The instruction counter contents are given and the operator may continue by pressing the Signal key on the console. If the Signal key is pressed, MCP will issue a $ABEOJ and the system will continue. A similar message will be written on the system output tape. In the case of the MK interrupt, after the channel signal is given, MCP removes the MK from core by reloading the affected location before continuing.

$MCP XXXX. IC 'location' IPL REQD.

An error interrupt has occurred in MCP. XXXX represents either MK, IK, IJ, EK, CPUS, AD, OP, DS, or USA interrupt. It is impossible for MCP to continue and the system must be reinitialized.

$COMD ERROR. IPL REQD.

The source of the command cannot be determined or the system was not taken out of the IPL Mode. The system must be reinitialized.

$TYPE XX ERROR. IPL REQD.

XX is one of the following codes:

75 Repeated failure of an MCP setup I/O operation
76 Repeated rejection by the exchange of some I/O command.
77 An interrupt was lost in the interrupt queue.
78 The available space in the interrupt queue has been exhausted.
79 The available space in the prime queue has been exhausted.
80 Attempted transfer from MCP major program to major program while auto stacked.
81 MCP received an unexpected EPGK interrupt.
82 There is a special assignment error. This error is usually caused by trying to assign a system unit that has already been assigned.
83 There is an error in the assign routine. MCP may be out of phase assigning jobs.
84 There is an error detected in job control. The disk has been written incorrectly.
85 Repeated unit checks on the disk.

Restart Messages

$READ FROM DISK OK.

This message is given after Restart has been positioned in core storage and has been given control.

$IPL OCCURRED DURING JOB 'ppname' 'system' MODE WITH IC = 'location'.

This message gives the status of the system when Restart was initiated.

$BEGIN WRITE OF $COMM OUTPUT.

This message is given before the commentator buffer is written on the typewriter.

$END WRITE OF $COMM OUTPUT.

This message is given after the write of the commentator buffer is completed.

$BEGIN WRITE OF OUTPUT BUFFERS.

This message is given before the print and punch buffers are written on the output tape.

$NO OUTPUT

This message means that there is no output available in the output buffers.

$XXX ON TP WRITE, OUTPUT SKIPPED

XXX = EKJ or UK

$MNT AND RDY OUTPUT TP ON CHXX, UN0

At the time when Restart was initiated, an output tape was not ready. The operator must ready a tape on unit zero of the requested channel.

$PUT RING IN TP NOW UNLOADING AND RDY IT.

The output tape to be used for the buffer write was file protected. After this message is given, the indicated tape will be unloaded.

$END BUFFER WRITE

This message signals that the output buffers have been successfully written on the tape.

$OPTR PUSH CNSL CS FOR DUMP.

The dump option was selected and the dump will be executed if the limits are set up correctly in the numeric switches and the console Signal key is pressed.

$NO CHECK SUM

This message is given in response to the option selected by key 60 to disregard calculating the PROSA check sum.

$CHECK SUM INCORRECT

$IPL THE TAPE

These two messages are given by Disk IPL and by the Restart program in MCP. They signal that the system is contaminated and must be reinitialized from the master system tape.

Standard Job Output Messages

The following messages are given by MCP for each job run, when they apply to that job.

JOB, 'ppname'_____

63 characters of the JOB card

TIME HH/MM/SS, DATE mm/dd/yy, VERSION mm/dd/yy

DATE = current date

VERSION = the current system level of MCP

B TYPE,_____

79 characters from the current job's TYPE card

BIODNAME IOD,_____

B REEL,_____

80 characters from the current job's IOD and REEL cards (if any), if the job successfully enters the GO phase.

BEGIN EXECUTION HH÷MM÷SS

Indicates the time clock contents when the current job successfully entered the GO phase.

System Rejects

JOB REJECTED 'Reason'

'Reason' is one of the following:

BY OPERATOR IN PHASE ONE.

The operator has inserted a COMD, REJECT in the card reader (in most cases after a malformed program deck) to reject the job preceding the COMD card.

DUE TO REPEATED UKS IN SCANNING.

The input program in MCP has been unable to read the definition cards for preassignment purposes because of UK difficulty.

TYPE CARD ERROR.

LIM CARD ERROR.

IOD CARD XXXX ERROR.

XXXX is the IOD reference number.

TAPE UNIT-CHANNEL CONFLICT.

The problem program has requested more tape units than are available in the machine's hardware configuration.

REEL CARD FOLLOWS A NON-TAPE IOD.

TYPE IS ILLEGAL ON AN IOD CARD.

TWO INFINITY DISK REQUESTS.

An infinity disk request is a disk IOD that requests the remainder of the disk (entire disk, minus PROSA, minus the arcs containing the program's binary deck). If two such requests reference the same disk channel, the job cannot be run.

FIRST IOD CARD IS ILLEGAL.

I/O REQ. INCOMP WITH MACH CONF.

The problem program has requested a set of I/O devices that cannot be satisfied by the current logical I/O device configuration (physical hardware minus those I/O devices used exclusively by MCP).

IOD CARD HAS A TYPE X ERROR.

X is one of the following numbers representing mispunched fields on an IOD card.

1 = disposition error

2 = mode error

3 = disposition and mode errors

4 = density error

5 = disposition and density errors

6 = mode and density errors

7 = disposition, mode and density errors

IOD CARD HAS AN INVALID REFERENCE NUMBER.

TOO MUCH DISK REQUESTED.

LIMIT IS TOO HIGH FOR I/O REQUESTS TO BE HONORED.

The space between the problem program's upper limit and MCP was insufficient to accommodate the IOD tables.

Loader Rejects

LOD. REJ. CD. XXXX ID YYYYYYYY 'reason' (Line 1)

LAST CORRECT CARD LOADED IS SEQUENCE NO. XXXX ID YYYYYYYY (Line 2)

where:

1. XXXX is the sequence number taken from column 3 of the binary card.

2. YYYYYYYY is the identification taken from columns 73-80 of the binary card.

3. 'reason' is one of the following:

CHECKSUM ERROR.

SEQUENCE ERROR.

ILLEGAL TYPE OF CARD.

ILLEGAL NON-BSS FUNCTION

-TRIED TO LOAD OUTSIDE LIMITS.

C, P, D, K, A, OR Z CD PUNCH ERR.

NO ORIGIN ON FIRST C OR K CARD.

NO ORIGIN IN FIRST CARD.

JOB CANNOT BE RUN BECAUSE OF INPUT UK DIFFICULTIES.

NO BRANCH CARD IN CURRENT DECK-JOB REJECTED.

Errors During Execution

JOB ENDED, - ERROR TYPE YY

Where YY is one of the following:

01 The problem program caused an IF interrupt, and the contents of the instruction counter -.32 is not a B, $MCP.

02 The problem program issued an invalid pseudo-operation.

03 The problem program's last referenced IOD, addressed a channel which is not available to the problem program.

04 The problem program issued an I/O request that referred to an illegal IOD reference number.

05 The problem program specified an illegal problem program I/O Table of Exits on an IOD card.

06 The problem program's last unload request could not be honored because problem program core storage, for the Reel Pool Table, was exhausted.

07 The problem program's last I/O reference, specified an invalid Control word.

08 The problem program's last I/O reference, specified an invalid Control word address.

09 The problem program attempted to communicate with protected storage, with one of the following pseudo-operations: $FIXUP, $TIME, $STLR, $FELR, $STRG, $FECRG.

10 The problem program specified a non-tape IOD in a $ATID calling sequence.

11 The problem program specified an invalid exit address in a $CHEX calling sequence.

12 The problem program attempted to actuate an I/O unit that had an interrupt stacked.

13 The problem program exceeded the limit of maskable interrupts that MCP will process.

14 The problem program caused either an OP, AD, USA, or DS interrupt while in the auto stacked mode.

15 MCP encountered repeated UKs while verifying a problem program tape label.

16 The problem program executed a successful branch to location 32.0 or 32.32.

17 The problem program specified an illegal PTOE address in the refill field of $15.

18 The problem program specified an invalid FWA in a $COMM calling sequence.

19 The problem program specified an illegal instruction in a $FIXUP calling sequence (either a SIC;BD, BD or an I/O instruction).

20 The problem program issued a $STLR when not in the Auto Stacked mode.

21 MCP encountered repeated UKs while reading the problem program's input.

22 The problem program requested a tape that could not be verified.

23 The problem program specified an error in a $SCR, $SPR, or $SPU calling sequence.

For error types 15 and 22, REEL 'NAME' is added to the JOB ENDED message to indicate which reel caused the problem.

The preceding error type numbers correspond to the $ABEX error code. Currently, codes 24-64 are unused.

XXXX INTERRUPT AT LOCATION 'location'

where XXXX is one of the following:

MK	(ABEX error code 65)
IK	(ABEX error code 66)
IJ	(ABEX error code 67)
EK	(ABEX error code 68)
CPUS	(ABEX error code 69)
EKJ	(ABEX error code 70)
UNRJ	(ABEX error code 71)
CBJ	(ABEX error code 72)
OP	(ABEX error code 73)
AD	(ABEX error code 74)
USA	(ABEX error code 75)
DS	(ABEX error code 76)

Note: Both types of "Errors During Execution" will cause a $ABEOJ dump which will include the location of the instruction following the one that caused the error.

Operation Rejection

EOJ REQUESTED BY OPERATOR

ABNORMAL EOJ REQUESTED BY OPERATOR
(ABEX error code 0)

ABNORMAL EOJ TO P. P. MCP NEEDS CHANNEL.
(ABEX error code 80)

Miscellaneous Messages

COMD, COMMENT, maximum 40 character message
Entered either by a COMD card or by the operator's console.

XXXX INTERRUPT AT LOCATION 'location'
XXXX may be any of the maskable interrupts.

$ABEX TABLE IS OUT OF PP BOUNDS
$ABEX specified a bad address, the problem program will continue but cannot regain control if a $ABEOJ occurs.

YOU HAVE BEEN GIVEN CONTROL VIA $ABEX, AT LOCATION 'location'.

OUTPUT AT IPL TIME FOLLOWS. SOME OUTPUT MAY BE REPEATED...
Restart gives this message before listing the contents of the output buffer.

Reel History Messages

The following lines are added to a job's output if any tapes were requested in the GO phase:

JOB 'ppname' - REELS. . . (Line 1)
CHXX UNY LAST IOD NO. ('number') 8 (Line 2)
AAANNNNN 'no.' UKS 'no.' UK-EOPS 'no.' RDS 'no.' WRTS 'no.' CTLS. 'saved' (Line 3)

AAA = one of the following tape descriptors
NLB, NUL, PLB, PUL

NNNNN = 'reel' or ***** for an unused scratch tape

Line 3 is repeated for as many reels as were requested for this channel and unit.

Lines 2 and 3 are repeated for each channel and unit requested by this problem program.

APPENDIX F. CHARACTER CODES

The following table lists the octal codes, by character, for eight-bit IQS, six-bit BCD, and the corresponding column binary punches. The IQS codes given for the letters and numbers are for upper case. For lower case letters, simply change the last bit of the upper case code to a 0 bit.

Character	IQS	BCD	Punches
A	055	61	12-1
B	057	62	12-2
C	061	63	12-3
D	063	64	12-4
E	065	65	12-5
F	067	66	12-6
G	071	67	12-7
H	073	70	12-8
I	075	71	12-9
J	077	41	11-1
K	101	42	11-2
L	103	43	11-3
M	105	44	11-4
N	107	45	11-5
O	111	46	11-6
P	113	47	11-7
Q	115	50	11-8
R	117	51	11-9
S	121	22	0-2
T	123	23	0-3
U	125	24	0-4
V	127	25	0-5
W	131	26	0-6
X	133	27	0-7
Y	135	30	0-8
Z	137	31	0-9
0	140	12	0
1	142	01	1
2	144	02	2
3	146	03	3
4	150	04	4
5	152	05	5
6	154	06	6
7	156	07	7
8	160	10	8
9	162	11	9
blank	000	20	none
+	041	60	12
$	042	53	11-3-8
=	043	13	3-8
*	044	54	11-4-8
(	045	34	0-4-8
/	046	21	0-1
)	047	74	12-4-8
'	050	33	0-3-8
;	051	52	11-0
,	052	14	4-8
"	053		
.	164	73	12-3-8
:	165		
-	166	40	11
?	167		

6/64: 1M - BP - 60

IBM 7030 Master Control Program Printed in U.S.A. C22-6678-1

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601