IBM 7030 DATA PROCESSING SYSTEM

MASTER CONTROL PROGRAM USER'S GUIDE

MCP, the Master Control Program, is designed to perform the following functions for the programmer:

1. Concurrently process one program while reading in subsequent jobs and putting out the results of previous jobs.
2. Call in processing programs and correction subroutines for execution.
3. Assign and protect storage areas for the programmer.
4. Maintain constant communication between the program and the operator.
5. Assign absolute input-output units to symbolic units defined in the problem program.
6. Actuate input-output operations as requested by the programmer via pseudo-operations.
7. Handle maskable interrupts for the programmer or, if desired, give him control of any of the maskable interrupts.
8. Handle I-O interrupts.
9. Perform various specialized functions for the programmer that are initiated by special pseudo-operations.

This manual is intended as a programmer's guide to the use of MCP. It provides him with descriptions of the above functions, and supplies calling sequences and table formats wherever necessary. A knowledge of the IBM 7030 Data Processing System Reference Manual, Form A22-6530, is assumed.

COMPOSITION OF THE INPUT DECK

Input to the system may consist of:

1. Definition cards
2. Symbolic Program Cards
3. Absolute Binary Cards
4. Data Cards

Column 1 of all cards is reserved for identification.

NOTE: This bulletin does not attempt to describe all of the functions performed by the Master Control Program. Instead, emphasis is placed on those conventions which the programmer must follow in his own programs in communicating with the control program and its facilities.
Moreover, all specifications and conventions applying to the Master Control Program that are described here are preliminary in nature and are subject to modification and/or expansion in the near future.

DEFINITION CARD

The five types of definition cards are:

1. Job Card
2. Type-of-Problem Card
3. LIM Card
4. IOD Card
5. Reel Card

A job card and a type-of-problem card must precede every deck in the order stated. If the first four types of definition cards are used, they must appear in the order in which they are listed above.

JOB CARD

This 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:

1	2 9	10 68	69 80
B		JOB, ANYIDENTIFICATION	

The identifying information is printed out on the typewriter and heads and follows any SPOOL output. At the option of the individual installation, an accounting program will also make use of the identifying information supplied on job cards.

TYPE-OF-PROBLEM CARD

This card must be punched in one of the following formats:

1	2 9	10 68	69 80
B		GO	
B		STRAP, NOLIST, NOPUNCH	
		or	
B		COLASL, NOLIST, NOPUNCH	
B		STRAPGO, NOLIST, NOPUNCH	
		or	
B		COLASLGO, NOLIST, NOPUNCH	

COLASL is the Los Alamos compiler.
NOLIST suppresses the listing produced by the compiler named.
NOPUNCH suppresses punching of the binary output deck by the compiler named. Either one, or both, of these fields may appear if a compiler is named. Neither is appropriate in a straight GO deck.

LIM CARDS

This card is present only in a GO deck, i.e., it can only follow a GO definition card. The LIM card, which is part of the output of STRAP II, is punched in the format:

1	2 9	10 68	69 80
B		LIM, A.0, B.0	

A and B are six-digit octal integers.
A = the full word address which contains the program's lowest bit. A must be $\geq 41_8$.
B = the full word address immediately following the program's highest bit address.

MCP uses the information contained in a LIM card for storage assignment; at load time, MCP loads A and B into the boundary registers.

IOD CARD

The configuration of I-O devices can vary not only from installation to installation, but also within a single installation from time to time. MCP assumes the responsibility of maintaining an up-to-date record of the status and availability of all I-O devices that are 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 an I-O requirement by giving parameters that specify the type of I-O device to be used, how the device is to be used, and the exit to be taken when the I-O operation is terminated. Several files may share one IOD statement.

STRAP II reproduces IOD cards exactly as they are submitted and includes them in the punched output. STRAP punches two additional fields in each duplicate IOD card: the I-O Reference Number field (columns 69-72), and the Absolute Exit field (columns 73-80). Thus, when writing an IOD statement, the programmer can use only columns 1-68.

The problem program names each IOD card with a symbol and thereafter refers to the particular I-O device by using this symbol as a parameter in the calling sequence for a pseudo I-O operation. (See Pseudo I-O Operations.)

The IOD card is divided into five fields:

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

2. Name: columns 2 through 9 contain a symbol up to 8 characters in length that names the symbolic file. Symbolic file is used here as a synonym for the I-O requirement defined by the parameters appearing on the IOD card.

3. Statement: columns 10 through 68 contain parameters, each separated by a comma, that describe the symbolic I-O file. These parameters vary with the type of I-O device requested, but the pseudo-op code and the first three parameters are the same for all types of units: IOD, TYPE, EXIT, and CHANNEL. IOD is the mnemonic for the pseudo-operation "I-O Definition." TYPE symbolizes the field where the programmer specifies the type of input-output unit required. TYPE may be PRINTER, CONSOLE, READER, PUNCH, DISK, TRACK, or TAPE. EXIT represents a programmer symbol that specifies the location of the first word of the I-O Table of Exits. (See I-O Interrupts.) The symbol EXIT may not contain more than eight characters. CHANNEL represents a programmer symbol, not more than eight characters in length, that will be assigned an absolute channel number by MCP. If the programmer supplies several IOD's and he wishes them all to be assigned to units on the same channel, he must use the identical channel symbol on each IOD card; if each request is to be assigned to a different channel, a different channel symbol must be specified on each IOD card. If no channel symbol is supplied (null CHANNEL field) on an IOD card, MCP assumes that channel assignment is of no importance to the problem program and will assign any unit available, regardless of the channel to which it is attached. If special channel assignments are requested and they cannot be honored by MCP, the problem program is run and the channels actually assigned conform as closely as possible tothe programmer's request.

4. I-O Reference Number: columns 69-72 are left blank by the problem program. On the output IOD's, STRAP II converts the IODNAME to an absolute number and punches this 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, etc.

5. Absolute Exit: columns 73-80 are left blank by the problem program. On the output IOD's, STRAP II punches the octal equivalent of the evaluated symbol for the first word address of the I-O Table of Exits in this field. (See "Input-Output Interrupts.")

The general format for an IOD card is:

1	2 9	10 68	69 72	73 80
B	IODNAME	IOD,TYPE,EXIT,CHANNEL,....	I-O REFERENCE NUMBER	ABSOLUTE EXIT

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.

1. Printer, Console IOD Card

1	2 9	10 68	69 72	73 80
B	IODNAME	IOD, TYPE, EXIT, CHANNEL	I-O REFERENCE NUMBER	ABSOLUTE EXIT

where TYPE is PRINTER, CONSOLE

2. Reader, Punch IOD Card

1	2 9	10 68	69 72	73 80
B	IODNAME	IOD, TYPE, EXIT, CHANNEL, MODE	I-O REFERENCE NUMBER	ABSOLUTE EXIT

where TYPE is READER or PUNCH
MODE is either ECC
NOECC
If MODE is null, NOECC is assumed.

3. Disk, Track IOD Card

1	2 9	10 68	69 72	73 80
B	IODNAME	IOD, TYPE, EXIT, CHANNEL, NUMBER	I-O REFERENCE NUMBER	ABSOLUTE EXIT

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 8 arcs, the first reserved arc being the first arc of a track. This permits the programmer to perform an auto access elimination operation, but his calling sequence must meet the requirements of the 7030 as stated in the Automatic Access Elimination section of the 7030 Reference Manual.

4. Tape IOD Card

1	2 9	10 68	69 72	73 80
B	IODNAME	IOD, TYPE, EXIT, CHANNEL, UNIT, MODE, DENSITY, DISPOSITION	I-O REFERENCE NUMBER	ABSOLUTE EXIT

where TYPE is TAPE

UNIT represents a programmer symbol which specifies an I-O unit of the type specified. The symbol may not be longer than eight characters. It is associated with a channel by means of the CHANNEL symbol. UNIT is assigned an absolute value by MCP after CHANNEL has been examined. The method of requesting several units on different channels has been explained under CHANNEL above. Except where TYPE is TAPE, all channels are one-unit channels; thus, in most cases, CHANNEL and UNIT are synonymous.

MODE is either ODD - odd parity, no ECC
EVEN - even parity, no ECC
ECC - odd parity, ECC

Density is either HD - high density
LD - low density

If MODE and/or DENSITY are null, the missing information is taken from the reel label (see Reel Labeling section). If the information cannot be obtained in this manner, the program is removed from the computer.

DISPOSITION may be NSAVE - do not save reels in any case
CSAVE - save reels <u>only if</u> job is complete
ISAVE - save reels <u>only if</u> job is incomplete
SAVE - save reels in <u>any</u> case

IF DISPOSITION is null, NSAVE is assumed.

A REEL card must immediately follow the tape IOD card to which it refers, unless there are 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 <u>that unit and channel.</u> A REEL card may follow another REEL card that refers to the same unit and channel. A REEL card is punched in the format:

1	2 9	10 68	69 72	73 80
B	blank	REEL, REEL1, REEL2, etc.	blank	blank

REEL represents a pseudo operation code that identifies a card that lists reel numbers of tapes associated with a TAPE IOD statement.

REELi represents any symbol up to eight characters in length. The first three characters are not part of the reel identification, but specify whether the tape is labelled or unlabelled and whether the tape is protected (ring out) or not protected. The remaining 5 characters agree with the identification shown on the physical reel. Thus REELi may be:

PLBxxxxx	protected, labelled
PULxxxxx	protected, unlabelled
NLBxxxxx	unprotected, labelled

If REELi is null, a labelled, unprotected tape is assumed.

SPOOL

SPOOL is the routine within MCP that performs the following services for the user:

1. Preprocesses input (data and programs).
2. Attempts to minimize handling of jobs by the operator.
3. Attempts to optimize use of the high-speed on-line devices.

The three functions in SPOOL are Read, Punch, and Print. Any of these functions can be operating simultaneously with another function or with the problem program. Furthermore, SPOOL may be operated in one of two modes, overlapped or not overlapped.

In the overlapped mode, successive jobs are read in or the results of successive jobs are printed out or punched out while the problem program itself, or another problem program, is being executed. Input and output is performed on, and only on, the I-O devices assigned to SPOOL. In the not-overlapped mode, the problem program has the exclusive use of SPOOL I-O devices. In effect, a program running in the not-overlapped mode is granted top priority and, therefore, by-passes all other jobs which may be in queue on SPOOL input and output tapes.

SPOOL operates in the overlapped mode unless a system command is given by the operator which places it in the not overlapped mode.

Once the mode has been established, the problem program communicates with SPOOL by means of pseudo-instructions that use calling sequences to initiate reading, punching and printing. These operations move data between SPOOL and the program via SPOOL I-O devices. The SPOOL I-O devices in effect at any time are defined by the operator.

SPOOL READER

The SPOOL reader maintains buffers that contain 15 words for each problem program data card. Eighty columns of each card are read. Data are moved from these buffers whenever the problem program uses the following calling sequence:

```
B, $MCP
 , $SCR
 , FWA(I)
 , N.
 , (N')
 ,    (End Return)
      (Normal Return)
```

The pseudo-operation symbolized by $SCR is interpreted as a request to move N column binary card records (15 word records) to the problem program storage area whose effective first word address (an 18-bit address field) is specified by the problem program symbol FWA(I). Control is returned to the problem program at the normal return in the calling sequence, when 15N words have been moved. N is a decimal integer.

If N card records have been specified, but there are fewer than N card records, the card records that are in the reader buffer are moved but return of control is made to the end return in the calling sequence and the number of cards actually moved is recorded in bits 0-17 of N' in the same calling sequence.

Example: At location ABLE, the problem program requests that the 10 cards in the reader buffer be moved to locations BAKER to BAKER +149, inclusive.

```
ABLE        B, $MCP
+  .32      , $SCR
+ 1.0       , BAKER
+ 1.32      , 10.
+ 2.0
+ 2.32      B, END
+ 3.0       ---
            ---
            ---
            ---
```

If the number of cards in the data file ≥ 10, control will be returned at ABLE + 3.0. If the number of cards in the data file <10 (8, for example), locations BAKER to BAKER + 119.0 inclusive will contain the data moved, bits 0-17 of location ABLE + 2.0 will contain 8, and control will be transferred to ABLE + 2.32.

SPOOL PUNCH

The SPOOL punch routine maintains buffers to hold output column binary card records (80 columns) in preparation for punching on the SPOOL card punch. Data is moved from a specified problem program storage area to these buffers whenever the following calling sequence is written:

```
B, $MCP
, $SPU
, FWA(I)
, N.
( Return )
```

SPOOL PRINT

Output data, in the form of 132 - character lines, are moved from the problem program's storage area to buffers maintained by the SPOOL printer whenever the following calling sequence is issued:

```
B, $MCP
, $SPR
, FWA(I)
, N.
( Return )
```

The pseudo-operation symbolized by $SPR is interpreted as a request to move edited lines, from the problem program's storage area whose effective first word address is specified by FWA(I), to the SPOOL buffer. N is a decimal integer that specifies the number of lines to be moved. Each line is assumed to consist of 17 words where successive 8-bit bytes are BCD characters, except for the last 24 bits of the 17th word, which are all zero. The first character of each line is a control character which must be one of the following:

-	$00\ 100000_2$	single space before printing
J	$00\ 100001_2$	double space before printing
1	$00\ 000001_2$	channel one before printing (restore)

When the N lines have been moved from problem program storage to the SPOOL buffers, control is returned to the return entry in the calling sequence.

The output of each program is preceded and followed by a line that identifies the output. This information is taken from the JOB card.

Example: The problem program has formed three lines (54 words) of output beginning at location BAKER. At location ABLE, the program requests that this block be moved to output SPOOL.

```
ABLE     B, $MCP
+  .32   , $SPR
+ 1.0    , BAKER
+ 1.32   , 3.
+ 2.0    ---
         ---
         ---
```

If this was the first use of output SPOOL, the problem program's identification is written before the output block is moved. Output SPOOL moves the contents of BAKER to BAKER + 53 to its buffers and returns control to ABLE +2.

INPUT-OUTPUT

Prior to problem program execution, MCP assigns an absolute channel and unit to each IOD. This information is also printed as a message to the operator. As part of this assignment, MCP stores the parameters appearing on the IOD cards in special tables. The IODNAME is then used to form the necessary addresses to retrieve the parameters on any particular IOD card from these tables. Thus, MCP relieves the programmer of any concern over absolute units and channels.

PSEUDO I-O OPERATIONS

All I-O devices are actuated by the Actuator subroutine within MCP. Use of an I-O device is requested by the programmer in calling sequence form, which varies with the type of operation requested. In general, the calling sequence takes the form:

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

where: OP represents an I-O operation
IODNAME is the symbol in the name field of an IOD card
I represents any index register

The dashes at the end of the calling sequence format indicate the space where the relative arc numbers or the symbolic locations of control words would be placed if required by the operation being requested.

The branch to $MCP causes control to be given to the Actuator subroutine. The IODNAME refers to a word in a table that in turn locates the block of parameters specified on the IOD card. If the channel is busy because an operation is in progress on that channel, the instruction counter is frozen until the channel is free.

If both the channel and the unit are free, the Actuator begins the set-up of the requested operation. This set-up may include:

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

In general, it may be said that Actuator supplies the necessary 7030 instructions that are required to perform the requested I-O operation.

When the operation has been set up and accepted by the exchange, control is given to the return address associated with the calling sequence (the location after the last control word given, or, if no control word is given, the location immediately after the half word where the IODNAME is specified).

Two restrictions apply to any control words that are addressed in an I-O calling sequence:

1. The control word(s) must be in storage.
2. The chain bit in the last control word must be 0.

All I-O operations available in the 7030 system may be actuated by means of the above calling sequence linkage. In addition, all I-O operations, with the exception of Copy Control Word, may be suffixed by the letter S (which represents SEOP) to suppress the EOP interrupt. The complete calling sequences for all I-O operations are listed in the Appendix.

REEL LABELING

The initial file of every tape reel used by MCP must be prepared in a special format and the information contained in this file is used only by MCP. This file consists of one record containing:

1. Reel identification - this matches the label on the physical reel.
2. Density in which the remainder of the reel is written.
3. Common mode for the reel; if a tape IOD card contains a null mode field, the tape unit is set to the mode specified in the reel label.

The reel identification is written off-line prior to use of the reel in the system. The tape mark following the initial file is written in the density given for the entire reel. The label itself is always written in the standard mode for the installation (here meant to be high density, even parity).

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 Signal 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.

COMMUNICATION BETWEEN MCP AND THE PROBLEM PROGRAM

Any problem program may issue a request to MCP to perform a specific function, such as stack I-O interrupts, write end-of-file, turn tape indicator off, etc.

Requests such as these are issued as pseudo-operations in the form of calling sequences. The first instruction in all pseudo-operation calling sequences is a B, $MCP. $MCP is a system symbol for an address in protected storage. The attempted branch causes an instruction fetch interrupt. The instruction fetch interrupt handling routine within MCP examines the rest of the calling sequence to determine which pseudo-operation is being requested and to initiate the appropriate action. Thus, the programmer is cautioned that his program must be enabled and the IF mask bit set to one.

Most of the pseudo-operations in this category are described in detail in other sections of this bulletin, such as Program Interrupts and Input-Output Operations. Two additional pseudo-operations are discussed here. The general format is:

```
B, $MCP
, $OP
```

where OP may be:

1. RESLD - Resume Loading. This pseudo-operation calls on the loader to resume loading binary cards from the appropriate SPOOL buffer, or from the disk if the

program is being run subsequent to a compilation. Control is returned to the address punched in the next branch card encountered.

2. EOJ - End of Job. This pseudo-operation terminates execution of the current problem program. MCP saves or releases the tapes used by the program in accordance with the dispositions specified on the IOD cards. The loader then calls in the next job from the appropriate SPOOL buffer.

INTERRUPTS

The linkage between MCP and a problem program, both intentional and non-intentional, is wholly dependent upon the 7030 interrupt system. Therefore, the problem program must run in the enabled mode. The programmer retains the option of ignoring any or all of the maskable interrupts during part or all of his program by setting the appropriate mask bit(s) to zero. Naturally, non-maskable interrupts must remain masked on or off as the circuitry dictates. The only exception to this rule is that Instruction Fetch must be masked on as previously explained. Interrupts are divided into the following four categories:

	Group	Indicators
1.	Equipment interrupts	0-8
2.	Input-Output interrupts	9-13
3.	Instruction Error interrupts	15-17, 19, 21*
4.	Program interrupts	18, 20, 22-47

* Indicator 21, Instruction Fetch, may represent a valid program interrupt when functioning as the linkage between MCP and the problem program.

EQUIPMENT INTERRUPTS

All equipment interrupts are completely outside the control of the problem program and are handled completely by MCP.

INPUT-OUTPUT INTERRUPTS

Receptor is the routine within MCP that handles processing of I-O interrupts. When any I-O interrupt occurs, the indicated six steps are performed sequentially.

Step One

MCP determines which I-O operation was being processed when the interrupt occurred.

Step Two

MCP saves the lower registers.

Step Three

MCP determines if the problem program is in the stacked or released mode. When a program is in the released mode, the problem programmer wishes to process immediately any I-O interrupt that may occur. When a program is placed in the stacked mode, the intention is to place any I-O interrupts that may occur in a queue and have them held in that status until such time as the released mode is entered. Then interrupts that have been stacked will be processed, one at a time, in the order in which they were stacked.

Pseudo-operations are used to invoke these modes:

```
B, $MCP
, $SIO
```

causes the program to operate in the stacked mode,

```
B, $MCP
, $RIO
```

places the program in the released mode.

If a program is in the released mode and an I-O interrupt is being processed, any further interrupts that occur during that interrupt will automatically invoke the stacked mode. Restoration to the released mode is also accomplished automatically by MCP and actually need not be of any concern to the programmer.

Step Four

MCP determines the appropriate I-O Table of Exits. Every IOD has a table of exits associated with it; several IOD's may, however, share a table of exits. Thus, if a program has N IOD statements, there may be as many as N or as few as one I-O tables of exits.

The purpose of an I-O table of exits is to provide storage space for information pertinent to an I-O interrupt. Each table is composed of six full words, and must begin at a full word address. Each of the last four words contains instructions supplied by the programmer.

word 0	0 17 Reference Number		18 31 blank	32 50 Actuated Addr.	51 63 blank
word 1	0 8 blank	9 13 I-O Inds.	14 31 blank	32 50 Interrupted Addr.	51 63 blank
word 2	ERROR RETURN INSTRUCTION				
word 3	END EXCEPTION RETURN INSTRUCTION				
word 4	SIGNAL RETURN INSTRUCTION				
word 5	NORMAL RETURN INSTRUCTION				

When an I-O interrupt is released to the problem programmer, the indicators are stored in the I-O indicators field of word 1. The actuated address, or the location of the first word of the I-O pseudo-operation calling sequence (the B, $MCP instruction) that requested the particular I-O operation that was being processed when the interrupt occurred, is stored in bits 32-50 of word 0. The reference number of the appropriate IOD statement is stored in bits 0-17 of word 0. Finally, the interrupted address (the location of the instruction being executed when the interrupt occurred) is stored in bits 32-50 of word 1.

The I-O table of exits must be constructed by the programmer. An easy way to accomplish this is to assemble the instructions

```
     CNOP
EXIT DRZ(N), 2
```

followed by four full-word instructions for the four returns. EXIT, the name of the first word of the table, is the symbol that appears in the IOD statement.

The pseudo-operation $CHEX, Change I-O Tables of Exits, permits the problem programmer to change the symbol EXIT in an IOD statement to a new symbol that represents a new location for the table (NEXIT). The format for this pseudo-operation is:

```
NAME    B, $MCP
        , $CHEX
        , IODNAME(I)
        , NEXIT
        (Return)
```

Control is returned to the problem program at the return word in the linkage when the operation has been completed.

The other four words in the I-O table of exits provide for entering correction routines. Normally, each word will contain two half-word instructions, one of which is a Branch instruction.

The significance of Signal Return, word 4, varies with the type of I-O device associated with the table. The alternatives are:

a. If unit is Reader, Printer, Punch or Console, signal return means a channel signal interrupt has occurred independently of the other I-O status bits.
b. If unit is a Disk, signal return is given upon completion of a locate instruction.
c. If unit is Tape, signal return indicates an attempt to move backward over the reel label or its associated tape mark (as previously explained in "Reel Labeling").

Step Five

MCP now analyzes the setting of the indicators stored in word 1. Depending on the status of these indicators, control is given to word 2, 3, 4, or 5. As previously stated, these locations will normally contain branches to other routines to correct the situation that caused the interrupt. As soon as one of the instructions in locations 2, 3, 4, or 5 is executed, the program is considered to be in an I-O correction routine. Therefore, all but the read-only portions of the lower register (0-31) are cleared to zero.

Step Six

The programmer must terminate all his I-O correction routines with the pseudo-operation:

```
B, $MCP
, $RET
```

The Return pseudo-operation tells MCP that the routine has been executed and that the programmer wishes to return to the main-stream of his program. MCP determines the current mode of the problem program. If it is in the stacked mode, return is made immediately to the main-stream. If the program is in the released mode, all stacked interrupts are released from the queue and processed one at a time, the lower registers previously saved are restored, and then control is returned to the main-stream. If RET is issued outside of the correction routine, it is treated as a NOP.

One additional operation is available within MCP that is associated with I-O interrupts. The pseudo-operation WAIT is issued via the calling sequence:

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

The interpretation of WAIT by MCP is determined by the status of the IOD statement referenced by the calling sequence via IODNAME.

	PROGRAM	
IOD	Not Stacked	Stacked
not busy	NOP	NOP
actuated	Wait for any interrupt	Wait for specific interrupt
stacked		Release specific interrupt

INSTRUCTION ERROR INTERRUPTS

Instruction error interrupts involve bits 15, 16, 17, 19, and 21 of the indicator register. Indicators 15, 16, 17, and 19 are permanently masked on.

In order to operate within MCP, it is the programmer's responsibility to keep indicator 21, Instruction Fetch, masked on. Failure to do so will result in loss of communication between the problem program and MCP and removal of the program from the computer.

All instruction error interrupts are handled completely by MCP; therefore, the programmer need not provide any correction routines for this class of interrupts.

PROGRAM INTERRUPTS

Program interrupts involve the following indicators:

18	Execute Exception
20	Data Fetch
22-34	Result Exception
35-38	Flagging
39-40	Transit
41-47	Program Indicators

Indicator bit 18 is permanently masked on. All other indicators in this category are maskable by the problem programmer. Once a programmer indicates that he wishes to take a program interrupt (by setting the appropriate mask bit to one), he may then elect to use either a special correction routine that he provides within his own program or a standard one provided with MCP. This choice is specified within a Program Table of Exits (PTE). The format of this table is:

word 1	0 18 INSTRUCTION COUNTER	blank
word 2	INDICATOR REGISTER	
word 3	MASK REGISTER	
word 4	PATTERN	
word 5	INSTRUCTION FOR FIRST PATTERN BIT SET TO ONE	
word 6	INSTRUCTION FOR SECOND PATTERN BIT SET TO ONE	
	------ ------ ------ ------ ------	
*word n+4	INSTRUCTION FOR THE Nth PATTERN BIT SET TO ONE	

*N equals the number of bits in the pattern set to one

The selection of standard or special correction routines is made in the PATTERN word of the PTE. In word four of the PTE there is one bit corresponding to each indicator in the program interrupt group, and each of these bits is in the same position in word 4 as are the indicators in the indicator register. If the programmer wishes to use a special routine for a particular program interrupt, he sets to one the bit in word 4 which corresponds to that indicator. If he wishes 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 PTE is required.

Word 5 of the PTE contains the instruction related to the first bit in the PATTERN (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.

If the programmer is supplying a Program Table of Exits, he must tell MCP where the table is located. This is accomplished by placing the location of the first word of the PTE in the refill field of index register 15, which is reserved for this purpose. (Obviously, then, the PTE must begin at a full-word address since the refill field of $15 is 18 bits in length.) Whenever the program reaches a phase where the mask that is in effect permits program interrupts to be taken, the initial location of the PTE must be in $15. Normally, then, one of the first instructions of a problem program would load this location, or zero, into the refill field of $15. If this field is zero, MCP assumes no PTE is provided. If the programmer wishes the use of special correction routines for an interrupt that occurs while a special correction routine is in progress, it is his responsibility to load the initial location of the new PTE (if one is required) in $15 at the appropriate time.

When a program interrupt is taken, MCP saves the instruction counter, the indicator register and the mask register in the first three words of the PTE, which must be reserved by the programmer for this purpose. At the conclusion of a special correction routine the programmer may elect to restore these three registers himself or have MCP restore the registers. To specify the latter choice, the programmer must provide a pseudo-operation in calling sequence form:

```
B, $MCP
, $RAM
( Return )
```

When the $RAM pseudo-operation is encountered, MCP restores the instruction counter, the indicator register and the mask register, and returns control to the problem program.

APPENDIX: INPUT-OUTPUT CALLING SEQUENCES

READ

```
B, $MCP
, $RD
, IODNAME(I)
, CTLWD(I')
( Return )
```

WRITE

```
B, $MCP
, $W
, IODNAME(I)
, CTLWD(I')
( Return )
```

LOCATE (for Disk)

```
B, $MCP
, $LOC
, IODNAME(I)
, ARC(I')
( Return )
```

Here ARC(I') is an indexed decimal integer which is relative to the number of arcs requested by the IOD. That is, ARC(I') is less than or equal to the NUMBER field of the disk IOD for IODNAME(I).

COPY CONTROL WORD

```
B, $MCP
, $CCW
, IODNAME(I)
, CTLWD(I')
( Return )
```

RELEASE

```
B, $MCP
, $REL
, IODNAME(I)
( Return )
```

RESERVED LIGHT OFF

```
B, $MCP
, $RLF
, IODNAME(I)
( Return )
```

RESERVED LIGHT ON

```
B, $MCP
, $RLN
, IODNAME(I)
( Return )
```

CHECK LIGHT ON

```
B, $MCP
, $KLN
, IODNAME(I)
( Return )
```

FEED CARD	B, $MCP , $FC , IODNAME(I) (Return)
SOUND GONG	B, $MCP , $GONG , IODNAME(I) (Return)
SPACE BLOCK	B, $MCP , $SP , IODNAME(I) (Return)
BACKSPACE BLOCK	B, $MCP , $BSP , IODNAME(I) (Return)
SPACE FILE	B, $MCP , $SPFL , IODNAME(I) (Return)
BACKSPACE FILE	B, $MCP , $BSFL , IODNAME(I) (Return)
ERASE LONG GAP	B, $MCP , $ERG , IODNAME(I) (Return)
WRITE END OF FILE	B, $MCP , $WEF , IODNAME(I) (Return)
TAPE INDICATOR OFF	B, $MCP , $TIF , IODNAME(I) (Return)
TAPE INDICATOR ON	B, $MCP , $TIN , IODNAME(I) (Return)
REWIND	B, $MCP , $REW , IODNAME(I) (Return)
UNLOAD	B, $MCP , $UNLD , IODNAME(I) (Return)

This pseudo-operation requests that the reel be rewound and removed. If, however, the programmer has specified no further reels for the particular file, the operator will mount a scratch tape. If two or more reels are asked for, it is advisable for the program to supply two IOD's to avoid waiting for the tape to be mounted.

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

This pseudo-operation accomplishes the following:

If the disposition for the particular file is either SAVE or CSAVE, the tape is rewound and unloaded. The empty tape drive is released. If the disposition is either ISAVE or NSAVE, the drive is released and the tape becomes a scratch tape.

Any of the pseudo-operations, except CCW and FREE, may be suffixed by S, which suppresses the EOP interrupt, except in rewind and unload, where the suffix S actually suppresses the channel signal interrupt.

IBM®

International Business Machines Corporation
Data Processing Division, 112 East Post Road, White Plains, N.Y. Printed in U.S.A. J22-6559 12/60