REFERENCE MANUAL

MODEL 80 DISPLAY IOCS

SYSTEM TEN COMPUTER BY SINGER

SINGER
BUSINESS MACHINES

REFERENCE MANUAL

MODEL 80 DISPLAY IOCS

SYSTEM TEN* COMPUTER BY SINGER

PUBLICATION NO. 40-700
OCTOBER 1972

SINGER
BUSINESS MACHINES
2350 WASHINGTON AVE.
SAN LEANDRO, CALIF. 94577

PRINTED IN U.S.A.

TABLE OF CONTENTS

Section 1
INTRODUCTION

MODEL 80 DISPLAY IOCS

DISPLAY FORMAT UTILITY

SYSTEM IMPLEMENTATION

INTRODUCTION

MODEL 80 DISPLAY IOCS

The Model 80 Display IOCS is an input-output control system designed to facilitate use of the Singer Model 80 Display for the System Ten* computer by Singer. The IOCS Macros enable the user to easily program the control, format, read, and write operations for the Model 80. The IOCS system supports a minimum of one and a maximum of ten Model 80 Display units per partition.

Figure 1-1 shows the IOCS macros. The macros are called from the user's program at assembly time, causing the applicable IOCS Macro expansions to be inserted directly into the user's applications program.

Requirements

Model 80 Display IOCS macros are used only in System Ten source programs assembled using Assembler II. Maximum memory requirement for the IOCS routines is 2K in partition.

Minimum Hardware Configuration

- Model 20 CPU, 2K partition storage for IOCS
- Model 80 Display Unit

Optional Equipment

- Additional Model 80 Display units
- Model 40/42 Disc Drives
- Model 70 Workstation
- Model 50 Line Printer

*A trademark of the Singer Company.

DISPLAY FORMAT UTILITY

The Format Utility program allows the user to create new format files, create format files based on existing files, and display and update existing files.

Requirements

The Model 80 Display Format Utility program is used under DMF and requires the DMF Partition LIOCS OPEN and CLOSE routines, and the SYSPOL._WORK file. The minimum memory requirement for the Format Utility program is a 9K System Ten partition.

Minimum Hardware Configuration

- Model 20 CPU, 9K Partition
- Model 80 Display Unit
- Model 40/42 Disc Drive,

Optional Equipment

- Additional Model 40 Disc Drives
- Model 70 Workstation

SYSTEM IMPLEMENTATION

Model 80 Display IOCS

The Model 80 Display IOCS program is released to System Ten installations as a set of Assembler II language macros in a single package; there are six separate macros which must be filed in any pool except SYSPOL using the names provided.

Model 80 Display Format Utility

The Model 80 Display Format Utility program is released to System Ten installations as object text loaded through the DMF Conversational Loader.

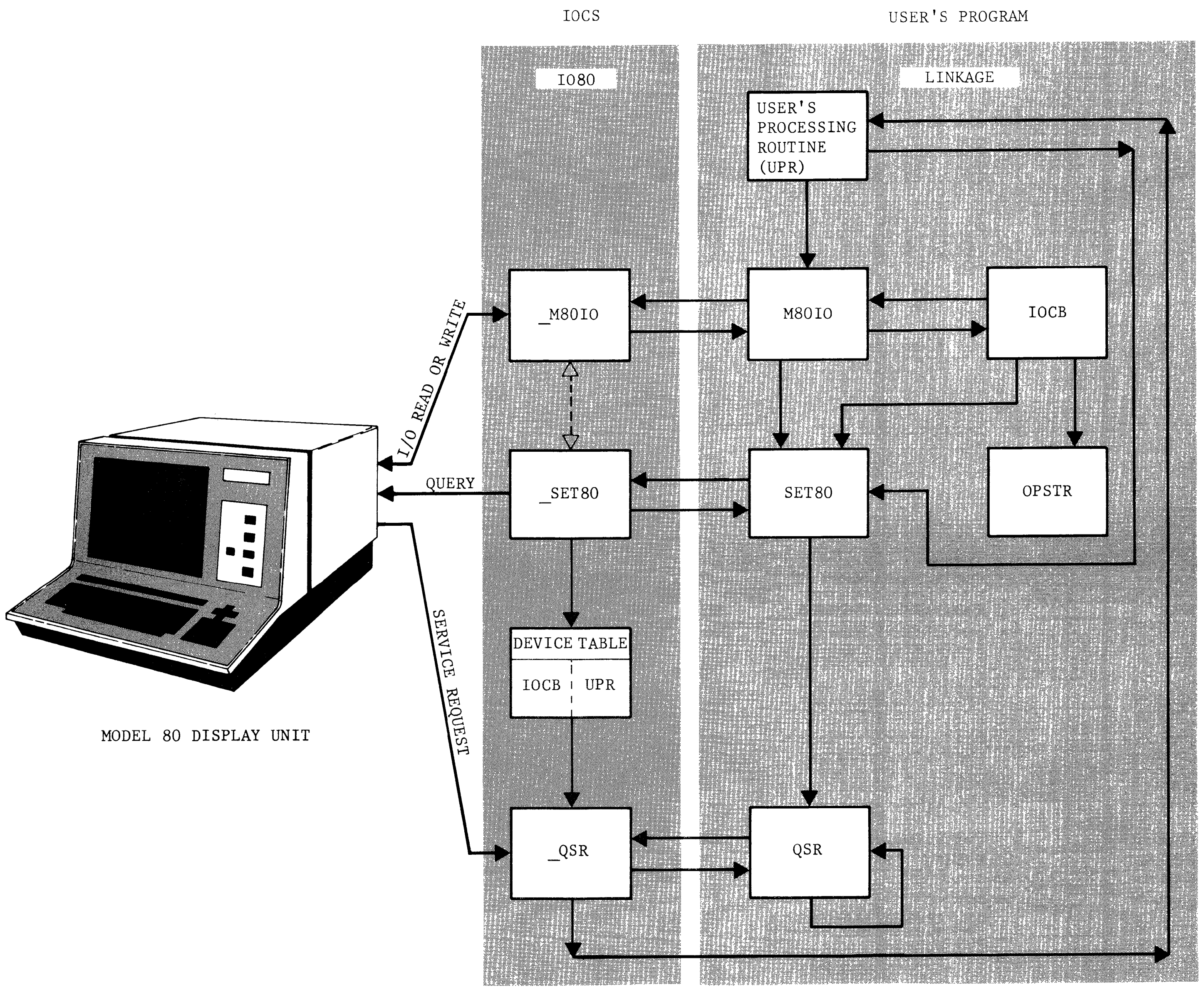


FIGURE 1-1. MODEL 80 DISPLAY IOCS MACRO BLOCK DIAGRAM

Section 2
IOCS SYSTEM MACROS

IOCS SYSTEM MACROS

GENERAL DESCRIPTION

The operations performed by the IOCS macros include read, write, and write control operations plus the capability to execute a 'format' file from disc. To accomplish this task, the IOCS macros require:

- An Input Output Control Block (IOCB) which contains:

  Address of the buffer area

  Length that is associated with the buffer area

  Device number

  Address of the Operation String

- An Operation String consisting of a string of operators and parameters describing the operation. The Operators are listed below:

  Write (W) for write operations

  Control (C) for a write control operation such as cursor movement, screen and keyboard conditioning.

  Read (R) for read operations

  Format (F) indicates a format file is to be executed. (A format file is an operation string filed on disc.)

  End (E) indicates the end of the operation string.

- Calling Sequences which pass parameters lists to the appropriate IOCS subroutines.

Other capabilities are provided to assist the user in controlling as many as ten Model 80's in a partition. These capabilities include:

- A 10-Entry Device Table which may be defined as part of the IOCS.

- A Service Request subroutine is provided to query service requests.

- An IOCS subroutine is provided to set or drop entries from the 10-Entry Device Table.

To support the functions performed by IOCS, the following six (6) macros are provided.

IO80

Inserts three IOCS subroutines (_SET80, _QSR, and _M80IO) and a 10-Entry Device Table into the user's program.

SET80

Establishes the link to the _SET80 subroutine that sets or drops entries from the 10-Entry Device Table.

QSR

Establishes the link to the _QSR subroutine that tests for a Service Request, and saves the device number.

M80IO

Establishes the link to the _M80IO subroutine that executes the Format, Input and Output Operations specified by the Operation String pointed to by the IOCB.

IOCB

Generates a 23-character IOCB that contains control information for the _M80IO subroutine.

OPSTR

Generates an Operation String that represents a series of operations directed to one or more Model 80 Display units.

[label]	IO80	[FORMT=$\left\{\begin{array}{l}\underline{\text{YES}}\\ \text{NO}\end{array}\right\}$][,WRITE=$\left\{\begin{array}{l}\underline{\text{YES}}\\ \text{NO}\end{array}\right\}$][,READ=$\left\{\begin{array}{l}\underline{\text{YES}}\\ \text{NO}\end{array}\right\}$]$\left\{\begin{array}{l}\text{,ERROR=label}\\ \text{,SRBR=(label1,label2)}\end{array}\right\}$

[label]	SET80	DEV=$\left\{\begin{array}{l}\text{ALL}\\ \text{QSRDEV}\\ \text{(n,n,....,n)}\\ \text{n}\end{array}\right\}$,ENTRY=label,IOCB=label[,ERROR=$\left\{\begin{array}{l}\underline{\text{IGN}}\\ \text{label}\end{array}\right\}$][,EXEC=label]

[label]	SET80	DROP=$\left\{\begin{array}{l}\text{ALL}\\ \text{QSRDEV}\\ \text{(n,n,....,n)}\\ \text{n}\end{array}\right\}$

[label]	QSR	(no operands)

[label]	M80IO	EXCEP=label[,IOCB=label][,DEV=$\left\{\begin{array}{l}\text{QSRDEV}\\ \underline{\text{n}}\end{array}\right\}$]

[label]	IOCB	OPSTR=label,IOADR=label[,IOLEN=nnnn][,DEV=n][,SHORT=$\left\{\begin{array}{l}\text{YES}\\ \underline{\text{NO}}\end{array}\right\}$]

[label]	IOCB	SHORT=YES,OPSTR=label[,DEV=n]

[label]	OPSTR	[$\left\{\begin{array}{l}\text{FORMAT(pool.file)}\\ \text{F(pool.file)}\end{array}\right\}$][,$\left\{\begin{array}{l}\text{READ}\\ \text{READ(0000)}\\ \text{R}\\ \text{R(0000)}\\ \text{READ(entry1,entry2)}\\ \text{R(entry1,entry2)}\\ \text{READ(label)}\\ \text{R(label)}\end{array}\right\}$][,$\left\{\begin{array}{l}\text{WRITE}\\ \text{WRITE(0000)}\\ \text{W}\\ \text{W(0000)}\\ \text{WRITE('data string')}\\ \text{W('data string')}\end{array}\right\}$] [,$\left\{\begin{array}{l}\text{CONTROL}\\ \text{CNTL}\\ \text{C}\end{array}\right\}$ $\left\{\begin{array}{l}\text{(string of parameters p,p,....,p)}\\ \text{('control character string')}\end{array}\right\}$][,$\left\{\begin{array}{l}\text{NOEND}\\ \underline{\text{END}}\\ \text{E}\end{array}\right\}$]

FIGURE 2-1. IOCS MACRO INSTRUCTIONS

IO80

This macro call is used once within the user's program and must precede all other IOCS macro calls. It requires a maximum of 2K of core partition to generate these subroutines if all features are desired.

IO80 Keyword Operands

[label]	IO80	[FORMT= $\left\{\begin{array}{l}\underline{\text{YES}}\\ \text{NO}\end{array}\right\}$][,WRITE= $\left\{\begin{array}{l}\underline{\text{YES}}\\ \text{NO}\end{array}\right\}$][,READ= $\left\{\begin{array}{l}\underline{\text{YES}}\\ \text{NO}\end{array}\right\}$] $\left\{\begin{array}{l}\text{,ERROR=label}\\ \text{,SRBR=(label1,label2)}\end{array}\right\}$

$\text{FORMAT=}\left\{\begin{array}{l}\text{YES}\\ \text{NO}\end{array}\right.$

Use FORMT=NO to eliminate the IOCS code that executes a format operation. This operand makes all format operations invalid and saves approximately 500 core positions. If this operand is omitted, YES is assumed.

$\text{WRITE=}\left\{\begin{array}{l}\text{YES}\\ \text{NO}\end{array}\right.$

Use WRITE=NO to eliminate the IOCS code that executes a write and a write control operation. This operand makes all write and write control operations invalid, and saves approximately 200 core positions. If the format file contains any write or write control operations, the format operation is also invalid. If this operand is omitted, YES is assumed.

$\text{READ=}\left\{\begin{array}{l}\text{YES}\\ \text{NO}\end{array}\right.$

Use READ=NO to eliminate the IOCS code that executes a read operation. This operand makes all read operations invalid and saves approximately 400 core positions. If this operand is omitted, YES is assumed.

ERROR=label

Use this operand to establish the address of the user routine that receives control whenever a Service Request is received from a device which has no entry in the device table. This operand is required when the SRBR operand is not specified.

SRBR=(label1,label2)

Use this operand to eliminate the 10-Entry Device Table and SET80 subroutine. This operand is used whenever all devices use one IOCB and one user's processing routine and saves approximately 400 core positions. The user routine (label1) is where control is transferred each time a Service Request is recognized. The IOCB (label2) is where the device number is stored.

SET80

To generate the linkage to the _SET80 subroutine that either sets or drops entries in the 10-Entry Device Table, issue the SET80 macro instruction and specify one or more of the keyword operands listed below.

SET80 Keyword Operands

```
                    ( ALL          )
[label] | SET80 | DEV=< QSRDEV       > ,ENTRY=label,IOCB=label[,ERROR= { IGN   } ][,EXEC=label]
                    ( (n,n,....,n) )                                   { label }
                    ( n            )
```

```
                     ( ALL          )
[label] | SET80 | DROP=< QSRDEV       >
                     ( (n,n,....,n) )
                     ( n            )
```

DEV=	ALL	Devices 0 thru 9
	QSRDEV	Device number stored by _QSR subroutine
	(n,n,n,.,.,.,n)	series of selected devices
	n	single device entry

The parameter specifies the device entries that will be modified when the storing function is executed. This operand is required.

ENTRY=label

The label is the entry point of the user's processing routine to be stored in the 10-Entry Device Table. This operand is required.

IOCB=label

The label refers to the address of the IOCB to be stored in the 10-Entry Device Table. This operand is required.

EXEC=label

This label represents the address of an IOCB and the presence of this operand causes the execution of the operation string pointed to by the IOCB to each on-line device specified in the DEV= operand.

$$\text{DROP=}\begin{cases}\text{ALL}\\ \text{QSRDEV}\\ \text{(n,n,n,.,.,n) (see DEV= operand)}\\ \text{n}\end{cases}$$

Use this operand to drop entries.

The 10-Entry Device Table is an optional feature generated by the IO80 macro. The table is 100 characters in length, 10 entries of 10 characters each, and represents the addresses of the IOCB and the user's processing routine for each device.

Figure 2-2 illustrates the 10-Entry Device table after initialization.

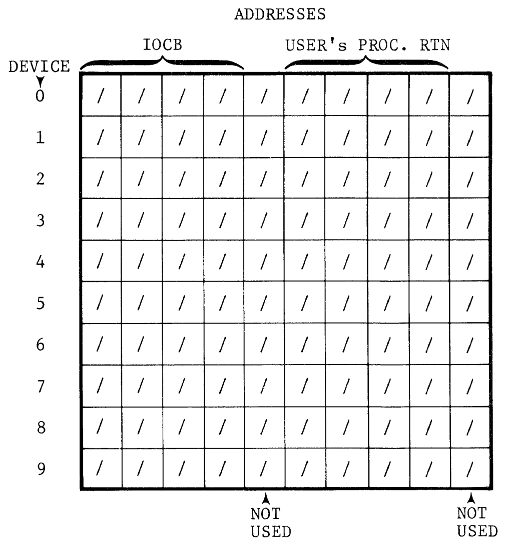
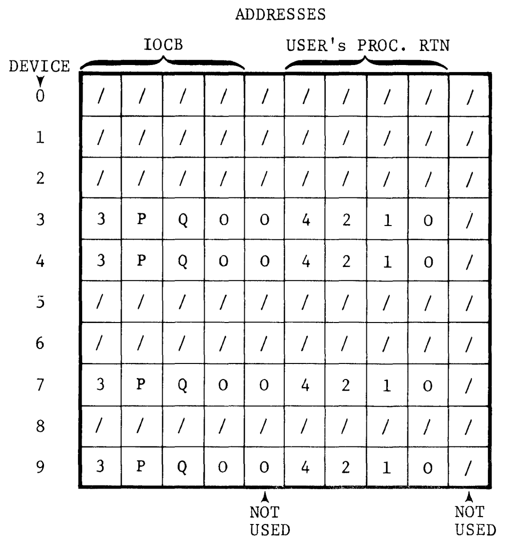


FIGURE 2-2. INITIALIZED 10-ENTRY DEVICE TABLE

To set entries in the table, issue the SET80 macro call.

    SET80 DEV=(3,4,7,9),IOCB=ABC,EXTRY=XYZ

For example, SET80 was called to set entries for device 3, 4, 7 and 9, the IOCB address (ABC=3010 in partition) and the user's processing routine address (XYZ=4210 in partition) in Figure 2-3.

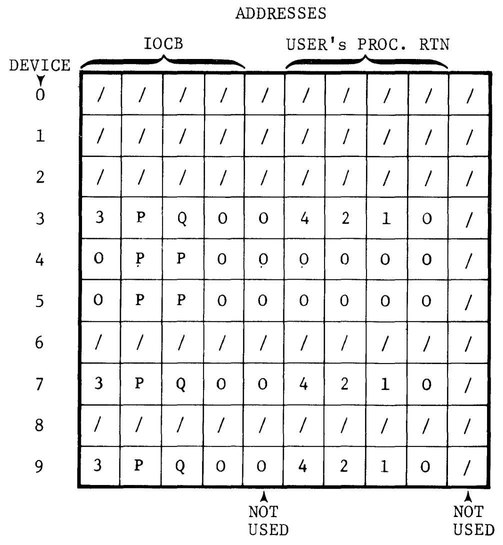

ADDRESSES
IOCB
USER's PROC. RTN
NOT USED
NOT USED

DEVICE										
0	/	/	/	/	/	/	/	/	/	/
1	/	/	/	/	/	/	/	/	/	/
2	/	/	/	/	/	/	/	/	/	/
3	3	P	Q	0	0	4	2	1	0	/
4	3	P	Q	0	0	4	2	1	0	/
5	/	/	/	/	/	/	/	/	/	/
6	/	/	/	/	/	/	/	/	/	/
7	3	P	Q	0	0	4	2	1	0	/
8	/	/	/	/	/	/	/	/	/	/
9	3	P	Q	0	0	4	2	1	0	/

FIGURE 2-3. I/O ENTRY DEVICE TABLE SHOWING STORED ENTRIES

To drop entries from the device table, issue the SET80 macro call for the devices to be dropped. Figure 2-4 illustrates entries dropped for devices 4 and 5. Example:

    DROP DEV=(4,5)


ADDRESSES
IOCB
USER's PROC. RTN
NOT USED
NOT USED

DEVICE										
0	/	/	/	/	/	/	/	/	/	/
1	/	/	/	/	/	/	/	/	/	/
2	/	/	/	/	/	/	/	/	/	/
3	3	P	Q	0	0	4	2	1	0	/
4	0	P	P	0	0	0	0	0	0	/
5	0	P	P	0	0	0	0	0	0	/
6	/	/	/	/	/	/	/	/	/	/
7	3	P	Q	0	0	4	2	1	0	/
8	/	/	/	/	/	/	/	/	/	/
9	3	P	Q	0	0	4	2	1	0	/

FIGURE 2-4. I/O ENTRY DEVICE TABLE SHOWING DELETED ENTRIES

QSR

QSR generates the link instruction to _QSR which were generated by the IO80 macro. _QSR tests for a Service Request coming from any device on that partition. If a Service Request was posted, _QSR will retrieve IOCB and entry point data (set by the SET80 macro call) from the device table relative to the recognized device. The IOCB address is stored in Index Register 3 and the device number is stored in the IOCB. Control is then passed to the user's processing routine. If no user processing routine was entered in the device table, the error routine specified by the ERROR= operand of the IO80 macro receives control. When a service request is recognized, the device number will be stored in a field labelled QSRDEV. This field is accessable by the user and will remain set until another service request is recognized by _QSR or a call is made to the _M80IO subroutine. If no Service Request is posted, control is passed to the user's next sequential instruction.

[label]	QSR	(no operands)

M80IO

To generate the linkage to the _M80IO subroutine that executes the format, input and output operations specified by the Operation string, issue the M80IO macro instruction and specify one or more of the keyword operands listed below.

There are three basic forms for the M80IO macro.

The first is the macro name with a single operand, the name of the user's exception routine (EXCEP=label). This calling sequence is used when control has come from _QSR (the device number and the IOCB Address are already known).

The second calling sequence is the macro name with two operands, the IOCB name and the name of the user's exception routine (IOCB=label, EXCEP=label). This calling sequence is used when the user program initiates I/O in a specific IOCB without having received control through _QSR.

The third calling sequence will store a device number in the specified IOCB and executes it as in the second calling sequence (EXCEP=label,IOCB=label,DEV=QSRDEV).

M80IO Keyword Operands

[label]	M80IO	EXCEP=label[,IOCB=label][,DEV= $\left\{\begin{matrix}\text{QSRDEV}\\ \text{n}\end{matrix}\right\}$]

EXCEP=label

Use this operand to specify the address of the user's error exception routine. This operand is required.

IOCB=label

Use this operand to specify the address of the IOCB that contains the device number, address of the Operation String, and the buffers required for execution of _M80IO. If this operand is omitted, the contents of Index Register 3 is used as the address of the IOCB.

> NOTE: _QSR moves the IOCB address from the device table into Index Register 3.

DEV= $\left\{\begin{matrix}\text{QSRDEV}\\ \\ \text{n}\end{matrix}\right.$

Use this operand to move either the device number stored by the _QSR subroutine or the specified device number into the IOCB specified. If DEV= is omitted, the device number is assumed to be preset.

IOCB

To generate an IOCB that contains control information for the _M80IO subroutine, issue the IOCB macro instruction and specify one or more of the keyword operands as listed below.

[label]	IOCB	OPSTR=label,IOADR=label[,IOLEN=nnnn][,DEV=n][,SHORT= $\left\{\begin{array}{l}\text{YES}\\ \underline{\text{NO}}\end{array}\right\}$]

There are two IOCB formats, short and standard.

The standard IOCB contains 23 characters and allows all operators to be used in an Operation String.

IOCB Keyword Operand Standard

[label]	IOCB	OPSTR=label,IOADR=label[,IOLEN=nnnn][,DEV=n][,SHORT= NO]

OPSTR=label

Use this operand to set the label of the Operation String into the IOCB. This operand is required.

IOADR=label

Use this operand to set the label of the data area into the IOCB. This operand is required.

IOLEN=nnnn

Use this operand to set the length of the data area into the IOCB. One to four numeric characters are allowed, however, this parameter must be non-zero. If this operand is omitted, the length of the data area given for IOADR is assumed.

DEV=n

Use this operand to set the device number into the IOCB. This field is set by the _QSR subroutine or the user before calling upon the _M80IO subroutine. If this operand is omitted, device zero is assumed.

$$\text{SHORT=}\left\{\begin{array}{l}\text{YES}\\ \\ \text{NO}\end{array}\right.$$

Use SHORT=YES to generate a Short IOCB. If this operand is omitted, NO is assumed.

IOCB Short Form

[label]	IOCB	SHORT=YES,OPSTR=label[,DEV=n]

The Short IOCB contains only ten characters and allows only certain Operators within the Operation String. These Operators are as follows:

- CONTROL
- END
- FORMAT (is allowed when the format file contains no read operators and the read length (characters 1-4 in the first record of the file) is Zero.
- WRITE

NOTE: This Operator can be used only when the characters that are to be written are contained within the WRITE ('data string') form.

If a read operator is used, _M80IO assumes a standard IOCB and user fields will be destroyed beyond the end of the short IOCB.

IOCB Field Definitions and Labels

_IOTYP	_IODEV**	_IOPRM*	_IOCNT	_IOSC	_IOBUF*	_IOLEN*	_IOFLG	_IORD
0	1	2 5	6 8	9	10 13	14 17	18	19 22
TYPE IDENTIFIER	DEVICE NUMBER	ADDRESS OF OPERATION STRING	OPERATOR COUNT	STATUS CODE	ADDRESS OF BUFFER	LENGTH OF READ OR WRITE	CONTROL KEY TERMINATING READ or ?	NUMBER OF CHARACTERS READ

|←———— SHORT IOCB ————→| (bytes 0–9)

|←———————— LONG IOCB ————————→| (bytes 0–22)

OPSTR

To generate an Operation String that represents a series of operations directed to one or more Model 80 Display units, issue the OPSTR macro instruction and specify one or more of the operands listed below.

OPSTR Operands

```
                                               ⎧READ                  ⎫     ⎧WRITE                 ⎫
                                               ⎪READ(0000)            ⎪     ⎪WRITE(0000)           ⎪
                                               ⎪R                     ⎪     ⎪W                     ⎪
[label]  OPSTR  [ ⎧FORMAT(pool.file)⎫ ][,     ⎨R(0000)               ⎬ ][, ⎨W(0000)               ⎬ ]
                  ⎩F(pool.file)     ⎭         ⎪READ(entry1,entry2)   ⎪     ⎪WRITE('data string')  ⎪
                                               ⎪R(entry1,entry2)      ⎪     ⎩W('data string')      ⎭
                                               ⎪READ(label)           ⎪
                                               ⎩R(label)              ⎭

                [, ⎧CONTROL⎫ ⎧(string of parameters p,p,....,p)⎫ ][, ⎧NOEND⎫ ]
                   ⎨CNTL   ⎬ ⎩('control character string')     ⎭     ⎨END  ⎬
                   ⎩C      ⎭                                         ⎩E    ⎭
```

These operands may be repeated in a single OPSTR macro call as many times as necessary and in any order to achieve the desired results.

FORMAT(pool.file)

F(pool.file)

Use this operand to generate the Format (F) Operator. Pool name is six characters and must be included and separated from the six character file name by a period. Pool and file name represent the name of a format file.

READ

READ(0000)

R

R(0000)

READ(entry1,entry2)

R(entry1,entry2)

READ(label)

R(label)

In using the Read Operator, entry1 is the length of the read, entry2 is the buffer address, label is the buffer address and assumes the assigned buffer length as the length of the read. Omission of these fields assumes the length and address in the IOCB that initiated the read.

```
WRITE

WRITE(0000)

W

W(0000)

WRITE('data string')

W('data string')
```

The data to be written follows the Write Operator in the Operation String. If the quotes are omitted, a warning MNOTE is generated. If the 'data string' is not used, the length and address in the IOCB that initiated the write are assumed.

```
CONTROL(string of parameters p,p,...,p)
CNTL(string of parameters p,p,...,p)
C(string of parameters p,p,...,p)
CONTROL('control character string')
CNTL('control character string')
C('control character string')
```

The Control Operand is followed by a control character string or a parameter string. The control characters and parameters are shown in Figure 2-5. Control characters must be repeated as necessary. For example, the parameter NLINE=5 would be JJJJJ in a control character string.

PARAMETER	ALTERNATE PARAMETER	CONTROL CHARACTER	DESCRIPTION
LOCK	LK	Q	Lock keyboard
UNLOCK	UL	T	Unlock keyboard
BLOCK	B	R	Initiates the block mode read
FORMAT	F	S	Initiates the formatting mode write
HOME	H	↑	Cursor home
UP=nn	U=nn	\	Cursor up
DOWN=nn	D=nn	]	Cursor down
RIGHT=nn	R=nn	Y	Cursor right
LEFT=nn	L=nn	H	Cursor left
NLINE=nn	NL=nn	J	New line
TAB=nn	T=nn	I	Tab to next unprotected field
ERASEX	CLEAR	X	Erase all characters
ERASE	E	L	Erase all unprotected characters
NOTE: nn represents a number from 01 - 99 (defaults to 01)			

FIGURE 2-5. MODEL 80 DISPLAY UNIT PARAMETERS, ALTERNATE PARAMETERS, AND CONTROL CHARACTER SET

END

E

NOEND

Use END to generate the End(E) Operator. This operand terminates the Operation String. If this operand is omitted and the NOEND Operand is not specified, the End (E) Operator is automatically generated. This operand or the NOEND Operand must be the last operand specified. Use NOEND to generate a contigious (multi-labeled) Operation String. Whenever this operand is specified, the END Operand is not used.

PROGRAMMING CONSIDERATIONS

1. Locking and unlocking the Keyboard

   The Model 80's keyboard is locked prior to executing any Operation String, except where the first Operator in the Operation String is a Read. If a read is imbedded within an Operation String, the keyboard is unlocked just prior to the read. The keyboard is locked following any read. It is not unlocked automatically upon exiting _M80IO, but must be setup as part of a Control Operator, if the user wants to unlock it.

2. In a multi-terminal environment where the _QSR routine is used:

   A. First operator in an operation string must be a read operator (R).

   B. Last control operation must be a control containing keyboard unlock.

   C. If the next read is to be a block read, the last control operation must also contain set block mode.

3. When using a Format operation, a read length associated with that format file which is greater than '0000' will be stored in the IOCB(_IOLEN). Therefore, if a fixed length is expected in _IOLEN field of the IOCB care must be taken that:

   A. All format files used in connection with the specific IOCB have read length assigned as '0000'.

      or

   B. After each format operation (normal and/or error return) restore the fixed length in the IOCB.

4. When the fields _IOFLG and _IORD, in the IOCB are interrogated by the user, it is important to note that those fields represent values from the last read. They remain set until the next subsequent read on same IOCB which will clear those fields before the read, and set new values after the read.

5. Model 80 Programming Considerations

   A. Cursor goes to a new line after a read.

   B. A clear control operation also sets cursor home.

   C. A block mode control stays in effect until the next read, regardless of other intervening operations.

   D. A format mode control stays in effect until the next Write operator, regardless of other intervening operations.

   E. When a line is written that causes the cursor to stop at the first character of a new line, the condition is "remembered" by the Model 80 until a data character is written or a subsequent new line write control is received and ignored.

   F. When a read is short of the number of characters entered, a 'STOP CHARACTER' remains in the screen buffer and will 'cut short' any subsequent read of greater length. Therefore, it is a recommended practice to:

      - In line mode, read 0080 characters.

      - In block mode, read 1600 characters.

   G. When a block mode read is terminated on the last line of the screen, a status code 1 is set indicating a "page wrap" on the read since the cursor has gone to the top of the screen. This status code has priority over status code 3 which indicates a control key terminated the read and therefore, a control key will never be recognized in this situation.

Section 3
MODEL 80 DISPLAY FORMAT UTILITY

MODEL 80 DISPLAY FORMAT UTILITY

GENERAL DESCRIPTION

Model 80 Display Format Utility Program is an interactive utility program that builds a format file to be used by the Model 80 Display IOCS program to format the screen of the Model 80. The program requires a Model 80 display unit for operator input/program output and a work file containing a minimum of 40 sectors. The program enables the user to create a new format file, create a format file using existing files, display and update existing format files.

> NOTE: Throughout the discussion of the Model 80 Display Format Utility, the operator action to accept or approve the display is by depressing the ENTER key. To disapprove or reject the display, the operator depresses any control key except control key 5 (1-4, 6-10). Depressing control key 5 reinitializes the Format Utility Program. CTL key 11 is reserved to indicate a return to the DMF Conversational Loader.

Before starting a formatting job the user may want to create a work file or display and change the program delimiters.

WORK FILE

To establish a work file (other than the default work file) enter the command listed below.

```
WORK FILE=pool.file[,KEY='password']
```

The work file is established specifically for the Format Utility program. It is required by all of the utility commands except for DELIMITER.

The work file is used to keep an intermediate version of the format that is being processed to build a new format file or to update (without altering) an existing file. The default work file is SYSPOL._WORK.

The MAINT utility and FORMAT programs share the default file (SYSPOL._WORK) and must not be run concurrently.

Also, the FORMAT program uses Partition LIOCS and does not provide for interpartition contention of pool free sector list space when the output or work file are created.

DELIMITER

Use the DELIMITER command listed below to override the program generated delimiter values, or display the current delimiter values.

```
    DELIMITER
```

To create or update a format file, five special characters or delimiters are used to identify particular operations. All other characters entered by the operator represents a character that will be written as protected data. Listed below are the delimiters set by the program if this command is omitted.

DELIMITER	MEANING
(	Start-of-Format delimiter
)	Cursor-Right delimiter
@	Protected-Character delimiter
\	End-of-Line delimiter
^	Unprotected-Blank delimiter

- Start of Format Delimiter

The Start of Format delimiter is used to indicate that the next character is the start of the format when the format begins in any other position other than the home position. It may be used only one time in a format. All preceding characters must be either the Protected-Character or Unprotected-Blank delimiters. At run time the cursor control operation string positions the cursor at the character following the Start-of-Format delimiter. The format starts writing at this position and any data that was on the screen ahead of this point is left undisturbed.

- Cursor-Right Delimiter

  One Cursor-Right delimiter is used to indicate movement of the curosr each position to the right.

- Protected-Character Delimiter

  The Protected-Character delimiter is used to indicate the position of a protected character to be left on the screen.

- End-of-Line Delimiter

An End-of-Line delimiter is used to indicate the end of format line which is less than 80 characters in length.

- Unprotected-Blank Delimiter

  The Unprotected-Blank delimiter is used to indicate that this character position is to be replaced by an unprotected space in the format file.

The program displays the current delimiter values and meanings. Depress the ENTER key to accept the delimiter. Depress the CTL key to override, and enter the five new delimiters in the order specified by the program. The program redisplays the new delimiters for operator acceptance.

CREATING NEW FORMAT FILES

To create a new format file, enter the command listed below.

```
CREATE pool.file[,KEY='password']
```

If the file exists and is a null type '0' linked sequential file, the program continues. If the file is non-null, an error message is displayed.

A password may be assigned if a new file label is to be generated. An error message is displayed if a password exists on an existing null file and does not match the password provided.

A new file label is created as a linked sequential type '0' file, and contains the following information.

- Password or Blanks
- Block and Record Length (set at '0094')
- Text Description ('Model 80 FORMT')
- Creation Date '000000'
- Expiration Date '000000'

To create a format file, the Model 80 screen is filled with the Unprotected-Blank delimiter. The body of the format is created by using the following characters.

1. Any Unprotected-Blank delimiter left on the screen represents an unprotected space.

2. Any characters that are not delimiters represent protected data characters.

3. All other delimiters placed on the screen represents specific cursor movement.

Depress the ENTER key when the desired format has been created. The program analyzes the format, computes the read length, the pre-format and post-format cursor control, and screen conditioning control characters. The operator is requested to approve the control operations preceding the format or to enter new control operations. Control operations preceding the format, if the format begins at the home position, are defaulted to clear all characters from the screen. The cursor is set at the home position and sets the format 'mode write'. See Figure 3-1 for list of control operators.

If the format begins in other than the home position, the default control operations are:

1. set the format mode write

2. set the cursor at the home position

3. move the cursor to the beginning of the format

The operator is requested to approve the control operations following the format, or enter new ones. Control operations following the format represent where the cursor will be placed following the format. The default control operations are:

- Set Cursor Home (first unprotected character)

- Set Block Mode Read

- Unlock Keyboard

The screen is filled again with the unprotected blank delimiter and the format is displayed for acceptance. Upon accepting the format, a request is made to approve the computed read length or to enter a new read length. The read length is the computed value supplied by the program. It gives the number of unprotected character fields starting with the home position to the position of the cursor where the enter key was depressed when the format was entered. The user may adjust this read length with any value between '0' and '1600'. If zero is specified, it indicates the read length will be supplied by the user's application program. When the read length is established, the CREATE file operation is complete.

If the operator does not approve the format, the file is decoded to re-display the screen image as it was created, and the operator re-creates the format.

NOTE: The Format program assumes all cursor movement to be to the right or new lines and therefore, some cursor control operations (e.g., LEFT UP) will not be decoded and are ignored.

CONTROL OPERATIONS	ALTERNATE SPELLING	DESCRIPTION
LOCK	LO	Lock keyboard
UNLOCK	UN	Unlock keyboard
BLOCK	BL	Initiates the block mode read
FORMAT	FO	Initiates the formatting mode write
HOME	HO	Cursor home
UP=nn	UP=nn	Cursor up
DOWN=nn	DO=nn	Cursor down
RIGHT=nn	RI=nn	Cursor right
LEFT=nn	LE=nn	Cursor left
NLINE=nn	NL=nn	New line
TAB=nn	TA=nn	Tab to next unprotected Field
CLEAR	CL	Erase all characters
ERASE	ER	Erase all unprotected characters
NOTE: nn represents a number from 01 - 99 (defaults to 01)		

FIGURE 3-1 MODEL 80 CONTROL OPERATORS

CREATING NEW FORMAT FILES USING EXISTING FILES

To create a new format file using an existing format file, enter the command listed below.

```
CREATE pool.file[,KEY='password'],USE=pool1.file1[,KEY='password']
```

If the USE file non-existent or null, the program displays an error message and the operator is requested to try again. If a password was required and none was entered or entered incorrectly, the program displays an error message.

The USE= file is read into the work file and the work file is decoded into a 1599 character screen image as it was originally entered and displayed. The data on the screen is ready to be modified using the procedures described in the CREATE command.

UPDATING EXISTING FORMAT FILES

To update an existing format file, enter the command listed below.

```
UPDATE pool.file[,KEY='password']
```

If the file does not exist or is null, the program displays an error message and the operator is requested to try again. An error message is displayed if the password is incorrect.

The file is written to the work file and the work file is decoded into a 1599 character screen image as it was originally entered and again displayed. The data on the screen is modified by following the procedures described in the CREATE command.

DISPLAYING FORMAT FILES

To display a format file, enter the command listed below.

```
DISPLAY pool.file[,KEY'password']
```

The program retrieves the file to be displayed and writes it to the work file. The screen is filled with 1599 Unprotected-Blank delimiters. The Format that is on the working file is executed as it would be in the user's program using IOCS. This allows unerased unprotected areas to be identified. The program returns to the beginning of the program if CTL key 5 is depressed, otherwise, the operator is requested to reject or approve the format. If the format is rejected, the program proceeds with the UPDATE logic.

APPENDICES

Appendix A ASSEMBLY ERROR MESSAGES

SUMMARY OF MODEL 80 DISPLAY MACRO MNOTES

IO80

MNOTE E,'IO80: INVALID FORMT PARAMETER-DEFAULT ASSUMED'

Meaning;	This error indicates the FORMT=NO operand was incorrectly specified. FORMT=YES was assumed, resulting in the code that performs the format operation to be included in the generated IOCS subroutines.

MNOTE E,'IO80: INVALID WRITE PARAMETER-DEFAULT ASSUMED'

Meaning;	This error indicates the WRITE=NO operand was incorrectly specified. WRITE=YES was assumed and the code to perform the write and control operations was included in the macro expansion.

MNOTE E,'IO80: INVALID READ PARAMETER-DEFAULT ASSUMED'

Meaning;	This error indicates the READ=NO operand was incorrectly specified. READ=YES was assumed and the code to perform the read operation was included in the macro expansion.

MNOTE E,'IO80: MISSING ERROR PARAMETER--SEE QSR ROUTINE'

Meaning;	This error indicates that neither ERROR=operand or the SRBR=operand were given, therefore, the address of the error routine generated by this operand is equal to '////' which will be flagged by Assembler II as an error when the _QSR subroutine is generated.

MNOTE E,'IO80: INVALID SRBR PARAMETER'

Meaning;	This error indicates that the SRBR operand was incorrectly specified. Therefore, the 10-Entry Device Table and the _SET80 subroutine were included in the macro expansion as though the SRBR operand had not been used.

MNOTE E'IO80: INVALID KEYWORD OPERAND'

Meaning; This error indicates that a keyword operand was spelled incorrectly.

SET80

MNOTE E,'SET80: MISSING PARAMETER'

Meaning; This error indicates that a required operand was missing.

MNOTE E'SET80: INVALID DEV/DROP PARAMETER'

Meaning; This error indicates that more than 10 device numbers were entered or that NO device numbers were entered.

MNOTE E'SET80: INVALID KEYWORD OPERAND'

Meaning; This error indicates that a keyword operand was spelled incorrectly.

M80IO

MNOTE E'M80IO: MISSING PARAMETER'

Meaning; This error indicates that a required operand is missing.

MNOTE E'M80IO: INVALID DEV PARAMETER'

Meaning; This error indicates that more than one device number was entered.

MNOTE E,'M80IO: INVALID KEYWORD OPERAND'

Meaning; This error indicates that a keyword operand was spelled incorrectly.

IOCB

MNOTE E,'IOCB: INVALID SHORT PARAMETER

Meaning;	That the Short operand was incorrectly entered, therefore, the Long IOCB was generated.

MNOTE E,'IOCB: MISSING PARAMETER

Meaning;	This error indicates no operands were given. A Long IOCB was generated with the IOCB fields left undefined.

MNOTE W,'IOCB: IOADR,IOLEN PARAMETERS NOT USED IN SHORT IOCB'

Meaning;	This warning statement indicates that the IOADR and IOLEN operands were given, and that they were not used because a Short IOCB was generated.

MNOTE W,'IOCB: INVALID DEVICE PARAMETER'

Meaning;	This warning message indicates that more than one digit was entered as a device number. It recognized only the first digit and stored it as the device number.

MNOTE,'IOCB: INVALID KEYWORD OPERAND'

Meaning;	This error message indicates a keyword operand was spelled incorrectly.

OPSTR

MNOTE E,'OPSTR: INVALID PARAMETER (xxxxx) --- IGNORED'

Meaning;	This error indicates that an invalid CONTROL operand parameter (xxxxx) was entered and ignored.

MNOTE S,'OPSTR: NO PARAMETERS----NO OPERATION STRING CREATED'

Meaning;	This is a severe error. Because no operands were given, no operation string was created.

MNOTE E'OPSTR: INVALID POOL.FILE NAME'

Meaning;	This error indicates that the pool and file name given in the FORMAT operand was not separated by a period. The Format Operator was generated with a null pool and file name.

MNOTE W,'OPSTR: WRITE STRING NOT ENCLOSED IN QUOTES'

Meaning;	This warning statement indicates that the write string given in the Write operand was not enclosed within quotes, however, the macro attempted to generate the proper operator.

MNOTE E'OPSTR: INVALID CONTROL PARAMETER (zzzzz) --- IGNORED'

Meaning;	This error conditions indicates an invalid parameter (zzzzz) was given in the Control operand. The Control character was not generated.

MNOTE W,'OPSTR: END PARAMETER OUT OF SEQUENCE'

Meaning;	This warning statement indicates that the operands following the END operand was ignored.

```
MNOTE W,'OPSTR: IO80 CALL SHOULD PRECEDE OPSTR CALL ---                X
AN INVALID OPERATOR COULD BE INCLUDED IN OPERATION STRING'
```

Meaning;	This error indicates the IO80 call did not precede the OPSTR call and it is not known whether all operators are valid. All operators were allowed.

IO80 ERROR ROUTINE

The user's routine is specified in the IO80 Macro Call with the operand ERROR=. It receives control when the _QSR subroutine recognizes a service request from a device which has no entry set in the 10-entry device table. The device number is stored in the field labelled QSRDEV. If the device is a Model 80, and if the user wishes to add this device to the system, then read the device (satisfying the service request). Call _SET80 to store an entry for DEV=QSRDEV. (See sample program in Appendix C.)

M80IO EXCEPTION ROUTINE

The user's routine is specified in the M80IO Macro Call with the operand EXCEP=. It is executed when one of the following errors occurs:

STATUS CODE	DESCRIPTION
1	An invalid IOCB was given.
2	A service request line was up and a Read Operator was not the first operator in the operation string.
3	An invalid operator was encountered in the operation string.
4	A fault status was returned from the Model 80.
5	Not used.
6	A write operation caused a page wrap around.
7	An invalid format file was encountered.
8	A flag or parity status was returned on a read to disc.
9	A fault status was returned on a read to disc.

The IOCB address is stored in index register 3. The following pertainent information is stored in that IOCB:

CORE REFERENCE LABEL	INFORMATION
_IOSC	Status code.
_IOCNT	The number of the last operator executed. (001 represents that the first operator was executed and so on.)
_IODEV	Device number.

SET80 ERROR ROUTINE

The user's routine is specified in the SET80 macro call with the operand, ERROR=. The _SET80 routine, before storing an entry for a device, queries that device to see if it is on-line. If the operand is omitted or ERROR=IGN is specified, _SET80 will not store a device entry for that device but will continue to store other entries or give a normal return. If the ERROR= operand was specified and if any status other than Normal is returned from that query, the error exit is taken. Therefore, the following conditions will cause the error routine to be executed:

1. Device is not on-line.

2. Device had service request line up.

   NOTE: The query has cancelled the service request.

The device number is stored in _STDEV. If the user wishes to continue execution of _SET80, then the entry point _STRST must be used. If the user wishes to know the cause of the error, the first instruction of the error routine must be a branch on condition code 4 then 1.

Appendix B ERROR MESSAGES

The formatting job may or may not take place at the CONO console device. If seperate, the following messages are displayed at one or both of the devices.

On CONO Logical Device

S)FORMAT: DISC ERROR x AT nnnnnn.

Description: While resolving program overlay disc addresses, a bad status was returned from the disc where x represents the error status returned and nnnnnn represents the disc address of the error.

Action Taken: Program is aborted; DMF Conversational Loader is loaded.

T)FORMAT: DEVICE x IS OFF-LINE.

Description: Format entry device x is off-line.

Action Taken: Program is aborted; DMF Conversational Loader is loaded.

A)READY DEVICE Dn.

Description: Disc drive n is off-line.

Action Required: Issue service request and press the enter key when the device selects. The device is not selected immediately because the formatting device (if any) may have responded to the message.

On Model 80 Display Unit (where the formatting is taking place)

ENTER Command Error Messages

INVALID COMMAND

Description: An invalid command was entered.

Action Required: Re-ent-er command.

INVALID pool.file NAME

Description: An invalid pool.file was entered.

Action Required: Re-enter command.

INVALID PASSWORD ENTRY

Description: The password was incorrectly entered.

Action Required: Re-enter ommand.

INVALID FORMAT FILE

Description: The file entered is not a file type '0', and/or it is not valid format file. A valid format file is one containing a valid operation string as defined in the Model 80 Display IOCS.

Action Required: Enter new command using another pool.file.

Create Error Messages

PARENTHESIS ARE INVALID DATA CHARACTERS.

Description: One of the characters entered as protected data was a parenthesis.

Action Required: Re-enter format.

AT LEAST ONE DATA CHARACTER MUST BE ENTERED TO CONSTITUTE A VALID FORMAT.

Description: The format read contained all delimiters and no data characters. The result is an invalid format.

Action Required: Re-enter format.

INVALID CONTROL STRING: PAGE WRAP ON WRITE

Description: An error has been made in either the pre-format or post-format control operations that has caused the writing of the format to wrap around.

Action Required: Re-enter control operations.

INVALID READ

Description: The enter key was depressed when the cursor was in the home position.

Action Required: Re-enter format and depress the enter key at the end of the format.

ONLY FORMAT FILES CREATED BY THE MODEL 80 FORMAT UTILITY ARE VALID AS UPDATE OR USE FILES.

Description: While decoding the format, an unrecognizable character or string of characters was encountered.

Action Required: Enter new command.

DELIMITER ERROR MESSAGES

INVALID DELIMITERS: EACH OF FIVE DELIMITERS MUST BE UNIQUE

Description: Five delimiters were not entered, or those five were not unique characters.

Action Required: Enter five unique characters.

EXAMPLE OF MULTI-TERMINAL ENVIRONMENT PROGRAM

```
          FILE NAME   TYPE    CHANGE    DATE OF     DATE OF        DESCRIPTION
                              LEVEL    EXPIRATION   CREATION

            SAMPLE     S       000       -  -        -  -        IOCS SAMPLE


RECORD NO.

   1            TITLE  'MULTI-TERMINAL ENVIRONMENT  IOCS EXAMPLE'
   2            NORMAL
   3            ORG    1000
   4            I080   ERROR=NODEV                     * INCLUDE IOCS ROUTINE *
   5     SIOCB  IOCB   SHORT=YES,OPSTR=ST1             * DEFINE IOCB *
   6     ST1    OPSTR  CONTROL(CLEAR),WRITE('GOOD MORNING'),                    X
   7                   CONTROL(NLINE,UNLOCK),END       * DEFINE OP STRING *
   8     FIOCB  IOCB   IOADR=BUFFER,OPSTR=ST2          * DEFINE IOCB *
   9     ST2    OPSTR  READ,FORMAT(POOL1.FILE1),END    * DEFINE OP STRING *
  10     LIOCB  IOCB   IOADR=BUFFER,OPSTR=ST3          * DEFINE IOCB *
  11     ST3    OPSTR  READ,END                        * DEFINE OP STRING *
  12     BUFFER DM     C1600                           * BUFFER AREA *
  13     *      ***********************************************************
  14     *      * SET A ENTRY IN 10-ENTRY DEVICE TABLE FOR ALL DEVICES ON LINE *
  15     *      * ALSO SEND GOOD MORNING MESSAGE TO ALL DEVICES ON-LINE.     *
  16     *      ***********************************************************
  17     START  SET80  ENTRY=ENTRY1,IOCB=FIOCB,ERROR=IGN,DEV=ALL,EXEC=SIOCB
  18     *      <<<<<  (OPTIONAL) USER'S PROGRAM INITIALIZATION  >>>>>
  19     BEGIN  QSR                            GO TO SERVICE REQUEST ROUTINE
  20     *      <<<<<  (OPTIONAL) USER'S BACKGROUND PROCESSING HERE >>>>>
  21            BSW    BEGIN                   LOOP UNTIL SERVICE REQUEST
  22     *
  23     ENTRY1 EQU    *                       FIRST ENTRY POINT FROM QSR
  24            M80IO  EXCEP=MERROR            * IOCB AND DEV # SET BY QSR *
  25     *      ***********************************
  26     *      * MODIFY ENTRY POINT FOR THIS DEVICE *
  27     *      ***********************************
  28            SET80  ENTRY=ENTRY2,IOCB=LIOCB,DEV=QSRDEV,ERROR=IGN
  29            B      BEGIN                   RETURN TO PROCESS ANOTHER DEVICE
  30     *
  31     ENTRY2 EQU    *                       SECOND ENTRY POINT FROM QSR
  32            M80IO  EXCEP=MERROR            * IOCB AND DEV # SET BY QSR *
  33     *      <<<<   USER'S DATA PROCESSING ROUTINE >>>>>
  34            B      BEGIN                   RETURN TO PROCESS ANOTHER DEVICE
  35     *
  36     MERROR EQU    *                       PROCESS ERROR FROM M80IO
  37     *      <<<<<  USER'S HANDLING OF VARIOUS ERRORS >>>>>
  38            B      BEGIN                   RETURN TO PROCESS OTHER DEVICES
  39     *
  40     NODEV  EQU    *                       NO-ENTRY-IN-TABLE ERROR
  41     *                                     DEV # AND IOCB MUST BE SPECIFIED
  42            M80IO  IOCB=LIOCB,DEV=QSRDEV,EXCEP=MERROR    READ DEVICE
  43            SET80  ENTRY=ENTRY1,IOCB=FIOCB,DEV=QSRDEV
  44            B      BEGIN                   RETURN TO PROCESS OTHER DEVICES
  45     *
  46            EXEC   START
  47            END

FILE SAMPLE IN POOL MIS    CONTAINS     47 SECTORS
```

SAMPLE SCREEN ENTRY

Model 80 CRT Coding Form

```
 1                                MLB CORPORATION
 2                                2470 LEE AVENUE
 3                                MARIN, CA. 90000
 4                                (415) 973-7876
 5
 6 SHIP TO:^^^^^^^^^^^^^^^          ATTN:^^^^^^^^^^^^^^^        DATE:^^^^^
 7         ^^^^^^^^^^^^^^^          DEPT:^^^^^^^^^^^^^^^     INVOICE:^^^^^
 8         ^^^^^^^^^^ZIP^^^^^       SECT:^^^^^^^^^^^^^^^      VENDOR:^^^^^
 9
10 NUMBER      ITEM      QUANTITY        UNIT PRICE           BILLING RATE
11
12 ^^^-^^      ^^^-^       ^^^^          $^^,^^^.^^            $^^^,^^^.^^
13 ^^^-^^      ^^^-^       ^^^^          $^^,^^^.^^            $^^^,^^^.^^
14 ^^^-^^      ^^^-^       ^^^^          $^^,^^^.^^            $^^^,^^^.^^
15 ^^^-^^      ^^^-^       ^^^^          $^^,^^^.^^            $^^^,^^^.^^
16 ^^^-^^      ^^^-^       ^^^^          $^^,^^^.^^            $^^^,^^^.^^
17 ^^^-^^      ^^^-^       ^^^^          $^^,^^^.^^            $^^^,^^^.^^
18 ^^^-^^      ^^^-^       ^^^^          $^^,^^^.^^            $^^^,^^^.^^
19
20             TOTAL                $^^^,^^^,^^^.^^        $^^^,^^^,^^^.^^
```

Appendix D SUMMARY OF MODEL 80 DISPLAY IOCS MACRO INSTRUCTIONS

[label]	IO80	[FORMT=$\begin{Bmatrix}\underline{\text{YES}}\\ \text{NO}\end{Bmatrix}$][,WRITE=$\begin{Bmatrix}\underline{\text{YES}}\\ \text{NO}\end{Bmatrix}$][,READ=$\begin{Bmatrix}\underline{\text{YES}}\\ \text{NO}\end{Bmatrix}$] $\begin{Bmatrix}\text{,ERROR=label}\\ \text{,SRBR=(label1,label2)}\end{Bmatrix}$
[label]	SET80	DEV=$\begin{Bmatrix}\text{ALL}\\ \text{QSRDEV}\\ \text{(n,n,....,n)}\\ \text{n}\end{Bmatrix}$,ENTRY=label,IOCB=label[,ERROR=$\begin{Bmatrix}\underline{\text{IGN}}\\ \text{label}\end{Bmatrix}$][,EXEC=label]
[label]	SET80	DROP=$\begin{Bmatrix}\text{ALL}\\ \text{QSRDEV}\\ \text{(n,n,....,n)}\\ \text{n}\end{Bmatrix}$
[label]	QSR	(no operands)
[label]	M80IO	EXCEP=label[,IOCB=label][,DEV=$\begin{Bmatrix}\text{QSRDEV}\\ \text{n}\end{Bmatrix}$]
[label]	IOCB	OPSTR=label,IOADR=label[,IOLEN=nnnn][,DEV=n][,SHORT=$\begin{Bmatrix}\text{YES}\\ \underline{\text{NO}}\end{Bmatrix}$]
[label]	IOCB	SHORT=YES,OPSTR=label[,DEV=n]
[label]	OPSTR	[$\begin{Bmatrix}\text{FORMAT(pool.file)}\\ \text{F(pool.file)}\end{Bmatrix}$][, $\begin{Bmatrix}\text{READ}\\ \text{READ(0000)}\\ \text{R}\\ \text{R(0000)}\\ \text{READ(entry1,entry2)}\\ \text{R(entry1,entry2)}\\ \text{READ(label)}\\ \text{R(label)}\end{Bmatrix}$][, $\begin{Bmatrix}\text{WRITE}\\ \text{WRITE(0000)}\\ \text{W}\\ \text{W(0000)}\\ \text{WRITE('data string')}\\ \text{W('data string')}\end{Bmatrix}$] [, $\begin{Bmatrix}\text{CONTROL}\\ \text{CNTL}\\ \text{C}\end{Bmatrix}$ $\begin{Bmatrix}\text{(string of parameters p,p,....,p)}\\ \text{('control character string')}\end{Bmatrix}$][, $\begin{Bmatrix}\text{NOEND}\\ \underline{\text{END}}\\ \text{E}\end{Bmatrix}$]

MODEL 80 DISPLAY IOCS

Reference Manual

Publication No. 40-700

We produce manuals for you, and we want you to find them useful and informative. That's our job.

So we're asking you to help us furnish you with the best possible publications. Please take a few minutes to answer the following questions. Add any comments you wish. If you desire a reply to any question, be sure to include your name and address.

Thank you.

DETACH HERE

- Does this manual meet your needs? Yes ☐ No ☐
  If not, what additional information would be of help to you?

- Can you find what you're looking for quickly and easily? Yes ☐ No ☐
  How can the organization be improved?

- Is the material easy to read and to understand? Yes ☐ No ☐
  Are there enough illustrations to support the text? Yes ☐ No ☐
  Comments

- Did you find any errors or ambiguities in the manual? Yes ☐ No ☐
  If yes, please cite page, line, and/or figure number with your comments.

- Other comments.

- What is your relationship to the product described?
  - ☐ Operator.
  - ☐ Programmer.
  - ☐ Other (please specify)

SINGER
BUSINESS MACHINES

STAPLE STAPLE

FOLD BACK

FIRST CLASS
PERMIT No. 320
San Leandro, Calif.

BUSINESS REPLY MAIL
No postage stamp necessary if mailed in the United States

POSTAGE WILL BE PAID BY

SINGER BUSINESS MACHINES
2350 Washington Ave.
San Leandro, California 94577

Attn: Customer Technical Publications,
Department 753

FOLD BACK

SINGER
BUSINESS MACHINES

PUBLICATION NO. 40-700