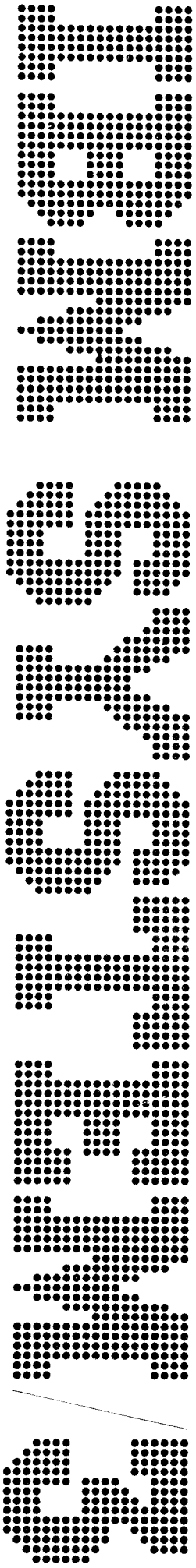


IBM System/3
Multiline/Multipoint Binary Synchronous Communications Reference Manual

Program Numbers:

5702-SC1 Model 10

5704-SC1 Model 15

5704-SC2 Model 15

5705-SC1 Model 12

GC21-7573-4
S3-30

Fifth Edition (December 1976)

This is a major revision of, and replaces, GC21-7573-3 and Technical Newsletters GN21-7775, GN21-5279, and GN21-5363. Changes are indicated by a vertical line at the left of the change. New or extensively revised illustrations are indicated by the symbol ● to the left of the caption.

This edition applies to Program Number 5702-SC1 (version 10 and modification 00) of IBM System/3 Model 10 Disk System, Program Number 5704-SC1 (version 01 and modification 00) of IBM System/3 Model 15, Program Number 5704-SC2 (version 01 and modification 00) of IBM System/3 Model 15, and Program Number 5705-SC1 (version 02 and modification 00) of IBM System/3 Model 12, and to all subsequent versions and modifications until otherwise indicated in new editions or technical newsletters.

Changes to the information herein are made periodically. Before using this publication to operate an IBM system, refer to the latest *IBM System/3 Bibliography*, GC20-8080, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A Readers Comments form is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901.

This manual provides the programming information required to use the Multiline/Multipoint Feature (MLMP) with System/3 Model 10, Model 12, or Model 15 binary synchronous communications programs.

On the Model 10 and Model 12, MLMP is a feature of the System Control Programming (5702-SC1, Features 6030 and 6031). On the Model 15, MLMP is included in the base System Control Programming (5704-SC1 or 5704-SC2).

Hereafter, the terms "MLMP" and "System/3" should be understood as applying to the System/3 Model 10 Disk System, the Model 12, and the Model 15, unless qualified by "Model 10 and Model 12 only" or "Model 15 only."

This reference manual is intended for applications programmers who are familiar with:

- Basic telecommunications concepts and practices
- IBM System/3 Basic Assembler language
- IBM System/3 Model 10 Disk System, Model 12, or Model 15

The manual describes MLMP and the functions of MLMP, the System/3 MLMP macro instructions, and the MLMP diagnostics and diagnostic aids. The book also lists MLMP system requirements and considerations. Appendixes contain a list of MLMP considerations unique to certain terminals, examples of coded MLMP macro instructions and a sample program, data area formats, BSC line control characters and codes, and a macro instruction summary.

As noted in the text, many of the cross references in this manual address index entries.

SYSTEM/3 MODEL 8

The System/3 Model 8 is supported by System/3 Model 10 Disk System control programming and program products. The facilities described in this publication for the Model 10 are also applicable to the Model 8, although the Model 8 is not referenced. However, the Integrated Communications Adapter (ICA) is only available on the Model 8. If you have either the ICA or local display adapter, it is always designated as BSCA line 2. Therefore, you must specify line 2 whenever the ICA or local display adapter is used, or enter the BSCA OCL statement (// BSCA LINE-2) at execution time. It should be noted that not all devices and features which are available on the Model 10 are available on the Model 8. Therefore, Model 8 users should be familiar with the contents of *IBM System/3 Model 8 Introduction,* GC21-5114.

Prerequisite Publications

- *General Information: Binary Synchronous Communications,* GA27-3004
- *IBM System/3 Models 8, 10, 12, and 15 Components Reference Manual,* GA21-9236

Related Publications

- *IBM System/3 Basic Assembler Reference Manual,* SC21-7509
- *IBM System/3 Models 4, 6, 8, 10, and 12 System Generation Reference Manual,* GC21-5126 or *IBM System/3 Model 15 System Generation Reference Manual,* GC21-7616
- *IBM System/3 Model 8 Operator's Guide,* GC21-7634, *IBM System/3 Model 10 Disk System Operator's Guide,* GC21-7508 or *IBM System/3 Model 15 Operator's Guide,* GC21-5075
- *IBM System/3 Model 10 Disk System Control Programming Reference Manual,* GC21-7512; *IBM System/3 Model 15 System Control Programming Reference Manual* (5704-SC1), GC21-5077, or *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual,* (5704-SC2), GC21-5162
- *IBM System/3 Model 10 Disk System Halt Guide,* GC21-7540 or *IBM System/3 Model 15 System Messages,* GC21-5076
- *IBM System/3 Disk System Control Programming Macros Reference Manual,* GC21-7562 or *IBM System/3 Model 15 System Control Programming Macros Programming Reference Manual,* GC21-7608
- *IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561
- *IBM System/3 Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual,* GC21-7560
- *IBM System/3 Model 12 System Control Programming Reference Manual,* GC21-5130
- *IBM System/3 Model 12 Operator's Guide,* GC21-5144
- *IBM System/3 Model 12 User's Guide,* GC21-5142
- *IBM System/3 Models 8, 10, 12, and 15 Components Reference Manual,* GA21-9236
- *IBM System/3 Model 12 Halt Guide,* GC21-5145
- *IBM System/7 Systems Summary,* GA34-0002
- *IBM System/7 Binary Synchronous Communications Module (RPG), Programming Guide and Reference Manual,* SC34-1510
- *IBM System/7 Teleprocessing Multiplexor "TPMM" Programming Guide and Reference Manual Supporting RPQ D08011,* SC34-1506
- *System Components: IBM 2770 Data Communication System,* GA27-3013
- *IBM 2780 Data Transmission Terminal: Component Description,* GA27-3005
- *Component Description: IBM 2972 Models 8 and 11 General Banking Terminal System,* GL27-3020
- *IBM 3270 Information Display System Component Description,* GA27-2749
- *IBM 3735 Programmer's Guide,* GC30-3001
- *IBM Systems 3735 Support Program Coding Manual,* GC21-5096
- *IBM 3600 Finance Communication System Programmer's Guide and Component Description,* GC27-0004

Contents

Multiline/Multipoint (MLMP) is a binary synchronous communications (BSC) feature of System/3. MLMP provides the assembler programmer access to the BSC I/O routines that support the Binary Synchronous Communications Adapter (BSCA) as an I/O device. For a description of the BSCA, see the appropriate components reference manual for your system listed in the *Preface.*

MLMP enables the assembler programmer to transmit and receive binary synchronous data over two telecommunications lines simultaneously (each line requires a BSCA). The two lines can be used in the same program or can be used independently in separate program levels (on Model 10 or Model 12 that has the dual programming feature installed) or in the program partitions (on Model 15). The lines can be nonswitched or switched. Figure 1 gives examples of the line configurations possible with MLMP.

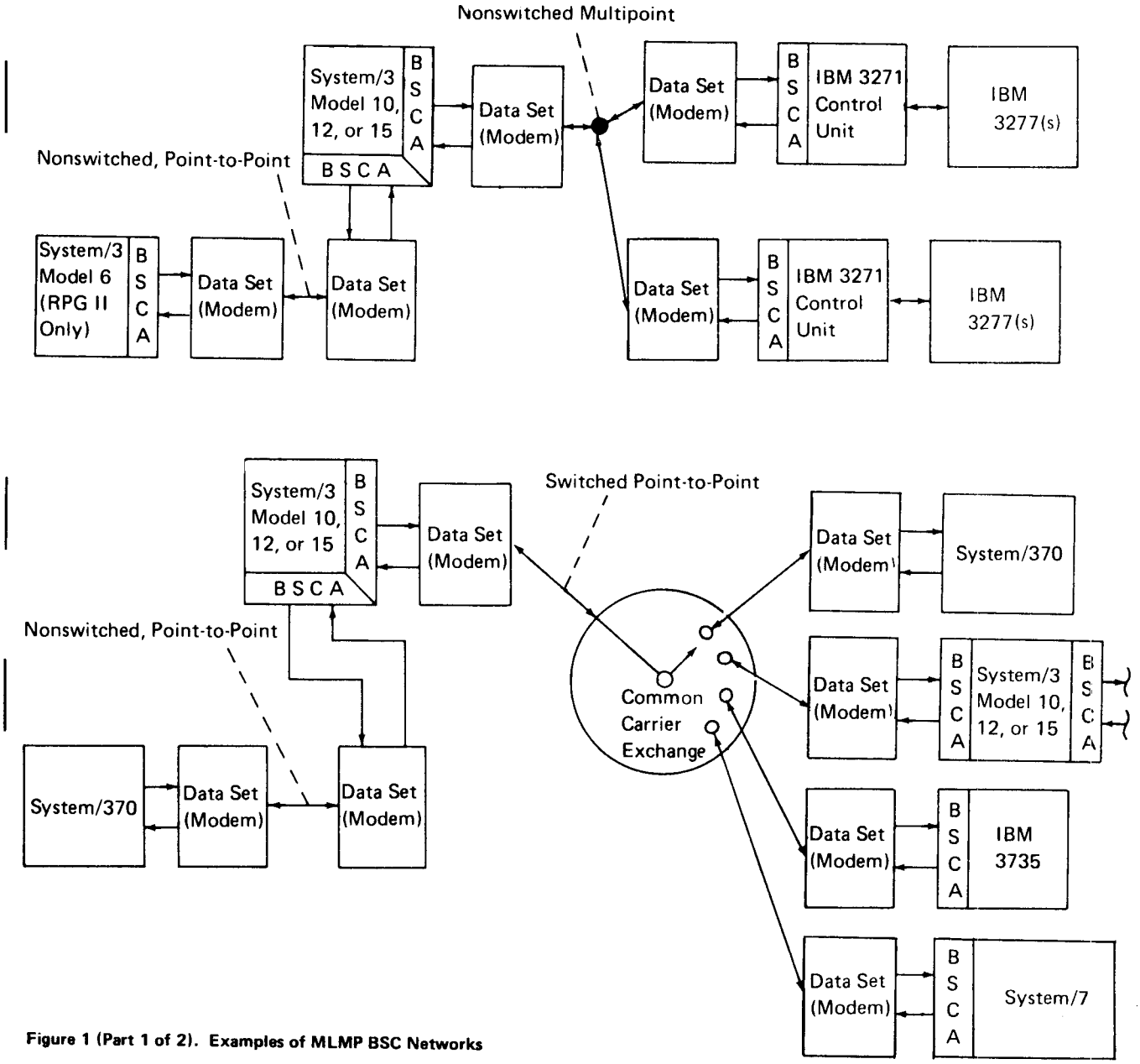
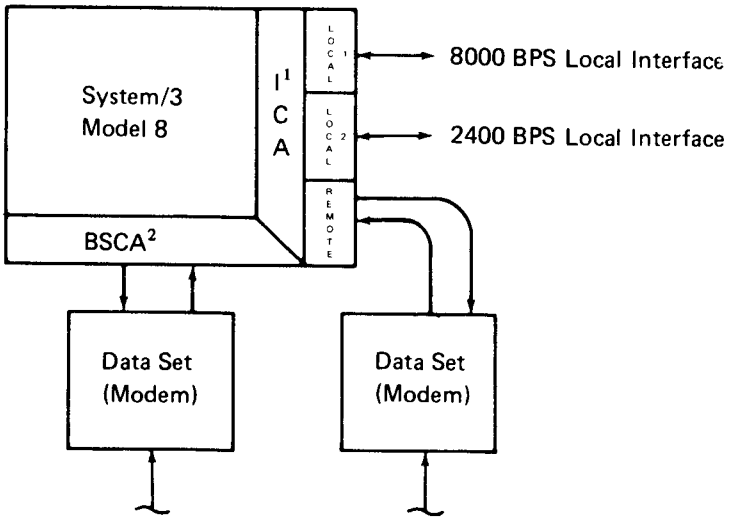


Figure 1 (Part 1 of 2). Examples of MLMP BSC Networks



[1] The Integrated Communications Adapter (ICA) must be addressed as BSCA line 2. The manual ICA switch can have only one interface active at any one time.

[2] See Figure 1 (Part 1 of 2) for examples of BSCA line configurations.

Figure 1 (Part 2 of 2). Examples of MLMP BSC Networks

The MLMP user specifies the functions of MLMP I/O routines by using System/3 assembler macro instructions (see Chapter 2). The IBM System/3 Macros Feature expands these macro instructions into linkage to MLMP routines.

Linkage to the MLMP routines is assembled as part of the user's program. The IBM System/3 Overlay Linkage Editor is then used to incorporate the MLMP routines in the user's object program.

Figure 2 shows the relation of MLMP to the macro processor, the overlay linkage editor, and user programs.

Figure 3 shows the relationship of a user MLMP program to MLMP I/O routines, BSC DTFs, BSC IOBs and buffers, and the BSCA.

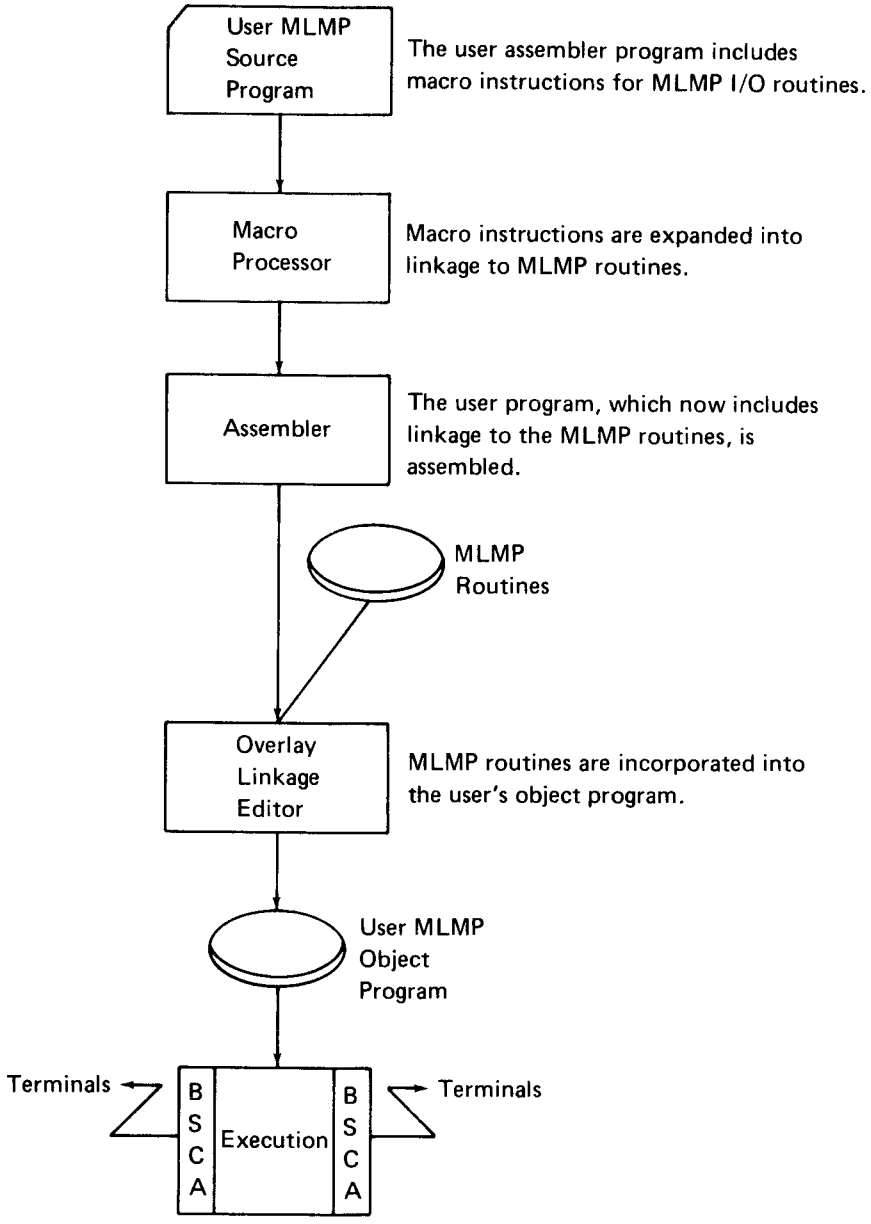
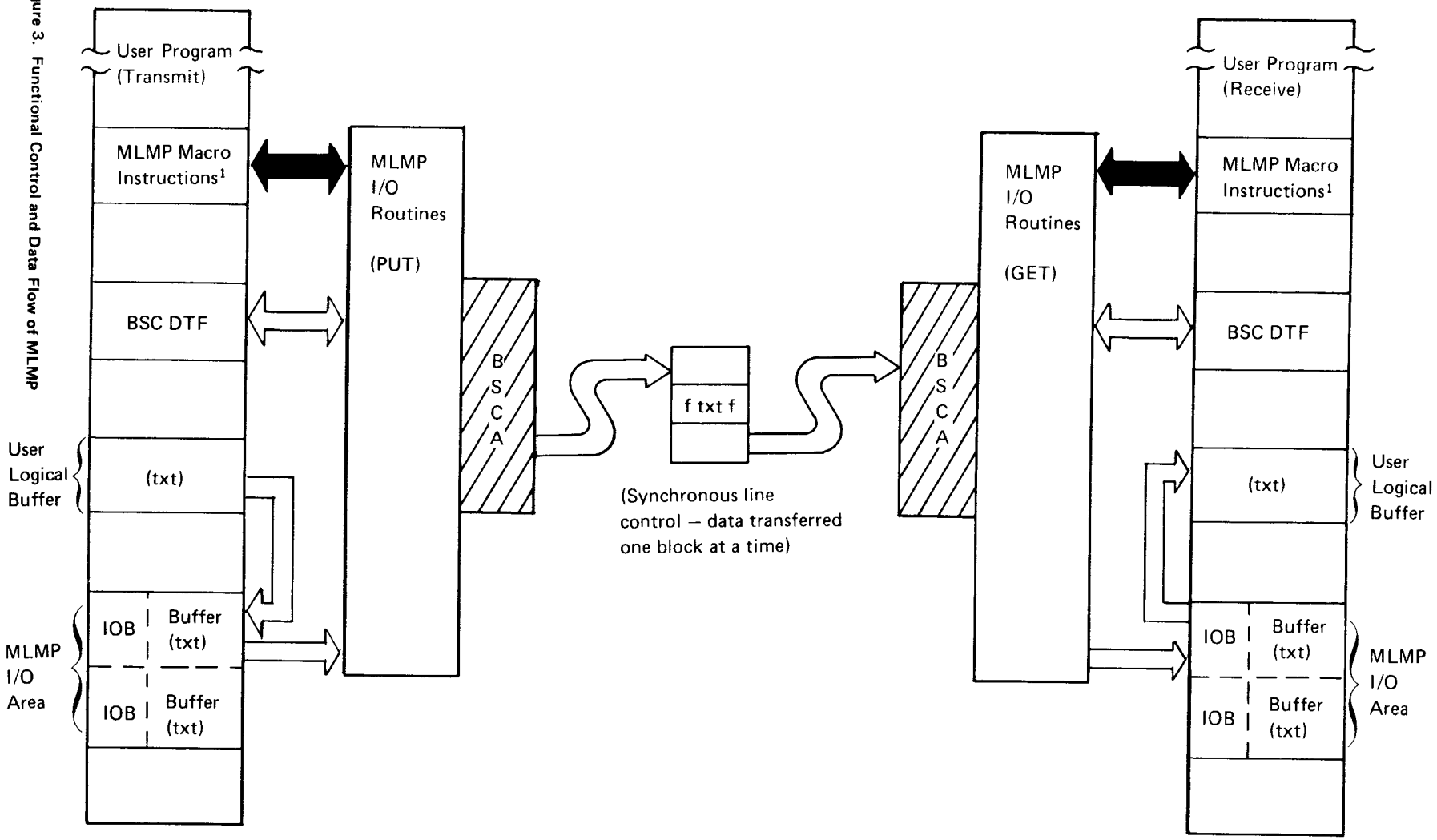


Figure 2. Generation of a User MLMP Object Program

Figure 3. Functional Control and Data Flow of MLMP



[1] MLMP macro instructions are described in Chapter 3.

Telecommunications Lines Supported

Following are the terminals and line connections supported by MLMP. For considerations unique to using a given terminal with MLMP, see Appendix A.

Nonswitched Point-to-Point

MLMP supports point-to-point nonswitched connections with central processing units programmed according to the conventions described in *General Information: Binary Synchronous Communications,* GA27-3004. MLMP also supports point-to-point nonswitched connections with the:

- IBM System/7 Teleprocessing Multiplexor (TPMM) as described in *IBM System/7 Teleprocessing Multiplexor "TPMM" Programming Guide and Reference Manual Supporting RPQ D08011,* SC34-1506
- IBM 2770 Data Communication System as described in *System Components: IBM 2770 Data Communication System,* GA27-3013
- IBM 2780 Data Transmission Terminal as described in *IBM 2780 Data Transmission Terminal: Component Description,* GA27-3005

Multipoint

MLMP supports System/3 as a control station and as a multidropped terminal. As a control station, System/3 can exchange binary synchronous data with:

- IBM System/3
- IBM System/7 as described in *IBM System/7 Binary Synchronous Communications Module Programming Guide and Reference Manual,* SC34-1510
- IBM System/7 Teleprocessing Multiplexor (TPMM) as described in *IBM System/7 Teleprocessing Multiplexor "TPMM" Programming Guide and Reference Manual Supporting RPQ D08011,* SC34-1506
- IBM 2770 Data Communication System as described in *System Components: IBM 2770 Data Communication System,* GA27-3013
- IBM 2780 Data Transmission Terminal as described in *IBM 2780 Data Transmission Terminal: Component Description,* GA27-3005
- IBM 2972/2980 Banking Terminal System (supported in the United States only) as described in *Component Description: IBM 2972 Models 8 and 11 General Banking Terminal Systems,* GL27-3020
- IBM 3270 Information Display System as described in *IBM 3270 Information Display System Component Description,* GA27-2749
- IBM 3735 Programmable Terminal as described in *IBM 3735 Programmer's Guide,* GC30-3001
- IBM 3600 Finance Communication System (Model 15 only) as described in *IBM 3600 Finance Communication System Programmer's Guide and Component Description,* GC27-0004

When System/3 is a control station, MLMP does not support intermixing of terminals on one line.

Note: For information on how to generate IBM 3735 form descriptor programs (FDPs) on System/3, see *IBM System/3 Model 10 Disk System 3735 Application Package Coding Manual,* GC21-5096. A conversion routine (FDP/Convert) is provided with MLMP to convert FDPs generated on OS or DOS to a format suitable for transmission from Model 10 or Model 12 to a 3735. For a description of FDP/Convert, see index entry *FDP/Convert.*

Switched Point-to-Point

MLMP supports point-to-point switched connections with the following:

- Central processing units programmed according to the conventions described in *General Information: Binary Synchronous Communications,* GA27-3004
- IBM System/7 Teleprocessing Multiplexor (TPMM) as described in *IBM System/7 Teleprocessing Multiplexor "TPMM" Programming Guide and Reference Manual Supporting RPQ D08011,* SC34-1506
- IBM 2770 Data Communication System as described in *System Components: IBM 2770 Data Communication System,* GA27-3013
- IBM 2780 Data Transmission Terminal as described in *IBM 2780 Data Transmission Terminal: Component Description,* GA27-3005

- IBM 3275 Display Station as described in *IBM 3270 Information Display System Component Description,* GA27-2749

- IBM 3735 Programmable Terminals as described in *IBM 3735 Programmer's Guide,* GC30-3001

- IBM System/7 as described in *IBM System/7 Systems Summary,* GA34-0002

For switched connections, MLMP supports autocall (United States only), manual call, autoanswer, manual answer, and the exchange of station identification characters.

Note: For information on how to generate IBM 3735 form descriptor programs (FDPs) on System/3, see *IBM System/3 Model 10 Disk System 3735 Support Program Reference Manual,* GC21-5096. A conversion routine (FDP/Convert) is provided with MLMP to convert FDPs generated on OS or DOS to a format suitable for transmission from Model 10 to a 3735. For a description of FDP/Convert, see index entry *FDP/Convert.*

Functions

The following program functions are available to the MLMP user:

1. Receive only (receive input data from a remote terminal).

2. Receive with transmittal of conversational reply (receive input data from a remote terminal and, when required, transmit data as an acknowledgement).

3. Transmit only (transmit data to a remote terminal).

4. Transmit with reception of conversational reply (transmit data to a remote terminal and, when required, receive data as an acknowledgement).

5. Transmit and receive—no conversational reply. Four modes of operation are possible:

   a. Transmit a file, then receive another file.
   b. Receive a file, then transmit another file.
   c. Transmit records of a file interspersed with receiving records of another file (receive RVI).
   d. Receive records of a file interspersed with transmitting records of another file (transmit RVI).

Depending on the terminal used and whether or not data is exchanged in conversational mode, records transmitted and received can be fixed length, variable length, or spanned (one record can occupy space in two contiguous blocks). Data can be exchanged in EBCDIC (Extended Binary Coded Decimal Interchange Code) or in ASCII (American National Standard Code for Information Interchange), depending on the BSCA used. Data translation is supported by translate tables generated with macro instructions. EBCDIC transparency and ITB (Intermediate Block Checking) are supported by MLMP.

MLMP does not transmit leading graphics. MLMP can receive leading graphics, but does not pass them to the user.

During program execution, MLMP automatically does the following:

- Blocks and deblocks data as required.

- Moves user data from the user's logical buffer to the telecommunications I/O buffers when sending data (PUT requests).

- Moves user data from the telecommunications I/O buffers to the user's logical buffer when receiving data (GET requests).

- Inserts and removes data-link control characters as necessary.

MLMP provides error recovery, error recording and, at user's request, online test (OLT). A trace module and a dump routine are also provided with MLMP.

Multiple Line Terminal Adapter

The BSCA and the Multiple Line Terminal Adapter (MLTA) can be used concurrently on the disk system. For information regarding MLTA, see *IBM System/3 Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual,* GC21-7560.

You inform MLMP of the functions your binary synchronous communications program requires by using System/3 macro instructions.

Description

A System/3 macro instruction causes a specified sequence of assembler source instructions to be generated. The format of a System/3 macro instruction is:

1		8		14	72
Name		Operation		Operands	
symbol or blank	b	macro name	b	no operands or one or more operands separated by commas	

Name

If you specify a name, it is assigned to the generated sequence of assembler instructions. The name becomes the symbolic address of the first byte of code generated by the macro instruction and it can be used to reference the code – that is, to modify the code or to branch to the code.

A name:

- Must begin with an alphabetic character in position 1 of the Assembler Coding Form
- Can be from one to six alphameric characters in length
- Must contain no special characters or blanks

Note: System/3 macro instructions generate labels beginning with the dollar sign ($). If you also define labels beginning with $ when you use System/3 macro instructions, duplicate labels may result.

Operation

The operation entry is the mnemonic operation code of the macro instruction. Each operation entry must begin in position 8 of the Assembler Coding Form.

Operands

Operands qualify the operation by specifying functions to be performed and data to be modified. Each operand consists of a keyword and a parameter joined by a dash. Keywords and parameters are predefined symbols available for use with individual macro instructions. You select keywords and parameters according to the rules that apply to the macro instruction you are writing.

- The first operand must begin in position 14 of the Assembler Coding Form.
- Operands must be separated by commas.
- Blanks are not permitted between operands coded on the same line.
- Blanks are not permitted between keywords and parameters.
- Operands cannot be specified beyond position 71.
- Operands can be written in any order.

Continuation Lines

The number of operands involved in some macro instructions may require more than one line of coding. If continuation is required, column 72 must contain a character and the last operand must be followed by a comma. An operand cannot be divided and continued on the next line. The operands of the continued field must begin in column 14. For an example of continuation coding, see Figure 4.

Comments

Comments can be placed after the last operand in a line if the comment is separated from the operand or comma by a blank. If a macro instruction has no operands, comments can be placed in the operand field if position 14 is left blank (Figure 4).

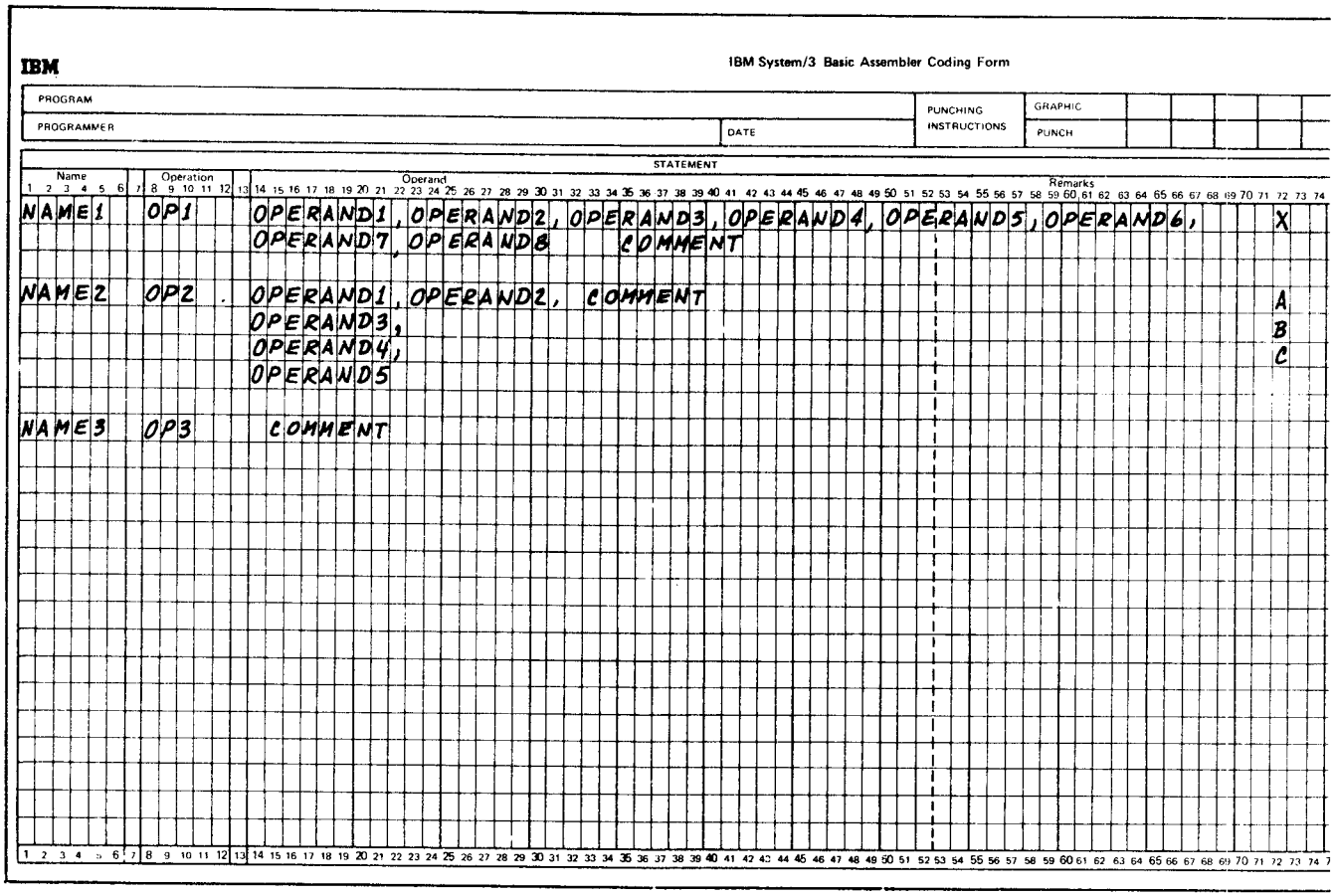


Figure 4. Sample of Continued Macro Instructions and Comments

Conventions

Certain symbols are used in this manual to abbreviate the descriptions of macro instructions.

1	8	14
[name]	$OP	KEYWORD–A/B/C

BRACKETS [ ] indicate an optional entry.

SLASHES / separate entries when a choice exists.

UNDERSCORE _ indicates the default if a parameter is not specified.

UPPER CASE LETTERS indicate words or abbreviations that must be entered as shown.

LOWER CASE LETTERS indicate information you must supply.

Each MLMP macro instruction is described in detail in Chapter 3. All MLMP macro instructions are summarized in a chart in Appendix E. The chart also gives the approximate length of code generated by each macro instruction.

Every MLMP program you write must accomplish three major functions:

1. Prepare BSC files for reception and/or transmission of data.
2. Initiate the transfer of data: receive and/or transmit.
3. Terminate the transfer of data.

Before programming these functions, however, note that one EXTRN must be defined in every MLMP program ($$BSMS). Other required EXTRNs are generated by the MLMP macro instructions when MLMP programs are assembled.

PREPARING FOR DATA TRANSFER

Preparing for data transfer always includes the following three steps:

1. Generate labels and equates for use by the various macro instructions and generate field displacements and labels for the BSC DTFs. Common labels and equates are generated by the $COMN macro instruction when your MLMP program is assembled. BSC DTF field displacements and labels are generated by the $DTOB macro instruction when your MLMP program is assembled.
2. Prepare BSC data files. Each BSC file must be:
   a. Defined ($DTFB macro instruction)
   b. Allocated ($ALOC macro instruction)
   c. Opened ($OPEN macro instruction)
3. Create a checklist to check for I/O completion. Entries in the checklist are created by the $CKL macro instruction.

If your station is a control station, you must create a polling/addressing list. Entries in a polling/addressing list are created by the $POLB macro instruction. Entries in a polling list can be activated or deactivated by using the $BCPL macro instruction (stations identified in active entries in a polling list are polled, stations identified in inactive entries are not polled). The $CHGB macro instruction can be used to generate the parameter list required by $BCPL.

Control stations must also provide space for logging terminal statistics. The $LOGB macro instruction can be used to allocate space for the Terminal Statistics Logging Area.

If you are using a switched answer line, you can create a list containing the station identification sequences (station IDs) your station will accept from a remote terminal. Entries in the switched ID list are created by the $SWIB macro instruction. The $BCSW macro instruction can be used to activate or deactivate entries in the list. (If an entry is inactive, your station will not accept the station ID given in that entry.) The $CHGB macro instruction can be used to generate the parameter list required by $BCSW.

If data in your BSC files requires translation, either before it is transmitted or after it is received, you must provide for data translation by constructing translate tables ($TRTB macro instruction for EBCDIC/ASCII tables) and generating a translate parameter list ($TRL macro instruction). When you want to translate data, generate the interface to the IBM-provided translate routine ($TRAN macro instruction).

Note: If you are transmitting translated data or receiving data that must be translated, be sure you have given the following information in the correct data code (EBCDIC or ASCII):

1. Polling and/or addressing characters if you are a control or tributary station.
2. Station identification sequences if you are on a switched line.
3. Online test parameter lists (generated by the $RFTL macro instruction) and online test messages if you intend to request online tests (see index entry *online test*).

Generate Common Equates ($COMN)

The $COMN macro instruction generates labels and equates used by System/3 macro instructions. This macro instruction must be issued once in every assembler program containing System/3 macro instructions.

$COMN Macro Instruction Format

	$COMN	

Generate BSC DTF Displacements and Labels ($DTOB)

BSC DTF field displacements and labels must be defined once in every MLMP program. The $DTOB macro instruction generates field displacements and associated labels for BSC DTFs. These displacements and labels, as well as BSC DTF byte and bit definitions and a definition of MLMP completion codes (see index entry *completion code*), are included in your assembly listing.

The generated BSC DTF labels are used by the code your MLMP macro instructions generate and you can use the labels to reference fields in the BSC DTFs.

See index entry *BSC DTF* for the labels generated by $DTOB.

$DTOB Macro Instruction Format

	$DTOB	

Define the File for BSC ($DTFB)

The $DTFB macro instruction generates a BSC DTF. More than one BSC file can be defined for each telecommunications line. See Appendix C for the format of the BSC DTFs. See also index entry *open BSC files ($OPEN).*

$DTFB Macro Instruction Format

[name]	$DTFB	RECL–decdig,BLKL–decdig, RCAD–address,FTYP–RCV/TSM [,BUFST–address,BUFEND–address] [,BUFNO–decdig] [,CODE–E/A] [,LINE–1/2] [,UP–binary/0] [,CHN–name] [,CONV–Y/N] [,ITB–Y/N] [,TRANSP–Y/N] [,RVIADR–address,RVIMSK–hex] [,DLYCT–decdig] [,TYPE–PP/MP/CS/AC/MC/AA/MA] [,TERMAD–hex] [,AUTORS–Y/N] [,LISTAD–address,ERRLOG–address] [,POLRES–Y/N] [,LIMIT–decdig] [,DIAL–address,DIALCT–decdig] [,RCVID–address,RCVCT–decdig/ ,SWLIST–Y/N] [,SNDID–address,SNDCT–decdig] [,SPAN–Y/N] [,RECSEP–hex] [,ERRCT–decdig]

name

If a name is specified, it is assigned to the first byte of the generated BSC DTF.

RECL—decdig

Specifies, in decimal, the maximum record length for this file, excluding line control characters. Record length is limited by available storage and terminal characteristics such as those listed in Appendix A. See index entry *$DTFB considerations.*

BLKL—decdig

Specifies, in decimal, the maximum block length for this file, excluding line control characters. Block length must be equal to or greater than record length (RECL operand). See index entry *$DTFB considerations.*

RCAD—address

Specifies the symbolic address identifying the first byte of your logical buffer. The required size of the logical buffer depends upon the kind of operations requested for this file:

- If this is a receive file and OPC—N will be specified in GET requests for this file, the logical buffer must be large enough to contain one record for this file.
- If this is a receive file and OPC—BLK will be specified in GET requests for this file, the logical buffer must be large enough to contain the largest block of data expected, including line control characters.

  For a description of OPC—N and OPC—BLK, see index entry *$GETB macro instruction.*
- If this is a transmit file, the logical buffer must be large enough to contain one record for this file.

  Records are moved from the logical buffer to the BSC I/O buffers on PUT requests ($PUTB macro instruction), and moved from the BSC I/O buffers to the logical buffer on GET requests ($GETB macro instruction). See index entry *move mode.*

FTYP—RCV/TSM

Indicates whether the first operation for this file is receive (RCV) or transmit (TSM). If you define a receive file (RCV), the first I/O request for the file must be a GET request (see index entry *$GETB macro instruction*); if you define a transmit file (TSM), the first I/O request for the file must be a PUT request or a request for an online test (see index entries *$PUTB macro instruction* and *online test*).

BUFST—address

Specifies the symbolic address identifying the first byte of the area available to this file for I/O buffers and IOBs (input/output blocks).

Note: Each BSC file requires a unique I/O area.

BUFEND—address

Specifies the symbolic address identifying the last byte of the area available to this file for I/O buffers and IOBs. For formulas necessary to calculate the length of MLMP I/O areas, see index entry *MLMP I/O area.* See also index entry *online test* if you intend to use the online test.

BUFNO—decdig

Specifies the number of I/O buffers and IOBs to be contained in the I/O area for this file, and specifies that the I/O area is to be allocated by this $DTFB macro instruction. (See index entry *MLMP I/O area.*)

Note: Either BUFST and BUFEND or BUFNO should be specified. Otherwise, the $DTFB macro instruction allocates enough I/O area to contain only one IOB and buffer.

CODE—E/A

Specifies whether the character code of your data is EBCDIC (E) or ASCII (A). The character code you use is determined by the transmission code feature installed on your BSCA.

LINE—1/2

Specifies the BSCA this file uses; adapter 1 or adapter 2. The // BSCA operation control language (OCL) statement can override this operand. For OCL information, see the appropriate system control programming reference manual listed in the *Preface*.

UP—binary/0

Specifies the conditional opening of a DTF. If the bits specified, in binary, are on in the external indicator setting given by the last SWITCH OCL statement, the DTF is opened. The default 0 specifies the unconditional opening of a DTF. For OCL information, see the appropriate system control programming reference manual listed in the *Preface*.

CHN—name

Specifies the symbolic address of the next DTF in the chain. Chained DTFs are allocated, opened, or closed at the same time as the first DTF in the chain. An end-of-chain indicator, X'FFFF', is entered in the DTF if no chain operand is given.

CONV—Y/N

Specifies whether conversational replies can be sent from or to this file: Y if yes, N if no.

Note: Block length (BLKL operand) must equal record length (RECL operand) for a conversational file.

ITB—Y/N

Specifies whether intermediate block checking is requested; Y if yes, N if no. Intermediate block checking is not permitted for a conversational file.

TRANSP—Y/N

Specifies whether data for this file will be transmitted in transparent mode; Y if yes, N if no. Transparency may be specified only if the transparency feature is installed on the BSCA used. If N is specified and transparent data is received, no error occurs and the $BCRAN bit is set in the $BDATT field of the BSC DTF.

RVIADR—address

Specifies the symbolic address of a one-byte field you provide. The field is used with the mask specified in the RVIMSK operand (following paragraph) to indicate when a reverse interrupt request (RVI) is received or is to be sent. See index entry *reverse interrupt* for examples of using reverse interrupts. RVIADR—address requires the RVIMSK operand.

RVIMSK—hex

Specifies two hexadecimal characters to represent the reverse interrupt (RVI) mask. The bits represented by the mask are set on by MLMP in the RVIADR field (preceding paragraph) if a reverse interrupt request (RVI) is received from a remote terminal. If a reverse interrupt request is to be sent to a remote terminal, you must set on the mask in the RVIADR field. See index entry *reverse interrupt* for examples of using reverse interrupts. RVIMSK—hex requires the RVIADR operand.

DLYCT—decdig

Specifies a decimal delay count. The delay count is the number of seconds after receiving or transmitting a block of data that MLMP will wait for you to receive or transmit another block of data for the same file. MLMP waits the specified number of seconds by using the WACK ENQ and TTD NAK line control sequences.

Except when you have received or transmitted end of file, MLMP aborts transmission and posts the $BCLST completion code if the delay count is exhausted between transmissions. (See index entry *completion code*.)

If you do not specify a number, a 180-second delay count is assumed. If you do specify a delay count, consider the time that may be required for such things as device errors, halts, and readying I/O devices.

TYPE–

This operand specifies the type of line connection to be established for this file. You must have the appropriate network attachment feature installed before specifying one of the following line types:

PP

Specifies that this file will use a point-to-point non-switched line. PP is assumed if no line type is specified.

MP

Specifies that this file will use a multipoint line, and this station is a tributary station. TYPE–MP requires the TERMAD operand.

CS

Specifies that this file will use a multipoint line, and this station is the control station. TYPE–CS requires the LISTAD and ERRLOG operands.

AC

Specifies that this file will use a switched line, autocall. TYPE–AC requires the DIAL and DIALCT operands.

MC

Specifies that this file will use a switched line, manual call.

AA

Specifies that this file will use a switched line, auto answer.

MA

Specifies that this file will use a switched line, manual answer.

TERMAD–hex

Specifies the hexadecimal representation of the two-character polling or addressing sequence used by this file. If this is a transmit file (FTYP–TSM), TERMAD specifies polling characters; if this is a receive file (FTYP–RCV), TERMAD specifies addressing characters.

Each tributary station on a multipoint line must have unique polling and/or addressing characters. See index entry *tributary System/3 polling and addressing characters* for the polling and addressing characters available to identify System/3 tributary stations. See also index entry *polling/addressing.*

The TERMAD operand is used only when TYPE–MP is specified.

AUTORS–Y/N

Specifies whether MLMP will automatically send a negative response to polling and addressing sequences received after data transfer for this file is complete; Y if yes, N if no. If AUTORS–Y is specified, MLMP will continue to respond negatively to polling and addressing sequences until another MLMP I/O request ($GETB, $PUTB, or $RFT) is issued for the line, or until a request to close the MLMP files ($CLOS) is issued. (See index entry *macro instructions.*) If AUTORS–N is specified or if AUTORS is not specified at all for a tributary station, the tributary station will be on-line with the control station only when an MLMP I/O request ($GETB, $PUTB, or $RFT) has been accepted.

AUTORS–Y enables a tributary station to remain online with the control station after initial data transfer, even though data transfer is not occurring. When a tributary remains online until all data transfer between it and the control station is complete, the control station spends no time waiting for the tributary to respond to polling or addressing.

AUTORS is used only when TYPE–MP is specified, and the AUTORS operand must be specified the same way in all $DTFB macro instructions written for the same BSC line.

LISTAD—address

Specifies the symbolic address identifying the first byte of the polling or addressing list used by this file. If this is a transmit file (FTYP—TSM), LISTAD points to an addressing list; if this is a receive file (FTYP—RCV), LISTAD points to a polling list. See index entry *polling/addressing.*

LISTAD is required only when TYPE—CS is specified.

ERRLOG—address

Specifies the symbolic address identifying the first byte of an area in main storage to be used for logging transmission statistics by terminal. You must provide one such area for each telecommunications line used ($LOGB macro instruction); and each area must be large enough to contain statistics for each unique polling and addressing sequence used in your program. For the format of this logging area and a formula for computing its size, see index entries *$LOGB Macro Instruction* and *Terminal Statistics Logging Area.*

ERRLOG is required only when TYPE—CS is specified.

Note: Only one terminal statistics logging area per line is required in your program. Therefore, all ERRLOG operands specified for DTFs using the same line will be identical.

POLRES—Y/N

Specifies whether the control station modules required to poll and address tributaries, log terminal statistics, and close active files are to be resident in main storage; Y if yes, N if no.

Specifying POLRES—Y for control stations saves a significant amount of execution time because transient modules do not have to be found and loaded from the disk object library each time the control station polls or addresses a tributary, logs terminal statistics, or closes an active file.

POLRES is used only when TYPE—CS is specified, and the POLRES operand must be specified the same way in all $DTFB macro instructions written for the same BSC line.

LIMIT—decdig

Specifies the number of times, in decimal, MLMP will accept a negative response from each terminal in a wrapped polling or addressing list before posting the $BCNEG completion code (see index entry *completion code*). Valid entries are 1-254. If no number is specified, MLMP passes through a wrapped list until a positive response is received, an error is encountered, or the poll is canceled (see index entry *$CANB macro instruction*).

LIMIT is used only when TYPE—CS is specified.

Note: Consider defining LIMIT for any addressing list created by a $POLB macro instruction in which LAST—WRAP was specified. See index entry *$POLB macro instruction.*

DIAL—address

Specifies the symbolic address identifying the first byte of the field containing the decimal number that must be dialed to extablish a switched connection. This operand is used only when TYPE—AC is specified. DIAL—address requires the DIALCT operand (see following paragraph).

DIALCT—decdig

Specifies, in decimal, the length of the number that must be dialed to establish a switched autocall connection. The maximum length permitted is 12. This operand is used only when TYPE—AC is specified, and requires the DIAL operand.

RCVID—address

Specifies the symbolic address of the first byte of either the identification sequence required from the remote station or of the switched ID list (see index entry *switched ID list* for the format of the list). RCVID—address requires either the RCVCT operand or SWLIST — Y. Using RCVID and RCVCT or RCVID and SWLIST—Y improves data security on switched lines; these operands are recommended for all switched lines.

RCVCT—decdig

Specifies, in decimal, the length of the identification sequence required from the remote station. Length can be 1-15. If 1 is specified, MLMP expects to receive 2 characters — duplicates of the character addressed by the RCVID operand (preceding paragraph). If no length is specified, 0 is assumed. RCVCT—decdig requires the RCVID operand.

SWLIST—Y/N

Specifies whether this switched answer line will use the switched ID list (see index entry *switched ID list* for the format of the list); Y if yes, N if no. SWLIST—Y can be specified only when TYPE—AA or TYPE—MA is specified. SWLIST—Y requires the RCVID operand.

SNDID—address

Specifies the symbolic address of the first byte of the identification sequence required by the remote station. SNDID—address requires the SNDCT operand. Using the SNDID and SNDCT operands improves data security on switched lines; these operands are recommended for all switched lines.

SNDCT—decdig

Specifies, in decimal, the length of the identification sequence required by the remote station. Length can be 1-15. If 1 is specified, MLMP transmits 2 characters — duplicates of the character addressed by the SNDID operand (preceding paragraph). SNDCT—decdig requires the SNDID operand.

SPAN—Y/N

Specifies whether records in this file will span blocks of text; Y if yes, N if no. A spanned record must be contained within two contiguous blocks of text.

When spanned records are received or transmitted, DTF fields $BDWKB and $BDREL are altered. The fields are restored after successful I/O completion. If an error occurs during transmission of spanned records, you must restore $BDWKB and $BDREL before requesting another operation for the file.

SPAN—Y requires the RECSEP operand if this is a receive file (FTYP—RCV). See index entry *$DTFB considerations*. See also index entry *BSC DTF* for the format of the BSC DTF.

RECSEP—hex

Specifies the hexadecimal representation of a one-character record separator used to separate variable length records in blocks of text. See index entry *$DTFB considerations*.

ERRCT—decdig

Specifies the number of times, in decimal, that MLMP retries an unsuccessful operation before posting an error condition. Valid entries are 1-254. If no number is specified, an error retry count of 7 is assumed.

Note: ERRCT specifies an error retry count only for the local MLMP program; ERRCT does not affect the remote terminal. In an error situation occurring between two terminals with different retry counts, the lower retry count determines when the error becomes permanent.

$DTFB Considerations

MLMP supports three kinds of record formats: fixed length, variable length, and spanned. The kind of format you choose determines the way in which four $DTFB operands must be specified: RECL—decdig, BLKL—decdig, SPAN—Y/N, and RESCEP—hex.

Fixed Length: If you choose to use fixed length records:

1. Specify the record length in the RECL operand.
2. Specify a multiple of the record length in the BLKL operand.
3. Do not code the SPAN and RECSEP operands.

Variable Length: Variable length records can change in length from one transmission to another. If you choose to use variable length records:

1. RECL—The value specified in the RECL operand should be the maximum expected record length. However, when you transmit variable length records you must move the length of each record to $BDREL in the DTF before you issue $PUTB to transmit the record. (See index entry *BSC DTF* for the format of the DTF.) After each successful GET request issued to receive one or more variable length records (completion code = $BCDNE):
   - The length of the block received, including control characters, is moved to $BDREL if you specified OPC-BLK in $GETB (see index entry *$GETB macro instruction*),
     or
   - The length of the record received, including the record separator, is moved to $BDREL if you specified the RECSEP operand in $DTFB (see item 4 following).
2. BLKL—The block length you specify should be the maximum expected block length.
3. SPAN—Variable length records may or may not be spanned (see following paragraph).
4. RECSEP—Although record separators are not required with variable length records, you may want to use a record separator to delimit the records. If you do specify a record separator, MLMP automatically inserts the record separator at the end of each record transmitted. The record separator is the last data character moved to your logical buffer (addressed by the RCAD operand) when you receive variable length records.

OPC—BLK is recommended for each GET request (see index entry *$GETB macro instruction*) if you are to receive variable length records that do not contain record separators. When OPC—BLK is specified in a GET request, the length of the block received is moved to $BDREL in the DTF. See index entry *BSC DTF* for the format of the DTF.

Spanned: Spanned records can be fixed or variable length. If you choose to use spanned records:

1. Specify RECL and BLKL according to the format, fixed or variable length, you choose (see two preceding paragraphs).
2. Specify SPAN—Y.
3. Specify a record separator character (RECSEP operand). See index entry *MLMP message formats* for an illustration of spanned records.

   Note: The 3735 transmits spanned records. For formatting considerations unique to the 3735, see *IBM 3735 Programmer's Guide,* GC30-3001.

Allocate BSC Files ($ALOC)

Every BSC file must be allocated; that is, the BSCA required by each file must be reserved for the file before the file can be processed. All files in your program must be allocated before you begin any telecommunications I/O operations. The $ALOC macro instruction generates code that effects a branch to a system allocate routine. The system allocate routine reserves the devices identified by the $ALOC macro instruction.

Note: $ALOC must not be issued while BSC I/O operations are being executed.

$ALOC Macro Instruction Format

[name]	$ALOC	[DTF—name]

If you specify the keyword DTF, enter, as the parameter, the name of the DTF (file) to be allocated. If the operand is not given, the address of the DTF is assumed to be in register 2.

If the DTF specified is in a chain, all DTFs following in the chain are opened by this request. (MLTA DTFs must not be in the chain. For information regarding MLTA, see *IBM System/3 Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual,* GC21-7560.)

After $ALOC is executed, register 2 contains the address of the first DTF allocated.

Open BSC Files ($OPEN)

Every BSC file must be opened. The DTF and I/O area used by the file must be prepared to accommodate the data composing the file, and a DTF must be generated. The $OPEN macro instruction generates code that effects a branch to a system open routine; the routine opens the files identified by the $OPEN macro instruction.

Note: If you reopen a BSC file that has been closed (see index entry *$CLOS macro instruction),* record length, block length, and all other file attributes will be the same as they were at the time the file was closed.

$OPEN Macro Instruction Format

[name]	$OPEN	[DTF–name]

If you specify the keyword DTF, enter, as the parameter, the name of the DTF (file) to be opened. If the operand is not given, the address of the DTF is assumed to be in register 2.

If the DTF specified is in a chain, all DTFs following in the chain are opened by this request. (MLTA DTFs must not be in the chain. For information regarding MLTA, see *IBM System/3 Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual,* GC21-7560.)

After $OPEN is executed, register 2 contains the address of the last DTF opened.

Generate a Model 10 and Model 12 Checklist ($CKL)

The Model 10 Disk System, Model 12, and Model 15 use a macro named $CKL to generate a checklist. Since there are differences in parameters and in function, checklist macros are discussed separately.

One $CKL macro instruction creates one entry in a checklist. A checklist identifies DTFs to be checked for I/O completion by the $CHK macro instruction. The kinds of DTFs that can be identified by the checklist are: BSC DTFs, MLTA DTFs, and console DTFs. The $CKL macro instructions required to create a particular checklist must be coded consecutively. See index entry *checklist* for the format of the checklist. See also index entry *check for I/O completion ($CHK).*

Model 10 and Model 12 $CKL Macro Instruction Format

[name]	$CKL	DTF–address [,SKIP–Y/N] [,REQK–Y/N or ,CONS–Y/N] [,RTN–Y/N] [,LAST–Y/N]

name

A name given to a $CKL macro instruction becomes the symbolic address of the first byte of the generated checklist entry. The name can then be used to address the entry if you want to change the entry.

DTF–address

Specifies the symbolic address of the first byte of the DTF for which this entry is being created.

SKIP–Y/N

Specifies whether this entry should be skipped when the checklist is scanned; Y if yes, N if no.

REQK–Y/N or CONS–Y/N

Specifies whether the check routine (see index entry *$CHK macro instruction*) should check for console requests (REQ key pressed); Y if yes, N if no. Whenever you want the check routine to check for console requests, you must include a console DTF in the checklist and specify REQK–Y or CONS–Y for that entry. REQK–Y and CONS–Y are ignored if they are specified for a DTF that is not a console DTF. (A device code of X'10' in the first byte of a DTF, field $BDDEV, denotes a console DTF.) If the operator has pressed the REQ key, a completion code of X'50' is posted in the console DTF.

If a console DTF is to be used for both request key and data input at the same time, two entries (REQK–Y and REQK–N) must be specified in the checklist for that DTF.

For more information regarding console operations, see *IBM System/3 Models 10 and 12 Control Programming Macros Reference Manual,* GC21-7562.

RTN–Y/N

Specifies whether you want control returned from the check routine even if no I/O operation is complete; Y if yes, N if no. The RTN operand is effective only when specified for the first entry in the checklist, and applies to all entries in the list. See completion code $BCCMP (index entry *completion code).*

LAST–Y/N

Specifies whether this is the last entry in the checklist; Y if yes, N if no. LAST–Y must be specified for the last entry in the checklist.

Generate a Model 15 Checklist ($CKL)

One $CKL macro instruction creates one entry in a checklist. A checklist identifies DTFs to be checked for I/O completion or for depression of Program Function Key 9 (PF9). The kinds of DTFs that can be identified by the checklist are: BSC DTFs, and dummy DTFs which are 15-byte DTFs used for the PF9 key. See index entry *checklist* for the format of the checklist. See also index entry *check for I/O completion ($CHK).*

Model 15 $CKL Macro Instruction Format

[name]	$CKL	DTF–address [,SKIP–Y/N] [REQK–Y/N] [,RTN–Y/N] [,LAST–Y/N]

name

A name given to a $CKL macro instruction becomes the symbolic address of the first byte of the generated checklist entry. The name can then be used to address the entry if you want to change the entry.

DTF–address

Specifies the symbolic address of the first byte of the DTF for which this entry is being created. A 15-byte dummy DTF is required to check whether Program Function Key 9 has been pressed. Displacement X'00' should contain X'10' and a completion code will be returned in displacement X'OI

SKIP–Y/N

Specifies whether this entry should be skipped when the checklist is scanned; Y if yes, N if no.

REQK–Y/N

Specifies whether the check routine (see index entry *$CHK macro instruction)* should check to see if the Program Function Key 9 (PF9) has been pressed; Y if yes, N if no. Whenever you want the check routine to check for PF9 requests, you must include a dummy DTF in the checklist and specify REQK–Y for that entry. REQK–Y is ignored if it is specified for a DTF that is not a dummy (PF9) DTF. (A device code of X'10' in the first byte of a DTF denotes a dummy [PF9] DTF.) If the operator has pressed the PF9 key, a completion code of X'50' is posted at displacement X'0E' of the dummy (PF9) DTF.

For more information regarding PF9 operations, see *IBM System/3 Model 15 System Control Programming Macros Reference Manual,* GC21-7608.

RTN–Y/N

Specifies whether you want control returned from the check routine even if no I/O operation is complete; Y if yes, N if no. The RTN operand is effective only when specified for the first entry in the checklist, and applies to all entries in the list. See completion code $BCCMP (index entry *completion code*).

LAST–Y/N

Specifies whether this is the last entry in the checklist, Y if yes, N if no. LAST–Y must be specified for the last entry in the checklist.

Generate a Polling/Addressing List ($POLB)

If your station is a control station (TYPE–CS in the $DTFB macro instruction), you must generate a polling or addressing list in order to poll or address tributary stations. GET requests require a polling list; PUT requests require an addressing list. See index entries *$GETB macro instruction,* and *$PUTB macro instruction.*

One $POLB macro instruction creates one entry in a polling or addressing list; each entry contains one sequence of polling or addressing characters. An addressing list contains only one entry. Polling lists may contain a number of entries; the $POLB macro instructions that create a polling list must be coded consecutively.

For the format of polling/addressing lists, see index entry *polling/addressing list.* See also index entry *tributary System/3 polling and addressing characters.*

$POLB Macro Instruction Format

[name]	$POLB	ID–hex,TERMAD–hex,LEN–decdig [,LAST–N/OPEN/WRAP]

name

A name given to a $POLB macro instruction becomes the symbolic address of the first byte of the polling/addressing entry generated. The name can then be used to address the entry if you want to change the entry. See also index entry *change a polling list ($BCPL).*

ID–hex

Specifies two hexadecimal characters to identify this entry in the polling/addressing list. Valid entries are X'00' through X'EF'.

When an entry in a polling/addressing list is used to poll or address a tributary station, MLMP places the entry ID in the BSC DTF at $BDIND (see index entry *BSC DTF* for the format of the DTF). If you specify a unique ID for each entry in a polling list, you can determine which station was polled last by checking the contents of $BDIND after a completion code has been posted (see index entry *$CHK macro instruction*).

You can specify which station will be polled first in a polling sequence by moving the ID from the related polling list entry to the DTF before you issue the first GET request for that file (see index entry *$GETB macro instruction*). The ID you move must be different from the previous ID in $BDIND.

If you move X'F0' to $BDIND all entries in the list are activated, and polling begins with the first entry in the list. If you move X'F1' to $BDIND polling begins with the first entry in the list, but only entries that are currently active are included in the poll.

When an MLMP program is assembled, $BDIND is set to X'F1'. MLMP always considers addressing lists to be active.

For more information on how to change a polling list, see index entry *change a polling list ($BCPL).*

TERMAD–hex

Specifies the hexadecimal representation of up to 7 polling or addressing characters. The polling or addressing characters must not include any of the line control characters shown in Appendix D. Appendix D also shows the polling and addressing characters available for System/3 tributary stations.

LEN—decdig

Specifies, in decimal, the number of bytes represented in the TERMAD operand. Decdig must not be greater than 7.

LAST—N

Specifies that this entry is not the last entry in the polling list.

—OPEN

Specifies that this is the last entry in the polling/addressing list, and that polling or addressing must end even if a positive response is not received ($BCNEG completion code).

Note: After polling the last station in an open polling list, move the ID of the first entry in the polling list, (X'F0', or X'F1') to $BDIND if you expect to use the polling list again in your program. See the preceding description of the ID operand.

—WRAP

Specifies that this is the last entry in a polling list, or is an addressing entry. If a positive response is not received, polling must continue from the beginning of the list, or addressing must continue, until a positive response is received, an error is encountered, or LIMIT is reached (LIMIT is defined in the $DTFB macro instruction).

Either LAST—OPEN or LAST—WRAP is required in the last entry of a polling list and in all addressing entries. If LAST—WRAP is specified for a polling list, the polling cycle can always be canceled by the $CANB macro instruction (see index entry *$CANB macro instruction*). However, to prevent an infinite addressing loop in the event that a negative response (NAK) is received from the terminal addressed, LIMIT must be defined in $DTFB (see index entry *$DTFB macro instruction*) whenever LAST—WRAP is specified for an addressing list.

Change a Polling List ($BCPL)

The $BCPL macro instruction enables you to activate and deactivate selected entries in a polling list created by the $POLB macro instruction. For example, you can use the $BCPL macro instruction to:

- Deactivate entries you have decided not to use. You may want to deactivate an entry if the terminal does not respond to polling ($BCNON completion code). See index entry *completion code.*
- Reactivate entries that have been deactivated.

Note: $BCPL requires approximately 110 bytes of main storage plus the space required by the parameter list. Unless you have a large number of polling entries to change, you can activate and deactivate polling entries more efficiently yourself by changing the status byte in polling entries. See index entry *polling/addressing list.*

$BCPL Macro Instruction Format

[name]	$BCPL	[PARM—address]

PARM—address

Specifies the symbolic address of a parameter list you provide. The parameter list defines the changes you want to make to a polling list. For the format of the parameter list, see index entry *parameter list for changing a polling list or switched ID list.* You can use the $CHGB macro instruction to generate the parameter list. See index entry *$CHGB macro instruction.*

If the PARM operand is not specified, the address of the parameter list is assumed to be in register 1.

After $BCPL is executed, register 1 contains the address of the parameter list.

Generate a Parameter List for Changing a Polling List or a Switched ID List ($CHGB)

The $CHGB macro instruction generates a parameter list for changing a polling list (see index entry *$BCPL macro instruction*) and for changing a switched ID list (see index entry *$BCSW macro instruction*). For the format of this parameter list, see index entry *parameter list for changing a polling list or switched ID list.*

$CHGB Macro Instruction Format

[name]	$CHGB	TYPE–AM/AN/DM/DN,DTF–address, NUM–hex,CHARS–hex

name

A name given to a $CHGB macro instruction becomes the symbolic address of the first byte of the generated parameter list. The name can then be used to address specific fields in the list if you want to change the fields.

TYPE–

Specifies the changes to be made in the polling/addressing or switched ID list.

AM

Activates selected entries, selecting only those entries whose characters exactly match the characters specified in the CHARS operand.

AN

Activates selected entries, selecting only those entries whose first N characters match the first N characters specified in the CHARS operand, where N is the number specified in the NUM operand.

DM

Deactivates selected entries, selecting only those entries whose characters exactly match the characters specified in the CHARS operand.

DN

Deactivates selected entries, selecting only those entires whose first N characters match the first N characters specified in the CHARS operand, where N is the number specified in the NUM operand.

DTF–address

Specifies the symbolic address of the DTF whose polling list or switched ID list is to be changed.

NUM–hex

Specifies in hexadecimal how many of the characters specified in the CHARS operand are to be compared to characters in polling list or switched ID list entries.

CHARS–hex

Specifies the hexadecimal representation of characters to be compared to characters in polling list or switched ID list entries.

Allocate the Terminal Statistics Logging Area ($LOGB)

The $LOGB macro instruction allocates space for the Terminal Statistics Logging Area (see index entry *Terminal Statistics Logging Area*) for control stations (TYPE–CS in $DTFB).

$LOGB Macro Instruction Format

[name]	$LOGB	NUM–decdig,LEN–decdig

name

A name given to a $LOGB macro instruction becomes the symbolic address of the first byte of the allocated Terminal Statistics Logging Area and can be specified in the ERRLOG operand of the appropriate $DTFB macro instruction.

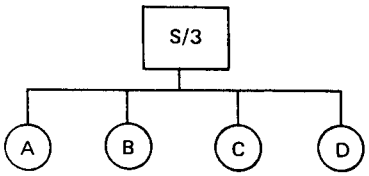
NUM—decdig

Specifies in decimal the number of entries the Terminal Statistics Logging Area must contain. The value for NUM is calculated in one of two ways:

1. Polling and/or addressing, no clusters:

   NUM = number of terminals to be polled + [number of terminals to be addressed or number of terminals for which an online test may be requested (see index entry *online test*)]

   For example:



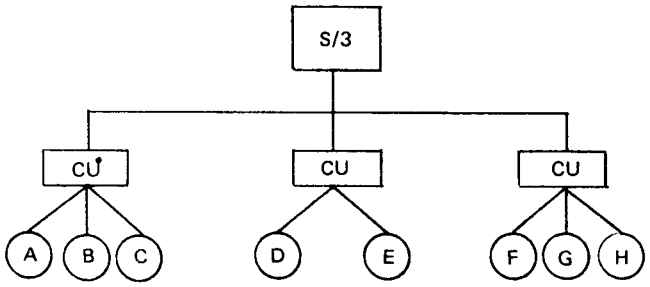
   Polled terminals = 4
   Addressed terminals = 4

   NUM = 8

2. Polling and/or addressing clusters:

   NUM = number of control units to be polled + number of devices to be polled + number of devices to be addressed

   For example:



   Control units polled = 3
   Devices polled = 8
   Devices addressed = 8

   NUM = 19

LEN—decdig

Specifies in decimal the number of polling/addressing characters used in a polling/addressing sequence for this program. The maximum number for each terminal is 7.

Generate a Switched ID List ($SWIB)

If your station is an autoanswer or manual answer station (TYPE—AA or TYPE—MA in $DTFB), you can use the $SWIB macro instruction to generate a switched ID list containing the station identification sequences that your station will accept. For the format of this list, see index entry *switched ID list format.*

Each $SWIB macro instruction creates one entry in the switched ID list; each entry contains one sequence of station ID characters. $SWIB macro instructions must be coded consecutively.

$SWIB Macro Instruction Format

[name]	$SWIB	SELECT—hex,STATID—hex, LEN—decdig [,LAST—Y/N]

name

A name given to a $SWIB macro instruction becomes the symbolic address of the first byte of the switched ID entry generated. The name can then be used to address the entry if you want to change the entry (for example, activate or deactivate an entry) by using the $$BCSW macro instruction.

SELECT—hex

Specifies two hexadecimal characters for selecting this entry in the switched ID list. Valid entry selection characters are X'00' through X'EF'.

When an entry in the switched ID list is compared to an identification sequence received from a remote terminal, MLMP places the entry selection characters in the BSC DTF at $BDRLN (see index entry *BSC DTF* for the format of the DTF). If you specify unique entry selection characters for each entry in a switched ID list, you can determine which identification sequence you accepted last by checking the contents of $BDRLN after a completion code has been posted on your last MLMP I/O request.

If you want to communicate with only one of several possible terminals, move the entry selection characters associated with the terminal's expected identification sequence to the DTF before you issue the first GET request to that file. The entry selection characters you move must be different from the previous entry selection characters in $BDRLN.

If you move X'F0' to $BDRLN, all entries in the switched ID list are activated before the identification sequence received is inspected. Entries in the switched ID list are then compared to the ID received, beginning with the first entry in the list. If you move X'F1' to $BDRLN, only those entries currently active in the switched ID list are used to inspect a received station ID.

When an MLMP program specifying SWLIST–Y in $DTFB is assembled, $BDRLN is set to X'F1' to indicate that any active identification sequence in the switched ID list is acceptable.

For more information on how to change a switched ID list, see index entry *$CHGB macro instruction.*

STATID–hex

Specifies the hexadecimal representation of a station identification sequence. The maximum number of characters is 15.

LEN–decdig

Specifies, in decimal, the number of bytes represented in the STATID operand. The maximum number of bytes is 15.

If you are using a switched ID list but you are willing to establish a connection with a terminal that does not send an identification sequence, you must define in your list an entry with LEN–0 specified. (If an entry with length = zero is encountered in the list, a connection is established with the calling station whether a station ID was received or not. Consequently, if an entry with LEN–0 must be used, place the entry at the end of your list so you can check station IDs that are received.)

LAST–Y/N

Specifies whether this is the last entry in the switched ID list; Y if yes, N if no. LAST–Y causes a one-byte end-of-list indicator to be generated. If end-of-list is encountered before an acceptable station identification sequence has been received, an invalid ID completion code ($BCBID) is posted (see index entry *completion code*).

Change a Switched ID List ($BCSW)

The $BCSW macro instruction enables you to activate and deactivate selected entries in a switched ID list created by the $SWIB macro instruction. For example, you can use $BCSW to:

- Deactivate entries you have decided not to use. You may want to deactivate an entry to prevent communication with a particular terminal, or to prevent communication with a terminal that does not send an identification sequence (deactivate the entry with LEN–0 specified).
- Reactivate entries that have been deactivated.

Note: $BCSW requires approximately 110 bytes of main storage plus the space required by the parameter list. Unless you have a large number of switched ID entries to change, you can activate and deactivate switched ID entries more efficiently yourself by changing the status byte in switched ID entries. See index entry *switched ID list.*

$BCSW Macro Instruction Format

[name]	$BCSW	[PARM–address]

PARM–address

Specifies the symbolic address of a parameter list you provide. The parameter list defines the changes you want to make to a switched ID list. For the format of the parameter list, see index entry *parameter list for changing a polling list or a switched ID list.* You can use the $CHGB macro instruction to generate the parameter list. See index entry *$CHGB macro instruction.*

If the PARM operand is not specified, the address of the parameter list is assumed to be in register 1.

After $BCSW is executed, register 1 contains the address of the parameter list.

Generate a Translate Parameter List ($TRL)

The $TRL macro instruction generates the parameter list required by the System/3 translate routine. Appendix C shows the format of the parameter list.

$TRL Macro Instruction Format

[name]	$TRL	TO–address,FROM–address, LEN–decdig,TRT–address

A name given to a $TRL macro instruction becomes the symbolic address of the first byte in the generated parameter list. The name can then be used to address specific fields in the list if you want to alter the fields.

TO – address

Specifies the symbolic address of the first byte of the area to which translated data will be moved.

FROM – address

Specifies the symbolic address of the first byte of the data field to be translated. This address can be the same as the address specified in the TO operand.

LEN – decdig

Specifies the decimal length of the FROM field.

TRT – address

Specifies the symbolic address of the first byte of the translate table.

Generate a Translate Table ($TRTB)

The $TRTB macro instruction generates an EBCDIC to ASCII or an ASCII to EBCDIC translate table. The table is generated in the format required by the $TRL macro instruction (see index entry *$TRL macro instruction*), and can be addressed by $TRL when you translate data.

$TRTB Macro Instruction Format

[name]	$TRTB	[CODE–E/A] [,HEX–hex]

name

A name given to a $TRTB macro instruction becomes the symbolic address of the first byte of the generated translate table.

CODE–E/A

Specifies whether the character code of the data to be translated is EBCDIC (E) or ASCII (A). $TRTB generates a 258-byte table if CODE–E is specified and a 130-byte table if CODE–A is specified.

Note: If you specify CODE–A, you might want to code "DC 128XL1'FF'" after the $TRTB macro instruction to allow for the occurrence of invalid ASCII characters. See index entry *System/3 translate tables.*

HEX–hex

Specifies the hexadecimal pattern with which to replace any invalid characters found during translation. If the HEX operand is not specified, the replacement character is the substitute character – EBCDIC-3F, ASCII-1A.

System/3 Translate Tables

Translate Tables Generated by $TRTB: Translate tables generated by the $TRTB macro instruction are generated in the following format:

Byte	Field Description
0	Contains X'FF'. This is the hexadecimal value used in the translate table to identify characters that cannot be translated from EBCDIC to ASCII if CODE–E was specified in $TRTB or from ASCII to EBCDIC if CODE–A was specified.
1	Byte contents to be used in place of characters that are not translated.
2-n	Actual translate table.

The translate table, bytes 2-n, is constructed so that the hexadecimal value of a character to be translated can be used as a displacement (from byte 2) to locate the correct translation in the table.

When the correct translation for a byte is located in the table, the translation is compared to byte 0. If the two bytes are the same:

- The completion code in the translate parameter list (see index entry *translate parameter list*) is set to indicate that an invalid character was detected, and
- The contents of byte 1 are substituted for the original character.

  If the translation of a character is not the same as the contents of byte 0, the hexadecimal value in the translate table is substituted for the original character.

User-Defined Translate Tables: If you don't want to translate certain valid EBCDIC or ASCII characters (you might not want to translate BSC line control characters, for example), you can generate your own translate table. However, you must generate the table in the format described in the preceding paragraphs.

Choose hexadecimal values for bytes 0 and 1 of the table. Then, as you construct the rest of the table, substitute the value of byte 0 for each character that cannot be translated and for each valid character that you choose not to translate.

Generate an Interface to the Translate Routine ($TRAN)

The $TRAN macro instruction generates linkage to the the System/3 translate routine. After issuing $TRAN, check the completion code in the translate parameter list to determine whether or not invalid characters were found. See index entry *translate parameter list* for the format of the translate parameter list.

$TRAN Macro Instruction Format

[name]	$TRAN	[TRL–address]

TRL–address

Specifies the symbolic address of the translate parameter list. If not given, the address is assumed to be in register 1. After $TRAN is executed, register 1 contains the address of the translate parameter list.

Generate an Online Test Parameter List ($RFTL)

The $RFTL macro instruction generates the parameter list required for online test requests (see index entry *online test requests*). For the format of the parameter list, see index entry *online test parameter list.*

$RFTL Macro Instruction Format

[name]	$RFTL	TYPE–00/01/06/14,NUM–decdig, LEN–decdig [,CODE–E/A] [,TERMAD–hex]

name

A name given to a $RFTL macro instruction becomes the symbolic address of the first byte of the generated parameter list. The name can then be used to address specific fields in the list if you want to change the fields.

TYPE–

Specifies, in decimal, the online test type:

00

Receive and acknowledge the test message the number of times specified in the NUM operand. The formatted test request must not be more than 300 characters long. See index entry *online test requests.*

01

Transmit the test message the number of times specified in the NUM operand. The formatted test request must not be more than 300 characters long. See index entry *online test requests.*

06

Transmit 36 alphameric characters, A-Z and 0-9, the number of times specified in the NUM operand. Transmit the characters in ASCII (ASCII adapter only).

14

Transmit 36 alphameric characters, A-Z and 0-9, the number of times specified in the NUM operand. Transmit the characters in EBCDIC (EBCDIC adapter only).

NUM–decdig

Specifies, in decimal, the number of times to transmit or receive the test message. Valid entries are 1-99 (leading zeroes must not be used).

LEN–decdig

Specifies, in decimal, the number of addressing characters (0-7). LEN must be 0 for:

- All non-multipoint lines (TYPE–PP,AC,MC,AA, or MA in $DTFB)
- Multipoint control stations (TYPE–CS in $DTFB)
- Multipoint tributary stations (TYPE–MP in $DTFB) requesting online test type 00.

LEN must be greater than 0 only for multipoint tributary stations (TYPE–MP in $DTFB) requesting some test type other than 00.

CODE–E/A

Specifies whether the character code of your data is EBCDIC (E) or ASCII (A). The character code you use is determined by the transmission code feature of your BSCA.

TERMAD–hex

Specifies the hexadecimal representation of the addressing characters to be used, not more than 7 bytes. The number of bytes required to contain the addressing characters must be equal to the number specified in the LEN operand.

INITIATING DATA TRANSFER

To initiate data transfer you must issue:

- GET requests to receive data ($GETB macro instruction), or
- PUT requests to transmit data ($PUTB macro instruction).

All GET and PUT requests are executed in move mode. The first request causes MLMP to establish line connections according to the operands specified in the $DTFB macro instruction. An initial GET request can be canceled after it has been issued ($CANB macro instruction).

You must check for the completion of every BSC I/O operation you initiate ($CHK macro instruction). That is, you must issue $CHK after every GET request, PUT request, and request for online test (see index entry *online test*) before you issue another GET request for the same line, PUT request, or request for online test. You must also issue $CHK after each $CANB macro instruction.

Move Mode

Data is moved from the MLMP I/O buffers to your logical buffer on GET requests, and from your logical buffer to the MLMP I/O buffers on PUT requests. A single GET or PUT request does not necessarily result in the actual transmission of data over a telecommunications line.

Records for conversational files (CONV–Y in the $DTFB macro instruction) are transmitted one at a time. Consequently, each GET or PUT request causes MLMP to receive or transmit one record.

For all nonconversational files, however, a GET request causes data to be transmitted from the remote terminal only if the GET request moves to your logical buffer the last record contained in an MLMP I/O buffer. After the GET request is executed, an I/O buffer is empty and available to MLMP for receiving more data, and transmission from the remote terminal can continue.

A PUT request for a nonconversational file causes data to be transmitted to the remote terminal only if the record to be moved to an MLMP I/O buffer cannot be contained in the current I/O buffer.

Issue a GET Request ($GETB)

The $GETB macro instruction instructs MLMP to move data from an MLMP I/O buffer to your logical buffer. Do not attempt to move data to or from your logical buffer after issuing a GET request until you have been posted a DTF completion code by the check routine. See index entry *$CHK macro instruction.*

$GETB Macro Instruction Format

[name]	$GETB	[DTF–address] [,REJECT–address] [,OPC–N/BLK]

DTF–address

Specifies the symbolic address of the DTF (file) for which the GET request is issued. If not given, the address of the DTF is assumed to be in register 2.

After $GETB is executed, register 2 contains the address of the DTF for which the GET request was issued.

REJECT–address

Specifies the symbolic address of a user routine to receive control if the GET request cannot be accepted by MLMP. You must provide the routine. The routine should check the DTF completion code to determine why the GET request was not accepted. See index entry *completion code.*

If the REJECT operand is not specified, check for a DTF completion code of $BCREQ after each GET request to determine whether or not the request was accepted. See index entry *completion code.*

You might want to print a message to signal rejected GET request. In most cases, a request is rejected because of a logic error in your program. Check your logic flow, parameter lists, and DTF for errors. Consider using the $SNAP macro instruction to dump your program. See index entry *$SNAP macro instruction.*

OPC–N

Specifies normal blocking and deblocking; that is, data received will be stripped of control characters and moved to your logical buffer (addressed by the RCAD operand in $DTFB) one record at a time.

OPC–BLK

Specifies that data received will be moved to logical buffer one block at a time. Each block of data moved will include line control characters. The length of the block moved will be placed by MLMP in $BDREL of the DTF. For the format of the DTF, see index entry *BSC DTF.*

OPC–BLK is recommended for GET requests for conversational files. See index entry *conversational reply* for the significance of OPC–BLK in GET requests for a conversational file. OPC–BLK is also recommended for GET requests issued to receive variable length records that do not contain record separators. See index entry *$DTFB considerations* for more information on variable length records.

Note: If you specify OPC–BLK, be sure your logical buffer (identified by the RCAD operand in the $DTFB macro instruction) is large enough to contain an entire block of data plus line control characters. For a description of the RCAD operand, see index entry *$DTFB macro instruction.*

Issue a PUT Request ($PUTB)

The $PUTB macro instruction instructs MLMP to move data from your logical buffer to an MLMP I/O buffer. Do not attempt to move data to or from your logical buffer after issuing a PUT request until you have been posted a DTF completion code by the check routine. See index entry *$CHK macro instruction.*

$PUTB Macro Instruction Format

[name]	$PUTB	[DTF–address] [,REJECT–address] [,OPC–N/EOB/EOF/EOW]

DTF–address

Specifies the symbolic address of the DTF (file) for which the PUT request is issued. If not given, the address of the DTF is assumed to be in register 2.

After $PUTB is executed, register 2 contains the address of the DTF for which the PUT request was issued.

REJECT–address

Specifies the symbolic address of a user routine to receive control if the PUT request cannot be accepted by MLMP. You must provide the routine. The routine should check the DTF completion code to determine why the PUT request was not accepted. See index entry *completion code.*

If the REJECT operand is not specified, check for a DTF completion code of $BCREQ after each PUT request to determine whether or not the request was accepted. See index entry *completion code.*

You might want to print a message to signal a rejected PUT request. In most cases, a request is rejected because of a logic error in your program. Check your logic flow, parameter lists, and DTF for errors. Consider using the $SNAP macro instruction to dump your program. See index entry *$SNAP macro instruction.*

OPC–N

Specifies normal blocking and deblocking. That is, data will be moved from your logical buffer (addressed by the RCAD operand in $DTFB) to available I/O buffers one record at a time.

OPC–EOB

Specifies that in addition to performing normal blocking operations, MLMP must make the current record the last record in the current output buffer, thereby forcing an end-of-block condition. ETB (ETX for conversational files) is transmitted at the end of this record.

See index entry *conversational reply* for the significance of OPC–EOB in PUT requests for conversational files.

OPC–EOF

Specifies that MLMP transmit EOT to indicate end-of-file.

Note: If you specify OPC–EOF in the first PUT request for a file, or in a PUT request that immediately follows a GET request for the same file, the PUT request will be rejected and the $BCIGN completion code will be posted in the DTF. See index entry *completion code.*

OPC–EOW

Specifies that EOT will be sent after each record in response to ACK or WACK. Each transmission consists of only one record when OPC–EOW is specified.

If OPC–EOW is specified in PUT requests for a System/3 control station's files (TYPE–CS in the $DTFB macro instruction), tributary stations can process each record as it is received from the control station without delaying transmissions between the control station and other tributaries in the network. After EOT has been sent to a tributary in response to ACK or WACK, other tributaries can be polled or addressed. The tributary that received EOT can also be polled or addressed again.

For more efficient use of line time, OPC–EOW is recommended when you transmit from a control station to a 2972/2980 terminal. You must specify OPC–EOW if you are transmitting to a 3270 printer. However, EOT is not a valid response to WACK for every kind of terminal. Know the restrictions pertaining to the terminals you use before specifying OPC–EOW.

Note: Don't specify OPC–EOW for a conversational file (CONV–Y in the $DTFB macro instruction) if you expect to receive a conversational reply. If you do specify OPC–EOW for a conversational file, conversational replies will be ignored and lost.

Cancel a GET Request ($CANB)

The $CANB macro instruction instructs MLMP to cancel a GET operation that is in progress. $CANB enables you to override wrapped polling lists if you are a control station (TYPE–CS in the $DTFB macro instruction) and cancel initial GET requests.

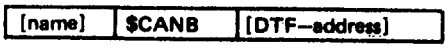
Issue $CHK for each accepted cancel GET request. If the check routine posts the $BCEOT completion code, the GET request was canceled by $CANB. If any other completion code is posted, it pertains to the GET request–$CANB has been ignored and you must proceed according to the completion code posted for the GET request.

$CHK needn't be issued for a GET request before it is canceled. However, if you are polling with a wrapped polling list or waiting to be addressed, a way to determine whether or not to cancel the GET request is:

1. Check the GET request with RTN–Y specified in $CKL (see index entry *$CKL macro instruction*).

2. Look for the $BCCMP completion code (see index entry *completion code*). If $BCCMP is posted by the check routine and none of the DTFs in the checklist indicate completion yet, you might want to stop polling or waiting to be addressed.

$CANB Macro Instruction Format



DTF–address

Specifies the symbolic address of the DTF (file) for which the cancel request is issued. If not given, the address is assumed to be in register 2.

After $CANB is executed, register 2 contains the address of the DTF for which the cancel GET request was issued.

Check for I/O Completion ($CHK)

The $CHK macro instruction generates linkage to a check routine. The check routine checks for I/O completion by examining the DTFs identified in the list created by the $CKL macro instruction. See index entries *checklist* and *$CKL macro instruction* for a description of the checklist.

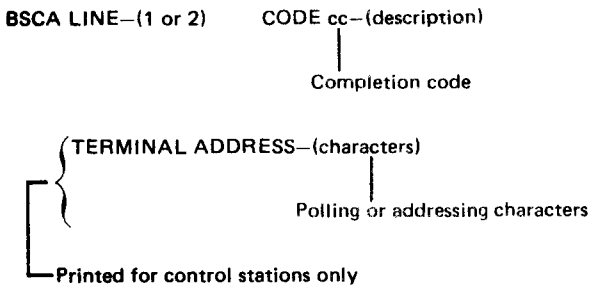
If an I/O operation is complete, the address of the completed DTF is returned in register 2 after the completion code is posted in the DTF (see index entry *completion code*).

- Model 10 and Model 12: If the REQ key on the 5471 Printer Keyboard was pressed, and the related DTF appears in the checklist (REQK–Y or CONS–Y specified for the entry in the checklist), a completion code of X'50' is posted in the console DTF and the address of the DTF is returned in register 2.

- Model 15: If the Program Function Key 9 (PF9) was pressed and the related dummy DTF appears in the checklist (REQK–Y specified for the entry in the checklist), a completion code of X'50' is posted in the dummy DTF and the address of the DTF is returned in register 2.

Subsequent DTFs in the checklist are not tested if a completed DTF or a console/PF9 request is found.

To get completion codes posted in all DTFs in a list, continue to issue $CHK. If the check routine finds a completed BSC DTF with a $BCERR, $BCTIM, $BCDAT, $BCLOS, $BCCON, $BCRSP, or $BCADP completion code (see index entry *completion code*). MLMP logs the following message on the system logging device:



If no I/O completion is found, the $BCCMP or $BCACD completion code is posted in the last DTF in the checklist and the address of the DTF is returned in register 2 if:

- You specified RTN–Y in the $CKL macro instruction that created the first entry in the checklist ($BCCMP completion code), or

- Each entry in the checklist is closed, inactive, or has the skip indicator on (SKIP–Y specified in $CKL,) ($BCACD completion code).

Model 10 and Model 12: Otherwise, the check routine does not return control, but issues a halt and waits for an interrupt. The halt displayed on the stick lights is [ ]. After a BSCA or MLTA interrupt, the [ ] halt is automatically reset and the check routine searches the checklist again from the beginning. The check routine does not automatically reset the [ ] halt after 5471 completions; you must manually reset the [ ] halt if you are currently using the printer keyboard.

You must check for completion of all telecommunications I/O operations, including those controlled by MLTA. (For information regarding MLTA, see *IBM System/3 Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual,* GC21-7560.) Issue $CHK for every accepted GET request ($GETB), PUT request ($PUTB), and request for online test ($RFT) before issuing another $GETB, $PUTB, or $RFT for the same line. (For a description of online test, see index entry *online test.*)

You must also issue $CHK after accepted cancel GET requests ($CANB). You can issue $CHK to check for completion of printer-keyboard I/O and request operations. (Model 10 and Model 12) or PF9 key requests (Model 15).

You can save time in your program by doing some processing, that is independent of a particular I/O request, before you check the request. When $CHK is then issued, the check routine will not have to wait so long for completion to occur, and will return control to you sooner.

Model 10 and Model 12 Note: If you want to use the printer keyboard REQ key for operator interaction and your program is running in a DPF (dual programming feature) system, allocate the REQ key to your program by link-editing the program as an inquiry invoking module (ATTR–INQ in the // OPTIONS statement). For more information on link editing, see *IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561. For information on DPF, see the appropriate components reference manual listed in the *Preface.*

$CHK Macro Instruction Format

[name]	$CHK	[CKL–address]

CKL–address

Specifies the symbolic address of the first byte of a checklist or group of checklist entries. If none is given, the address of the checklist or checklist entries is assumed to be in register 2.

Techniques for Initiating Data Transfer

Poll (Control Stations)

If your station is a control station (TYPE—CS in the $DTFB macro instruction), MLMP polls the tributary stations to receive data from them. To poll a tributary station:

1. Issue a GET request ($GETB) for the receive file. When the request is accepted, MLMP will poll the first station in the polling list identified by the receive file's DTF. MLMP continues to poll stations in the list until:

   - A station responds by sending data.
   - An active station fails to respond.
   - All stations are polled the number of times specified in the LIMIT operand of the $DTFB macro instruction.
   - No active entries exist in the polling list.

2. After issuing $GETB, determine whether the GET request was accepted (see index entry *completion code*). (If you specified REJECT—address in the $GETB macro instruction, you don't have to determine whether the GET request was accepted. See index entry *$GETB macro instruction.*)

3. If the GET request was accepted by MLMP, issue a check request ($CHK).

4. Continue according to the completion code posted. If the $BCDNE completion code (successful completion) is posted, you can continue to receive data (issue $GETB) until the $BCEOT completion code (end-of-file received) is posted. Another GET request then reinitiates polling; a PUT request ($PUTB) initiates addressing.

Address (Control Stations)

If your station is a control station (TYPE--CS in the $DTFB macro instruction), MLMP addresses a tributary station to transmit data to it. To address a tributary station:

1. Issue a PUT request ($PUTB) for the transmit file. When the request is accepted, MLMP will address the station in the addressing list identified by the transmit file's DTF.

2. After issuing $PUTB, determine whether the PUT request was accepted (see index entry *completion code*). (If you specified REJECT—address in the $PUTB macro instruction, you don't have to determine whether the PUT request was accepted. See index entry *$PUTB macro instruction.*)

3. If the PUT request is accepted by MLMP, issue a check request ($CHK).

4. Continue according to the completion code posted.

   If the completion code is $BCNEG, you may have received a reverse interrupt request (RVI) from the addressed terminal. Check RVIADR (specified in the $DTFB macro instruction) to determine whether or not you received an RVI whenever the $BCNEG completion code is posted to an addressing attempt. If you did receive an RVI from the addressed terminal, poll the terminal (see index entry *poll*).

Addressing Considerations: If files for the tributaries in a network are the same in terms of record length, block length, line code, conversational mode, ITB checking, and transparency, you need only one DTF to address all the tributary stations in the network. Using only one DTF, you can address the tributaries in either of the following two ways. (Though completion codes are not discussed in the following procedures, check for completion as always.)

1. Use several addressing entries:

   a. Create an addressing entry for each tributary you want to address (see index entry *$POLB macro instruction*).
   b. Specify in the LISTAD operand of the $DTFB macro instruction the location of the first addressing entry you want to use.
   c. Issue a PUT request ($PUTB).
   d. Issue a PUT request with OPC—EOF specified (see index entry *$PUTB macro instruction*) when you are done transmitting to the tributary.
   e. Change field $BDLST in your DTF (see index entry *BSC DTF* for the format of the DTF) to point to the next addressing entry you want to use.

   Repeat steps (c) through (e) until you have finished transmitting.

2. Use only one addressing entry:

   a. Create an addressing entry for the first tributary you want to address (see index entry *$POLB macro instruction*).
   b. Specify in the LISTAD operand of the $DTFB macro instruction the location of the addressing entry.
   c. Issue a PUT request ($PUTB).
   d. Issue a PUT request with OPC—EOF specified (see index entry *$PUTB macro instruction*) when you are done transmitting to the tributary.
   e. Move the next tributary's addressing characters into the third field of the existing addressing entry (see index entry *polling/addressing list* for the format of addressing entries).

   Repeat steps (c) through (e) until you have finished transmitting.

Respond to Polling or Addressing (Tributary Stations)

A tributary station (TYPE—MP in the $DTFB macro instruction) monitors a multipoint line for polling or addressing characters only when the tributary is in control mode. MLMP establishes control mode by performing a receive-initial (RCVI) operation. The receive-initial operation is performed when the tributary issues the first PUT ($PUTB) or GET ($GETB) request. If the tributary then receives its polling or addressing characters, the tributary enters text mode and data transmission to or from the control station can proceed.

To monitor the line for polling characters, issue a PUT request. If the $BCDNE completion code is posted after an accepted PUT request, you have transmitted data to the control station.

To monitor the line for addressing characters, issue a GET request. If the $BCDNE completion code is posted after an accepted GET request, you have received data from the control station.

To re-enter control mode after transmitting or receiving end-of-file (EOT), issue a PUT or GET request. (Control mode is re-established automatically if you specified AUTORS—Y in your $DTFB macro instructions. See index entry *$DTFB macro instruction.*)

Note: A System/3 tributary station is committed at any particular time to monitoring either for polling or for addressing characters. If you are looking for one kind and the control station is transmitting the other, data transmission between you and the control station will not occur; you will be posted the $BCNCN completion code (see index entry *completion code*).

Receive Only

Issue GET requests ($GETB) for the receive file until the $BCEOT completion code is posted in the DTF (after $CHK is issued). $BCEOT indicates that you have received end-of-file (EOT) from the remote terminal. See index entry *terminating data transfer* for MLMP end-of-job information.

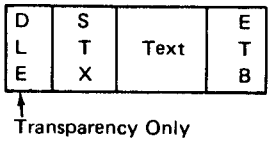
Transmit Only

Issue PUT requests ($PUTB) for the transmit file until the entire file has been transmitted, then send end-of-file (EOT). End-of-file can be sent by specifying OPC—EOF in a PUT request (see index entry *$PUTB macro instruction*), by issuing a request for a different file, or by closing ($CLOS) the file. See index entry *terminating data transfer* for MLMP end-of-job information.

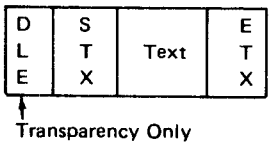
Conversational Reply

If you want to transmit or receive conversational replies, you should be aware of the BSC line conventions pertaining to the use of the ETB, ETX, and EOT line control characters. The following discussion describes the conventions as they relate to conversational replies. Whenever a BSC program violates the conventions described, the effect upon the program may not be predictable.

Transmitting from a Conversational File: When you transmit from a conversational file (CONV—Y in the $DTFB macro instruction), each PUT request for which OPC—N is specified causes transmission of one record in the following format:



PUT requests for which OPC—EOB is specified causes transmission of records in this format:



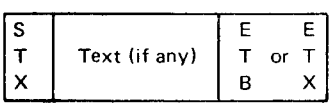
By convention, you cannot send a conversational reply in response to a message ending with ETB. If you are transmitting to a terminal that may want to send a conversational reply back to you and you are ready to accept a conversational reply, you must notify the terminal that you will accept a conversational reply by transmitting a message ending with ETX. Since line procedure requires that at least two messages be sent from one terminal before the other terminal can respond with a conversational reply, you must transmit a message ending with ETX in one of two ways:

- If you are transmitting two or more consecutive messages to the remote terminal, specify OPC—EOB in the last PUT request issued.
- If you are transmitting only one data message to the remote terminal, follow the message with a null message. A null message is STX ETX, and is transmitted by changing $BDREL in the DTF to zero, then issuing a PUT request for which OPC—EOB is specified. Before transmitting a null message, however, be sure the remote terminal can accept null messages. (If a null message is received, the $BCNDT completion code is posted. See index entry *completion code*.)

Once you have transmitted a message from a conversational file, even if it was a null message, you must wait for the $BCCRP completion code to be posted before you can issue a GET request for the file (see index entry *completion code*).

Note: If you are going to transmit from a conversational file after you have received records for the file, reset $BDREL if the record length you want to transmit is different from the record length you've been receiving.

Receiving to a Conversational File: If you are receiving messages to a conversational file (CONV—Y in the $DTFB macro instruction) and you want to transmit a conversational reply, you can determine whether the remote terminal will accept a conversational reply by looking for a message ending with ETX. By specifying OPC—BLK in the GET requests for a conversational file, you can have a message plus its control characters moved from the BSC I/O buffers to your logical buffer (addressed by the RCAD operand in $DTFB). That is, messages received will be moved to your logical buffer in the following format:



Since the length of the message received is in $BDREL of the DTF, you can use $BDREL to locate the last character received to determine whether it was ETB or ETX. If the message ended with ETB, you cannot send a conversational reply; you can send a conversational reply if the message ended with ETX.

Closing a Conversational File: After all messages for a conversational file have been sent and received, the terminal that sent the last message, even if it were a null message, must also send EOT. EOT can be sent by specifying OPC–EOF in a PUT request (see index entry *$PUTB macro instruction*), by issuing a request for a different file, or by closing ($CLOS) the file. See index entry *terminating data transfer* for MLMP end-of-job information.

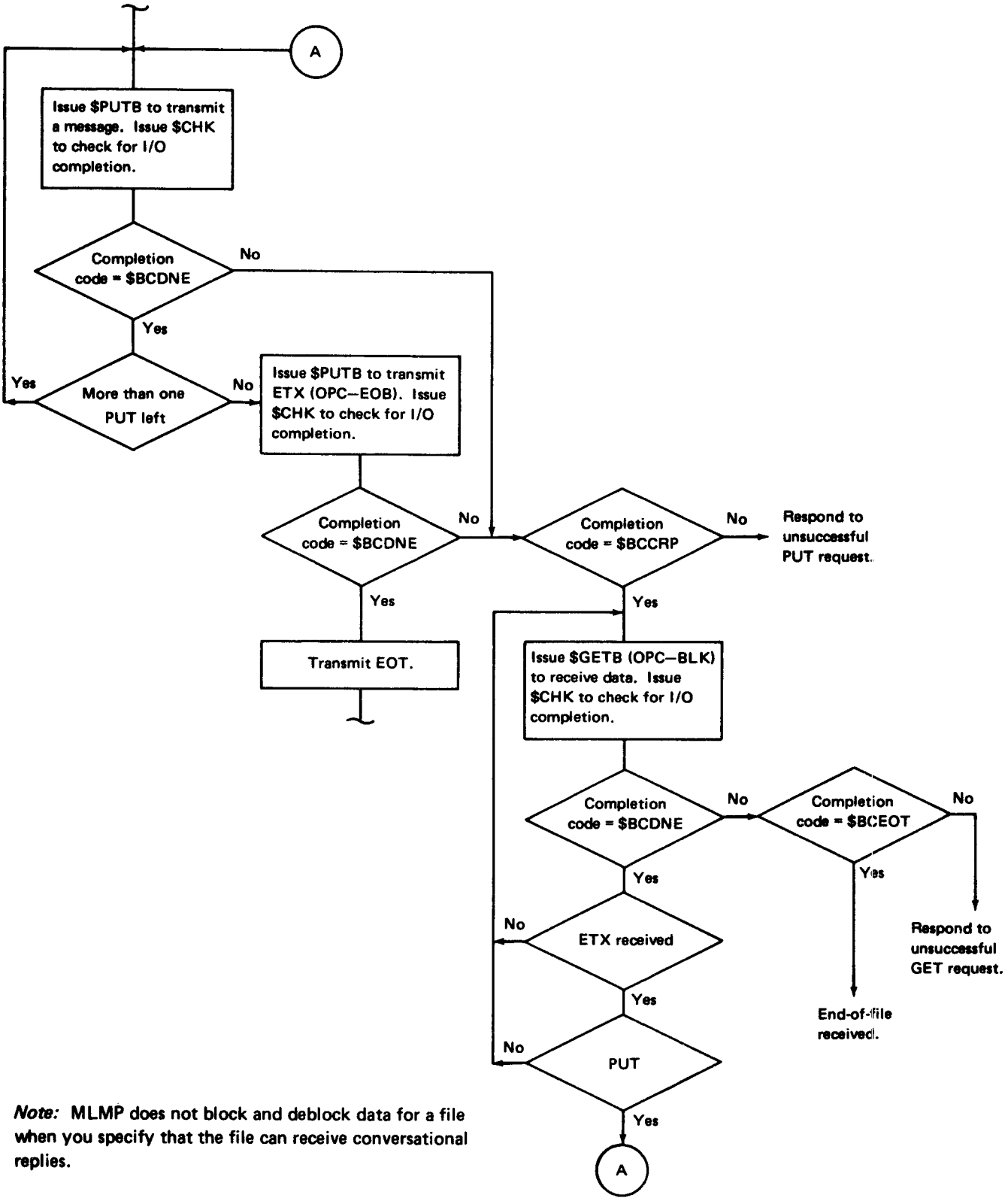
Always send EOT immediately after transmitting the last message for a file. If the terminal receiving the last message should transmit EOT before you do, your conversational file will be terminated with a permanent error posted in the DTF ($BCERR completion code).

Summary:

- If you are transmitting from a conversational file, notify the remote terminal that you will accept a conversational reply by: (1) specifying OPC–EOB in your last PUT request if you are sending two or more consecutive messages, or (2) transmitting a null message.
- If you are receiving messages to a conversational file, specify OPC–BLK in your GET requests and look for ETX at the end of messages received to determine when you can send a conversational reply.
- If all messages for your conversational file have been sent and received and you sent the last message, transmit EOT.

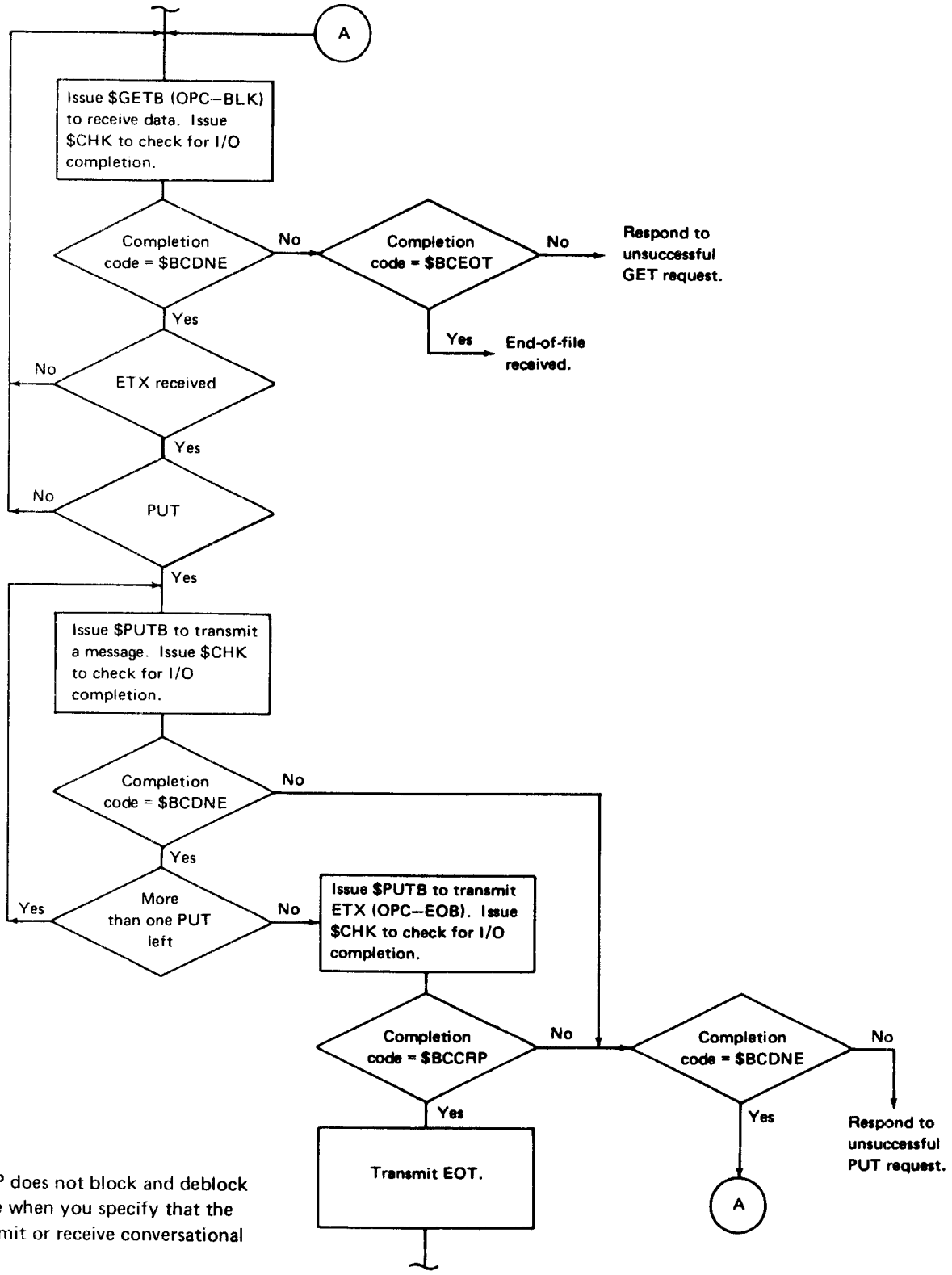
The two flowcharts that follow outline the techniques for transmit-with-reception-of-conversational-reply and receive-with-transmittal-of-conversational-reply.

Transmit with Reception of Conversational Reply: The GET and PUT requests shown are issued for the same file. See index entry *completion code* for a definition of the codes appearing in the following flowchart.



Note: MLMP does not block and deblock data for a file when you specify that the file can receive conversational replies.

Receive with Transmittal of Conversational Reply: The GET and PUT requests shown are issued for the same file. See index entry *completion code* for a definition of the codes appearing in the following flowchart.



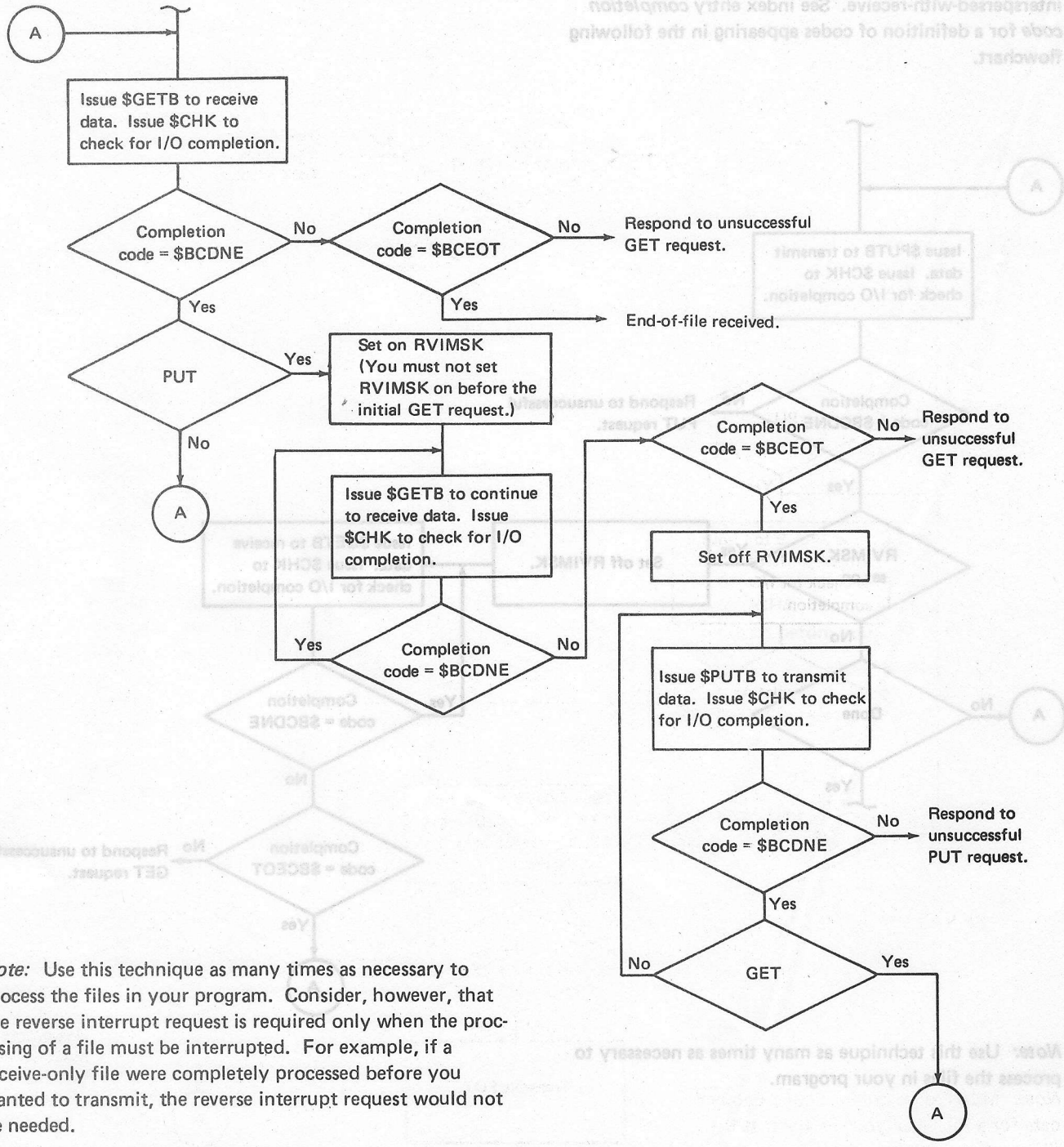
Note: MLMP does not block and deblock data for a file when you specify that the file can transmit or receive conversational replies.

Receive Interspersed with Transmit

Receive-interspersed-with-transmit differs from receive-with-transmittal-of-conversational-reply in two ways:

- One DTF is required to receive data; another is required to transmit data.
- Reverse interrupt (RVI) request is used.

The following chart outlines the technique for receive-interspersed-with-transmit. See index entry *completion code* for a definition of the codes appearing in the following flowchart.



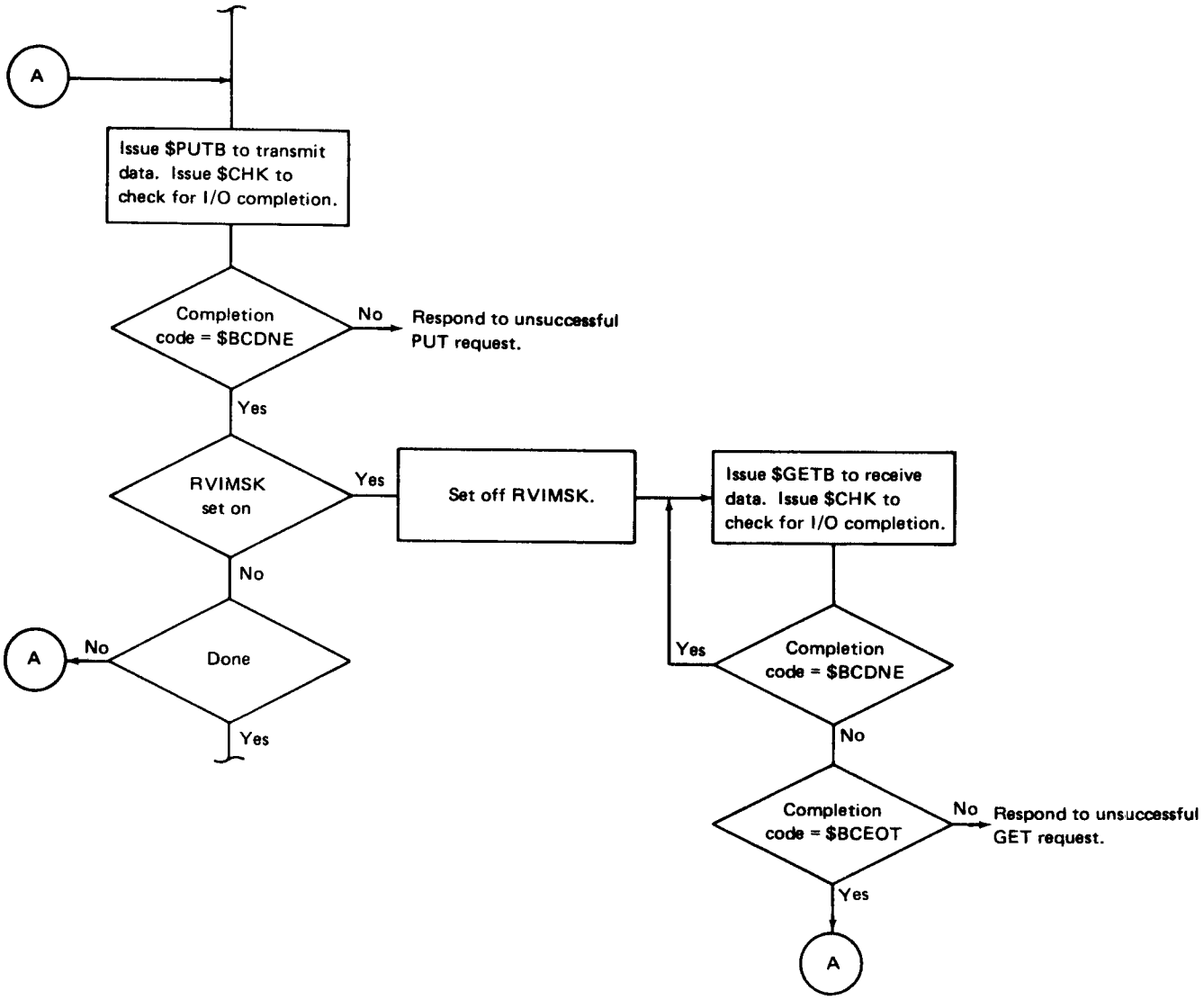
Note: Use this technique as many times as necessary to process the files in your program. Consider, however, that the reverse interrupt request is required only when the processing of a file must be interrupted. For example, if a receive-only file were completely processed before you wanted to transmit, the reverse interrupt request would not be needed.

Transmit Interspersed with Receive

Transmit-interspersed-with-receive differs from transmit-with-reception-of-conversational-reply in two ways:

- One DTF is required to transmit data; another is required to receive data.
- Reverse interrupt request (RVI) is used.

The following chart outlines the technique for transmit-interspersed-with-receive. See index entry *completion code* for a definition of codes appearing in the following flowchart.



Note: Use this technique as many times as necessary to process the files in your program.

TERMINATING DATA TRANSFER

Data transfer is terminated by:

- Terminating BSC files.
- Closing BSC files ($CLOS macro instruction).

Terminate BSC Files

Receive Files (FTYP–RCV)

Non-conversational receive files are terminated when an end-of-file indication (EOT) is received for the file ($BCEOT completion code posted). For information about terminating conversational files (CONV–Y in the $DTFB macro instruction), see index entry *closing a conversational file.*

Transmit Files (FTYP–TSM)

Transmit files are terminated by:

- A $CLOS macro instruction.
- A $PUTB macro instruction that specifies end-of-file (OPC–EOF).
- A GET request ($GETB), PUT request ($PUTB), or online test request ($RFT) for another DTF defined for the same telecommunications line.

When you issue a GET, PUT, or online test request for another DTF defined for the line, MLMP terminates the previous transmit file by transmitting any data remaining in the output buffers, then transmitting an end-of-file indication before accepting the new request.

Close BSC Files ($CLOS)

At the end of your MLMP program, you should close all files. The $CLOS macro instruction generates code that effects a branch to a system close routine. The system close routine then closes the DTFs identified in the $CLOS macro instruction.

$CLOS Macro Instruction Format

[name]	$CLOS	[DTF–address]

If you specify the keyword DTF, enter, as the parameter, the name of the DTF (file) you want to close. If the operand is not given, the address of the DTF is assumed to be in register 2.

If the DTF specified is in a chain, all DTFs following in the chain will be closed by this request. (MLTA DTFs must not be in the chain. For information regarding MLTA, see *IBM System/3 Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual,* GC21-7560.)

After $CLOS is executed, register 2 contains the address of the last DTF closed by the macro instruction.

Considerations for Closing Files

- To ensure that all data is transmitted and/or received satisfactorily, check ($CHK) for the completion of any outstanding I/O requests before you close the associated files.
- If you have established connection on a switched line with one terminal and want to use the line to communicate with a different terminal, you must close the file for the first terminal before you can establish connection with the second terminal. (When you close a file for a switched line, MLMP transmits DISC and disables the line.)
- If you reopen a BSC file that has been closed, record length, block length, and all other file attributes are the same as they were when the file was closed.

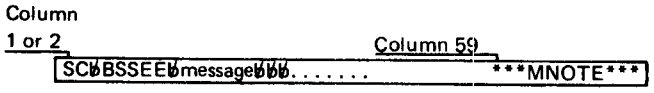
MLMP diagnoses many of the errors possible in writing macro instructions and in program execution. These errors are recorded in mnotes, halts, BSC DTF completion codes, and the BSC counters.

Also provided with MLMP are aids to help you isolate undetermined programming problems. The diagnostic aids provided are the online test, a trace module, and a dump routine.

Mnotes

Mnotes are error codes and messages pertaining to macro instruction formats (Figure 5). Mnotes are included in your assembly listing, printed beneath the macro instruction to which the mnotes apply.

MLMP Mnote Format



SC— Severity code; 04 or 08. Mnotes with a severity code of 04 are warnings, and are preceded by an asterisk (*) in column 1. Mnotes with a severity code of 08 are terminal and generate assembly errors.

B— Indicates that this mnote applies to MLMP.

SS— System message code; 00, 10, 20, 30, 40, or 50.
00—Signals miscellaneous errors not covered in the following five categories.
10—MISSING REQUIRED OPERAND. This operand is always required, or is required by another operand that is specified.
20—CONFLICTING OPERAND(S). This operand conflicts with another.
30—INVALID PARAMETER IN OPERAND. This parameter is not valid in this operand.
40—CONFLICTING PARAMETER(S). This parameter, as coded to the listed operand(s), conflicts with a parameter in another operand.
50—MISSING OPERAND. This operand is not always required, but may be required in this instance.

Note: If you get an mnote with a system message other than 00, 10, 20, 30, 40, or 50, contact your local IBM representative.

EE— Explanation code. Explanation codes identify specific operands and parameters.

Mnote Number	Related Macro Instruction	Explanation
B0001	$DTFB	Station IDs are recommended for switched lines.
B0002	$DTFB	If BUFST is specified, the BUFNO operand is ignored.
B0003	$DTFB	If CONV—Y is specified, the parameter specified in the BUFNO operand should be 1.
B1001	$POLB	The ID operand is required.
B1002	$POLB	The LEN operand is required.
B1003	$POLB	The TERMAD operand is required.
B1004	$DTFB	The RCAD operand is required.
B1005	$DTFB	If TYPE—AC is specified, the DIAL operand is required.
B1006	$DTFB	If TYPE—MC is specified, the TERMAD operand is required.
B1007	$DTFB	If TYPE—CS is specified, the LISTAD operand is required.
B1008	$DTFB	If TYPE—AC is specified, the DIALCT operand is required.
B1009	$DTFB	If RCVID is specified, the RCVCT operand is required.
B1010	$DTFB	If RCVCT is specified, the RCVID operand is required.
B1011	$DTFB	If SNDID is specified, the SNDCT operand is required.
B1012	$DTFB	If SNDCT is specified, the SNDID operand is required.
B1013	$DTFB	The RECL operand is required.
B1014	$DTFB	The BLKL operand is required.
B1015	$DTFB	The BUFST operand is required.
B1016	$DTFB	The BUFEND operand is required.
B1017	$DTFB	If TYPE—CS is specified, the ERRLOG operand is required.
B1018	$CHGB	The TYPE operand is required.
B1019	$CHGB	The DTF operand is required.
B1020	$CHGB	The NUM operand is required.
B1021	$CHGB	The CHARS operand is required.
B1023	$LOGB	The NUM operand is required.

Figure 5 (Part 1 of 3). Explanations for BSC Mnotes

Mnote Number	Related Macro Instruction	Explanation
B1024	$LOGB	The LEN operand is required.
B1025	$RFTL	The TYPE operand is required
B1026	$RFTL	The LEN operand is required.
B1027	$RFTL	The TERMAD operand is required.
B1028	$SWIB	The SELECT operand is required.
B1029	$SWIB	The LEN operand is required.
B1030	$SWIB	The STATID operand is required.
B2001	$DTFB	If TYPE—AC is not specified, the DIAL operand is invalid.
B2002	$DTFB	If TYPE—MP is not specified, the TERMAD operand is invalid.
B2003	$DTFB	If TYPE—CS is not specified, the LISTAD operand is invalid.
B2004	$DTFB	If TYPE—AC is not specified, the DIALCT operand is invalid.
B2005	$DTFB	If TYPE—AC, MC, AA, or MA is not specified, the RCVID is invalid.
B2006	$DTFB	If TYPE—AC, MC, AA, or MA is not specified, the RCVID and RCVCT operands are invalid.
B2007	$DTFB	If TYPE—AC, MC, AA, or MA is not specified, the SNDID and SNDCT operands are invalid.
B2008	$DTFB	If TYPE—AC, MC, AA, or MA is not specified, the SNDCT operand is invalid.
B2009	$DTFB	If RVIADR is specified, the RVIMSK operand is required. If RVIMSK is specified, the RVIADR operand is required.
B2010	$DTFB	If TYPE—AA or TYPE—MA is not specified, the SWLIST operand is invalid.
B2011	$DTFB	If TYPE—MP is not specified, the AUTORS operand is invalid.
B2012	$DTFB	If TYPE—CS is not specified, the POLRES operand is invalid.
B2013	$RFTL	If LEN—0 is specified, the TERMAD operand is invalid and is ignored.
B3001	$POLB	The parameter specified in the ID operand must be less than X'F0'.
B3002	$POLB	The parameter specified in the LEN operand must not be greater than 7.

Figure 5 (Part 2 of 3). Explanations for BSC Mnotes

Mnote Number	Related Macro Instruction	Explanation
B3003	$POLB	OPEN or WRAP must be specified in the LAST operand.
B3004	$DTFB	The parameter specified in the DIALCT operand must not be greater than 12.
B3005	$DTFB	The parameter specified in the RCVCT operand must not be greater than 15.
B3006	$DTFB	The parameter specified in the SNDCT operand must not be greater than 15.
B3007	$RFTL	The parameter specified in the NUM operand must be greater than 0 and less than 100.
B3008	$RFTL	The parameter specified in the LEN operand must not be greater than 7.
B3009	$RFTL	The parameter specified in the TYPE operand is invalid.
B3010	$SWIB	The parameter specified in the LEN operand must not be greater than 15.
B3011	$SWIB	The parameter specified in the SELECT operand must be less than F0.
B3012	$SWIB	The parameter specified in the LAST operand is invalid.
B3013	$TRTB	The parameter specified in the CODE operand is invalid.
B4001	$DTFB	The parameter specified with BLKL must be greater than or equal to the parameter specified with RECL.
B4002	$DTFB	If CONV–Y is specified, the parameter specified with BLKL must be equal to the parameter specified with RECL.
B4003	$DTFB	If CONV–Y is specified, ITB–Y is invalid.
B4004	$DTFB	If CODE–A is specified, TRANSP–Y is invalid.
B4005	$DTFB	If SPAN–Y is specified for a receive file, the RECSEP operand is required.
B4006	$DTFB	If neither BUFST and BUFEND nor BUFNO is specified, a default of BUFNO–1 is used.
B5001	$RFT	If online terminal test type 00 or 01 is requested, the LEN operand is required.
B5002	$RFT	If online terminal test type 00 or 01 is requested, the FROM operand is required.

Figure 5 (Part 3 of 3). Explanations for BSC Mnotes

Halts

Figure 6 shows halts issued by MLMP that require an operator response. For correct responses to the halts shown in Figure 6, see the appropriate halt/messages manual listed in the *Preface.*

On the Model 10 and Model 12, MLMP also issues the halt [ ]. However, this halt does not usually require a response, but indicates the check routine is waiting for an interrupt. See index entry *check for I/O completion ($CHK)* for more information regarding [ ] and the check routine.

Display	Meaning
P9	Error in running FDP/Convert
Y6	Error in: • Logging the control station terminal statistics • Opening a BSC file when: 1. Buffer area is not large enough 2. Record length = 0 • Attempting to initialize MLTERFIL on a disk file other than F1 • (Model 10 and Model 12) Attempting to use line 2 in an RPG II telecommunications program that was compiled on a system having only one BSC line. • DA microcode module cannot be found • DA microcode cannot be loaded correctly into adapter
Y7	Perform a manual call
Y8	Perform a manual answer

Figure 6. MLMP Halts that Require an Operator Response

Completion Codes

MLMP monitors every receive and transmit operation. MLMP indicates the status of each operation by posting a completion code in the associated DTF at $BDCMP (see index entry ***BSC DTF*** for the format of the BSC DTF). If an error occurs during polling or addressing, MLMP will retry the operation three times before posting a completion code; the error recovery retry count for other transmit and receive operations is seven (unless you specified some other retry count in the ERRCT operand of $DTFB —see index entry ***$DTFB macro instruction***).

The completion codes and the action you should take in response to them follow. The codes are divided into two groups: those posted after an I/O request ($GETB, $CANB, $PUTB, or $RFT) and those posted after a check request ($CHK).

Label	Value	Description
$BCREQ	X'00'	The request is being processed. Check for I/O completion ($CHK).
$BCUER	X'41'	You made an error in your last request. Issue the request again, or issue a request for another file. The error you made was one of the following: • You did not open the file ($OPEN) before issuing the request. • You issued an initial GET request ($GETB) for a PUT file (FTYP–TSM in $DTFB), or a initial PUT request ($PUTB) for a GET file (FTYP–RCV in $DTFB).
$BCOLT	X'48'	The request for an online test ($RFT) is invalid. Issue a valid request or close the MLMP files ($CLOS). The online test request was invalid for one of the following reasons: • You did not open the file ($OPEN) before issuing the request. • You issued an I/O request for the file before you issued $RFT for the file. • You issued $RFT for a GET file (FTYP–RCV in $DTFB). • The record length specified for the file (RECL operand in $DTFB) is greater than the block length specified for the file (BLKL operand in $DTFB). • The message length specified in $RFT is zero (LEN operand).
$BCIGN	X'4A'	Your last request was ignored because: • The previous operation was not complete, • You issued $PUTB, OPC–EOF as the first request for a file, or as the first request after issuing $GETB for the same file, or • You issued $CANB to cancel an operation that was not an initial GET. Check for completion of the previous operation ($CHK), or issue a different request for this file.
$BCCAL	X'4D'	You issued one of the following invalid requests: • A request for a new file before a previous receive file reached end-of-file. • A PUT request after a conversational reply was received. • A GET request for a conversational transmit file (FTYP–TSM, CONV–Y in $DTFB) before $BCCRP was posted. Issue a GET request for the active receive file, a PUT request for the conversational transmit file or close the MLMP files ($CLOS).

Label	Value	Description
$BCDNE	X'40'	The requested operation has been completed successfully. If the request was a GET request for a nonconversational file, data has been moved from an MLMP I/O buffer to your logical buffer. If the request was a PUT request for a nonconversational file, data has been moved from your logical buffer to an MLMP I/O buffer. If the request was a GET or PUT request for a conversational file (CONV–Y in $DTFB), data has been received or transmitted as well as moved between the MLMP I/O buffers and your logical buffer. If the request was for an online test ($RFT), the test has been completed. Issue the next request, or examine the logged results of the online test (see index entry *online test*). *Note:* Logged online test results are available only to the operator, not the MLMP program.
$BCEOT	X'42'	End-of-file has been received, or $GETB was cancelled ($CANB). Issue another request, or close the MLMP files ($CLOS).
$BCBID	X'43'	*Switched line:* Invalid ID exchange. Either your ID or the remote station's ID is invalid. Issue a request for another file or close the MLMP files ($CLOS). *Multipoint line:* The ID requested in the DTF is not in the associated polling list. You can: • Poll or address a different station. • Reinstate polling from the beginning of the list (move X'F0' or X'F1' to $BDIND). • Issue a request for another file. • Close the MLMP files ($CLOS).
$BCNEG	X'44'	All active stations in the polling or addressing list responded negatively. You can: • Poll or address a different station (reactivate an entry). See index entry *change a polling ($BCPL).* • Issue a request for another file. • Close the MLMP files ($CLOS). *Note:* $BCNEG also is used to indicate that you received an RVI in response to an addressing attempt. Whenever $BCNEG is posted after you try to address a terminal, check RVIADR (specified in $DTFB) to determine whether you received an RVI. See index entry *reverse interrupt* for information regarding RVIs.
$BCNON	X'45'	The station whose $POLB ID is in $BDIND did not respond to polling or addressing. You can: • Poll or address a different station. • Issue a request for another file. • Close the MLMP files ($CLOS). • Deactivate the polling entry.

Label	Value	Description
$BCCRP	X'46'	A record has been received from the remote station and is available in an MLMP I/O buffer (conversational reply pending). Issue a GET request for this file.
$BCNDT	X'47'	No data is available for this conversational GET request (a null message was received).
$BCOLT	X'48'	An error has occurred in executing a request for online test ($RFT), or an online test request you received was not followed by EOT. Check the format of your request (see index entry *online test*) if you just requested an online test. Use the trace module ($$BSTT) if problems persist (see index entry *trace*).
$BCNAC	X'49'	None of the entries in the polling list is active. You can: • Activate an entry and poll a specific terminal. • Activate all entries in the list and reinitiate polling from the beginning of the list. • Issue a request for another file. • Close the MLMP files ($CLOS).
$BCASC	X'4B'	An invalid ASCII character exists in the data. Close this file ($CLOS) and issue a request for another file, or close all MLMP files. *Note:* If you are transmitting or receiving ASCII data on a switched line, be sure all station IDs have been given in ASCII. If you are a control station transmitting or receiving ASCII data, be sure polling and addressing characters have been given in ASCII.
$BCNCN	X'4C'	MLMP has been unable to establish a connection with the remote station. Issue your last request again, issue a request for another file, or continue with other processing. Otherwise, close the MLMP files ($CLOS).
$BCLST	X'4E'	You did not send or receive a block of data within the time specified in the active DTF. (specified in the DLYCT operand of $DTFB). Issue a request for this or another file, or close the MLMP files ($CLOS).
$BCERR	X'4F'	MLMP encountered a permanent error condition. Some permanent errors are: • EOT received in response to data transmitted. • EOT received in response to an addressing attempt. • Forward abort received. That is: Received: TTD → EOT or DISC Transmitted: NAK • EOT or DISC is transmitted after an error recovery retry count has been exceeded.

Label	Value	Description
$BCTIM	X'50'	The remote station does not respond to attempted data transfer. Issue a request for this or another file, or close the MLMP files ($CLOS). *Note:* X'50' posted in a console DTF indicates that the operator pressed the REQ key (Model 10 and Model 12). X'50' posted in a dummy DTF indicates that the operator pressed the PF9 key (Model 15).
$BCDAT	X'51'	Data was received incorrectly (data check). Issue your last request again, or issue a request for another file. Otherwise, close the MLMP files ($CLOS).
$BCLOS	X'52'	Data received was lost because it exceeded the size of the input buffer, had no ending control character, or because no record separator was found within two contiguous blocks of spanned records. Issue your last request again, or issue a request to another station. Otherwise, close the MLMP files ($CLOS).
$BCCON	X'53'	The switched line connection with the remote station has been lost, or DISC was received in response to text. Issue your last request again, or issue a request for another file or station. Otherwise, close the MLMP files ($CLOS).
$BCRSP	X'54'	An invalid response was received from the remote station. Issue your last request again, or issue a request to another station. Otherwise, close the MLMP files ($CLOS).
$BCADP	X'55'	The BSCA is not working correctly (adapter check). Issue your last request again, or issue a request for another file. Otherwise, close the MLMP files ($CLOS). *Note:* The BSC line is disabled if: • The connection was lost ($BCCON completion code condition), • DISC was received ($BCCON completion code condition), or • Normal end of file was received ($BCEOT completion code condition).
$BCCMP	X'56'	None of the DTFs in the checklist indicates completion, but you have regained control as you requested in the first $CKL macro instruction (RTN=Y). See index entry *$CKL macro instruction.* $BCCMP is posted in the last DTF in the checklist.
$BCACD	X'57'	All the DTFs in the checklist are inactive or have been exempted from the test for completion. $BCACD is posted in the last DTF in the checklist.
$BCRLE	X'58'	The record received was larger than the specified maximum record length (RECL in $DTFB). *Note:* This completion code applies only if you are receiving spanned records separated by record separators.

Note: All data in the MLMP I/O buffers at the time one of the following completion codes is posted may be lost:

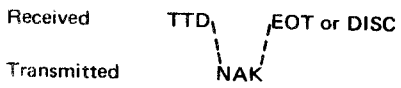
$BCASC	$BCTIM	$BCCON	$BCRLE
$BCLST	$BCDAT	$BCRSP	
$BCERR	$BCLOS	$BCADP	

The amount of data in the MLMP I/O buffers at any given time depends on record length, block size, buffer size, and the number of buffers you are using, as well as the number of I/O requests you have issued. See index entry *move mode.*

BSC Counters

MLMP compiles the following statistics as it monitors receive and transmit operations:

1. Number of text blocks sent successfully.
2. Number of text blocks received successfully.
3. Number of negative acknowledgements (NAK) received in response to text sent.
4. Number of data checks that occurred on text received.
5. Number of forward aborts received. A forward abort received is:



6. Number of EOTs ($BCERR completion code) received in response to data transmitted.
7. Number of adapter checks that occurred while transmitting.
8. Number of adapter checks that occurred while receiving.
9. Number of invalid responses received to text transmitted.
10. Number of inquiries (ENQ) sent in response to positive acknowledgements (ACK).
11. Number of blocks received from which data was lost.
12. Number of disconnect timeouts and abortive (cancel) disconnects.
13. Number of timeouts that occurred while receiving text.

For multipoint control stations the following statistics are also recorded (see the ERRLOG operand of the $DTFB macro instruction):

1. Number of unsuccessful transmissions for each terminal address.
2. Number of successful transmissions for each terminal address.

BSC counters and statistics are recorded in main storage whenever a BSC file is closed or before an online test. All BSC counters and statistics are logged to disk at end-of-job. After the BSC program is terminated, BSC counters and statistics can be displayed by the Device Counter Log-out program ($$BSDL). For operating procedures required to display the statistics, see the appropriate system operators guide listed in the *Preface.*

BSCA Terminal Log Area

You must provide a permanent file on F1 for logging control station terminal statistics (see index entry *Terminal Statistics Logging Area*). The permanent file, named MLTERFIL, requires one track. Part of MLTERFIL comprises the BSCA Terminal Log Area and is used for logging the control station terminal statistics. Another part of MLTERFIL is used for logging MLTA statistics if MLTA is present (see *IBM System/3 Multiple Line Terminal Adapter RPQ Reference and Component Description Manual,* GC21-7560).

Initializing MLTERFIL

To initialize MLTERFIL, MLMP provides, in the object library, module $$BSFI. The OCL required to initialize MLTERFIL is:

```
// LOAD $$BSFI,unit

// FILE NAME–MLTERFIL,UNIT–F1,PACK–pack,
   TRACKS–1,LOCATION–track number (optional),
   RETAIN–P

// RUN
```

MLTERFIL must be initialized after BSCA system generation, and before using MLMP or MLTA. If MLMP cannot find MLTERFIL on F1 while transmitting or receiving data for a control station, MLMP issues the Y6 halt. For a complete description of the Y6 halt, see the appropriate halt/messages manual listed in the *Preface.*

Note: MLTERFIL need be initialized only once to accomodate both MLMP and MLTA statistics. Don't initialize the file twice if you use both MLMP and MLTA.

Online Test

The online test enables you, or an IBM customer engineer, to test a line connection without interrupting data transfer on the other line. The test consists of sending a known message over a line, then determining whether the message was received accurately. When the test is completed, results are logged as follows: (see index entry *online test results*).

- Model 10 and Model 12 – Halt/Syslog is called to log the online test results on the system log device.
- Model 15 – Online test results are logged to the System History Area (SHA) and will be printed only if the printer is logged.

Requesting Online Test

To request an online test:

1. Build a test parameter list (see index entries *$RFTL macro instruction* and *online test parameter list*).
2. Provide a test message if one is required (see index entry *online test requests*).
3. Issue the $RFT macro instruction
4. Check the I/O request completion code to determine whether the request was accepted (see index entry *completion code*).
5. Check for completion of the test ($CHK).

   Note: The $BCDNE completion code, after an online test request, indicates completion of the test; $BCDNE does not indicate that the line is okay. To determine what happened on the line during the test, the operator must examine the logged results of the test (see index entry *online test results*).

$RFT Macro Instruction: The format of the $RFT macro instruction is:

[name]	$RFT	PARM–address [,FROM–address] [,LEN–decdig] [,DTF–address] [,REJECT–address]

PARM–address

Specifies the symbolic address of the first byte of the online test parameter list. See index entry *online test parameter list* for the format.

FROM–address

Specifies the symbolic address of the first byte of the test message, including control characters. Use this operand only for test types 00 and 01 (see index entry *online test parameter list* for test type descriptions).

LEN–decdig

Specifies in decimal the length of the test message, including control characters. See index entry *online test parameter list* for restrictions on the length of a test message. Use this operand only for test types 00 and 01 (see index entry *online test parameter list* for test type descriptions).

DTF–address

Specifies the symbolic address of the PUT DTF (FTYP–TSM) for which the online test request is issued. If not given, the address of the DTF is assumed to be in register 2. After $RFT is executed, register 2 contains the address of the DTF for which the online test request was issued.

REJECT–address

Specifies the symbolic address of a user routine to receive control if the online test request cannot be accepted by MLMP. You must provide the routine.

If the REJECT operand is not specified, check for the $BCREQ DTF completion code after each online test request to determine whether or not the request was accepted. See index entry *completion code.*

Accepting an Online Test Request

Valid online test requests transmitted from a remote terminal are accepted when you issue an initial GET request. MLMP then performs the test automatically, logs the results to your system logging device (see index entry *online test results*), and reissues the GET request to receive data.

If System/3 does not recognize an online test request you receive, the request is passed to you as data. The online test types recognized by System/3 are:

Test Type	Description
00	Receive and acknowledge the test message the number of times specified in bytes YY of the online test parameter list (see index entry *online test parameter list*). The formatted test request must not be more than 300 characters long. See index entry *online test requests.*
01	Transmit the test message the number of times specified in bytes YY of the online test parameter list (see index entry *online test parameter list*). The formatted test request must not be more than 300 characters long. See index entry *online test requests*).
06	Transmit 36 alphameric characters, A-Z and 0-9, the number of times specified in bytes YY of the online test parameter list (see index entry *online test parameter list*). Transmit the characters in ASCII (ASCII adapter only).
14	Transmit 36 alphameric characters, A-Z and 0-9, the number of times specified in bytes YY of the online test parameter list (see index entry *online test parameter list*). Transmit the characters in EBCDIC (EBCDIC adapter only).
23	3270 basic EBCDIC test message: This test checks all alphameric characters at a display station or printer. It checks the use of the WCC to sound the audible alarm and allows attribute field specifications to be checked at a display station. It starts a printer, printing only 40 characters to a line.
24	3270 Model 1 align EBCDIC test pattern: This test checks position alignment for the 480-character display station. It also checks the WCC to sound the audible alarm. It starts a printer, printing 40 characters to a line.
25	3270 Model 2 align EBCDIC test pattern: This test checks position alignment for the 1920-character display station. It also checks the WCC to sound the audible alarm. It will start a printer, printing 80 characters to a line.
26	3270 orders EBCDIC test message: This tests 3270 orders (SF, SBA, etc.), checks the WCC to sound the audible alarm, and uses display and intensified brightness. It starts the printer, printing 64 characters to a line.
27	3270 EBCDIC Universal Character Set test pattern: This test uses the Erase/Write command, displaying the Universal Character Set in EBCDIC. It checks the WCC to start the printer, sound the audible alarm (on a display), and print 132 characters per line on the printer. NL and EM are also tested on a printer. Display intensity is used. The SF, NL, EM, and IC orders are used.
28	3270 NL/EM EBCDIC test pattern: This test is mainly intended to test the end of message (EM) order and multiple new line (NL) orders on the printer. The WCC is checked to start the printer, sound the alarm (on a display), and print 132 characters to a line on the printer.
29-34	3270 ASCII test patterns: These tests correspond to tests 23-28 except that transmission is in ASCII.

Online Test Results

Results are logged in one of two formats, depending on whether the test message (not the test request) was transmitted or received.

Test Message Transmitted:

```
* BSC ONLINE TEST, LINE {1 or 2} {[TERMINAL ADDR HEX hex]}
MESSAGE TYPE tt, MESSAGE COUNT cc

 ACK RCVD   NAK RCVD   TIMEOUT   INVLD MSG

    xx         xx         xx         xx

* END ONLINE TEST
```

TERMINAL ADDR HEX hex identifies the terminal to which the test message was sent if the logging station is a control station (TYPE–CS in $DTFB).

tt identifies the test message type. See index entry *online test parameter list* for a description of the test types.

cc is the number of times the test message was to be transmitted. The message count is specified in the online test parameter list.

ACK RCVD xx is the number of times ACK was received as a reply to the test message.

NAK RCVD xx is the number of times NAK was received as a reply to the test message.

TIMEOUT xx is the number of 3-second timeouts recorded during the online test by the BSCA.

INVLD MSG xx is the number of invalid replies received in response to test messages sent.

Test Message Received:

```
* BSC ONLINE TEST, LINE {1 or 2} {[TERMINAL ADDR HEX hex]}
MESSAGE TYPE tt, MESSAGE COUNT cc

 TXT RCVD   DATA CHK   TIMEOUT   INVLD MSG

    xx         xx         xx         xx

* END ONLINE TEST
```

TERMINAL ADDR HEX hex identifies the terminal that transmitted the test message if the logging station is a control station (TYPE–CS in $DTFB).

tt identifies the test message type. See index entry *online test parameter list* for a description of the test types.

cc is the number of times the test message was to be transmitted. The message count is specified in the online test parameter list.

TXT RCVD xx is the number of times the test message was received correctly.

DATA CHK xx is the number of data checks recorded during the online test by the BSCA.

TIMEOUT xx is the number of 3-second timeouts recorded during the online test by the BSCA.

INVLD MSG xx is the number of test messages received incorrectly for which a data check or timeout was not recorded.

An online test only indicates line conditions existing at the time of the test. If the test reveals the presence of line problems, you must decide whether the probability of successful transmission is great enough to justify continued transmission over the line.

To discover significant trends in the appearance of line problems, consider online test results in conjunction with the BSC counters and control station terminal statistics (see index entry *BSC counters*).

Online Test Considerations

If you want to request an online test or expect to receive a request for an online test, consider that:

- The MLMP I/O buffers must be large enough to accomodate an online test request. See the RECL and BLKL operands in the $DTFB macro instruction, and index entries *online test requests* and *MLMP I/O area.*
- No data except the online test message can be sent or received over a line that is being tested until the online test is complete.
- An online test request that is not recognized by MLMP is accepted as data and moved to your logical buffer.

Considerations unique to requesting an online test are:

- An online test request for System/3 must be the first and only text message transmitted over a line. An online test request transmitted to System/3 after text has been sent will be received by System/3 and passed to you as data.
- You must transmit EOT after transmitting an online test request to System/3 if the message type is not 00. If you transmit data other than the test message before you send EOT, System/3 aborts transmission and posts the System/3 user with the $BCOLT completion code (see index entry *completion code*). The data transmitted before EOT is lost.
- $RFT should not be used unless the remote device can accept remotely initiated online test requests.
- $RFT must be issued only for a PUT DTF (FTYP–TSM in $DTFB) that is opened but not being used for data transfer.
- A multipoint control station (TYPE–CS in $DTFB) can only request test type 00 for a tributary station. See index entry *online test parameter list* for a description of the test types.
- A System/3 multipoint tributary (TYPE–MP in $DTFB) cannot request that an online test be sent to another System/3 tributary in the network.

See also index entry *how to request an online test from a 3270*.

Trace

If you are familiar with System/3 BSCA hardware and BSC line control procedures, you may find a record of the BSCA I/O sequence helpful in isolating an MLMP programming problem. MLMP provides a trace module ($$BSTT) to record I/O information after each BSCA interrupt. This information can be examined by you or an IBM customer engineer to diagnose a problem.

Once the trace module is included in your program, each MLMP I/O operation calls Trace to record the event in a trace table. The format of the table is shown in Appendix C. Dump the trace table when you are ready to examine the information recorded in it. You can use the $SNAP macro instruction to dump the table (see index entry *$SNAP macro instruction*). Dump main storage from symbolic address MTBSML to symbolic address MTBSMM, the beginning and ending addresses of the trace table. (MTBSML and MTBSMM each require that an EXTRN be defined in the program requesting the dump.)

Include Trace, Assembler

Include the trace module in your program by specifying EXTRN $$BSTT in your program, or by placing an INCLUDE card in the linkage editor input deck:

```
// INCLUDE NAME–$$BSTT,UNIT–xx
```

(xx is the unit name R1, F1, R2, or F2)

Note: If you use an INCLUDE statement to call the trace module, the overlay linkage editor generates a name not referenced error message (0L031). This error does not affect the output of the linkage editor, however, and should be ignored.

Include Trace, RPG II

If you are running under RPG II as a subroutine, $$BSMT is automatically link-edited as a dummy trace module. If you want to include the actual trace module in your program you must rename the dummy and actual trace modules. After renaming the modules, recompile your program to get the actual module link-edited. The following statements are used to rename the trace modules:

```
// LOAD $MAINT,xx

// RUN

// RENAME FROM–xx,LIBRARY–R,NAME–$$BSMT,
   NEWNAME–$$BSAV

// RENAME FROM–xx,LIBRARY–R,NAME–$$BSTT,
   NEWNAME–$$BSMT

// END
```

(xx is the unitname R1, F1, R2, or F2)

To replace the actual trace module with dummy trace module:

1. Rename the modules:

```
// LOAD $MAINT,xx
// RUN
// RENAME FROM–xx,LIBRARY–R,NAME–$$BSMT,
   NEWNAME–$$BSTT
// RENAME FROM–xx,LIBRARY–R,NAME–$$BSAV,
   NEWNAME–$$BSMT
// END
```

   (xx is the unitname R1, F1, R2, or F2)

2. Recompile your program.

Trace Considerations

- ITB interrupts, BSCA enabling operations, and BSCA disabling operations are not recorded by Trace.
- Trace entries are recorded independently of your programming operations. That is, entries are recorded when an interrupt occurs regardless of current operations occurring in your program, and can be recorded at any time, even during a snap dump (see following discussion). Consequently, be aware that entries may have been made to the trace table after a request to dump the table.
- Trace requires 512 bytes of main storage.
- For program efficiency, include Trace in your program only when you are trying to diagnose a problem.

Snap Dump Main Storage ($SNAP)

The $SNAP macro instruction generates linkage to a system storage dump routine. You must provide dump identification and dump limits. The output from the dump routine is printed on the system logging device. Output consists of:

1. The dump identification.
2. The contents of registers 1 and 2.
3. The address of the next sequential instruction after the $SNAP macro instruction.
4. The contents of main storage identified by the dump limits.

Since a printer is much faster than the console, it is recommended that the system logging device be a printer when you intend to use $SNAP.

$SNAP Macro Instruction Format

[name]	$SNAP	ID–hex,START–address, END–address

ID–hex

Specifies a 2-byte parameter to identify the dump. The parameter is printed at the beginning of the dump output.

START–address

Specifies the symbolic address of where the dump should begin.

END–address

Specifies the symbolic address of where the dump should end.

SYSTEM CONFIGURATION

The minimum system configuration and optional device support for MLMP is:

Model 8

The minimum Model 8 configuration is:

- 5408 Processing Unit Model A14 (16K bytes)
- 5444 Disk Storage Drive Model A1
- 5203 Printer Model 1
- 5471 Printer-Keyboard Model 1 or Directly attached 3741 Data Station Model 1
- Binary Synchronous Communications Adapter (BSCA), Local Display Adapter, or Integrated Communications Adapter (ICA)

Additional devices supported for the Model 8 are:

- 5408 Processing Unit Model A16 (32K), A17 (48K), or A18 (64K)
- 5444 Disk Storage Drive Model A2 or A3
- 5203 Printer Model 2 or 3
- Binary Synchronous Communications Adapter (BSCA), Local Display Adapter, or Integrated Communications Adapter (ICA)

Note: Two adapters can be present. The local display adapter, ICA, and BSCA-2 are mutually exclusive.

- Directly attached 3741 Data Station Model 2 or Programmable Work Station Model 3 or 4

Model 10

The minimum Model 10 configuration is:

- 5410 Processing Unit Model A13 (12K bytes) (if not a control station, a Model A14 (16K) is required for a control station)
- 5444 Disk Storage Drive Model 1
- 5424 MFCU Model A1
- 5203 Printer Model 1
- Binary Synchronous Communications Adapter (BSCA), or Local Communications Adapter (LCA)

Additional devices supported for the Model 10 are:

- 5410 Processing Unit Model A14 (16K), A15 (24K), A16 (32K), or A17 (48K)
- 5444 Disk Storage Drive Model 2, 3, A1, A2, or A3
- 5445 Disk Storage Model 1, 2, or 3
- 3410/3411 Magnetic Tape Subsystem Models 1, 2, and 3
- 1442 Card Read Punch Model 6 or 7
- 5471 Printer-Keyboard
- 5203 Printer Model 2 or 3
- 1403 Printer Model 2 or N1
- 5424 MFCU Model A2
- Binary Synchronous Communications Adapter (BSCA), or Local Communications Adapter (LCA) (both can be present)
- Directly attached 3741 Data Station Model 1 or 2, or Programmable Work Station Model 3 or 4

Model 12

The minimum Model 12 configuration is:

- 5412 Processing Unit Model B16 (32K bytes)
- 3340 Direct Access Storage Facility Model C2
- 5424 MFCU Model A1
- 5203 Printer Model 1
- Integrated Communications Adapter (ICA), Local Display Adapter, or Binary Synchronous Communications Adapter (BSCA)

Additional devices supported for the Model 12 are:

- 5412 Processing Unit Model B17 (48K) or B18 (64K)
- 3410/3411 Magnetic Tape Subsystem Models 1, 2, and 3
- 1442 Card Read Punch Models 6 or 7
- 5471 Printer-Keyboard
- 5203 Printer Model 2 or 3
- 1403 Printer Model 2, 5, or N1
- 5424 MFCU Model A2
- Binary Synchronous Communications Adapter (BSCA), Local Display Adapter, or Integrated Communications Adapter (ICA)

Note: Two adapters can be present. The local display adapter, ICA, and BSCA-2 are mutually exclusive.

- Directly attached 3741 Data Station Model 1 or 2 or Programmable Work Station Model 3 or 4

Model 15

The minimum Model 15 configuration is:

- 5415 Processing Unit Model A17 (48K bytes) and a 5444 Disk Stoarge Drive Model A2
  or
  5415 Processing Unit Model B17 (48K bytes) and a 3340 Direct Access Storage Facility Model A2
- 3277 Display Station (CRT/Keyboard)
- 5424 MFCU Model A1 or A2, 2560 MFCM Model A1 or A2
  or
  1442 Card Read Punch Model 6 or 7
- 1403 Printer Model 5
- Binary Synchronous Communications Adapter (BSCA), Display Adapter, or Local Communications Adapter (LCA)

Additonial devices supported for the Model 15 are:

- 5415 Processing Unit Model A18 (64K), A19 (96K), or A20 (128K) (with 5444/5445 disk units)
- 5415 Processing Unit Model B18 (64K), B19 (96K), or B20 (128K) (with 3340 disk units)
- 5415 Processing Unit Model C21 (160K), C22 (192K), C23 (224K), or C24 (256K) (with 3340 disk units)
- 5415 Processing Unit Model D19 (96K), D20 (128K), D21 (160K), D22 (192K), D23 (224K), D24 (256K) (with 3340/3344 disk units), D25 (384K), or D26 (512K)
- 3340 Direct Access Storage Facility Model B1 or B2 (available with 5415-B, -C, and -D models)
- 5444 Disk Storage Drive Model A3 (available with 5415-A models)
- 5445 Disk Storage Model 1, 2, or 3 (available with 5415-A models)
- 3410/3411 Magnetic Tape Subsystem Models 1, 2, and 3
- 1403 Printer Model 2 or N1
- 3284 Printer
- 2501 Card Reader
- Interval Timer
- Directly attached 3741 Data Station Model 1 or 2 or Programmable Work Station Model 3 or 4
- Binary Synchronous Communications Adapter (BSCA), Display Adapter, or Local Communications Adapter (LCA)

Note: Two adapters can be present.

Storage Requirements

MLMP resides in the Model 10 Disk System, Model 12, or Model 15 libraries and requires:

- 0.25K in the system nucleus. (The module $$BSIN is required in main storage at execution time – Model 10 only.)
- 5.25K of main storage to include:

  $$BMS–MLMP Data Management
  $$BMCH–Check Routine
  $$BSLG–Terminal Statistics Logging Routine
  $$BSAT–Line 2 Work Area

Additional main storage requirements for MLMP are:

0.25K for $$BSMD if AUTORS–Y is specified in $DTFB

2.00K for $$BSMA, $$BSMB, $$BSMC, and $$BSMF if POLRES–Y is specified in $DTFB

0.50K for $$BSID if the display adapter is supported

0.75K for Trace ($$BSTT) if Trace is used

- Main storage for user's code, including I/O buffers, DTFs, polling lists, $GETBs, $PUTBs, etc.
- Two cylinders of disk storage space for object code.
- Five tracks of disk storage space in the source library for MLMP macro instructions.
- One track of disk storage space for error logging. See index entry *BSCA Terminal Log Area.*

Programming Requirements

A. Model 10

- IBM System/3 Model 10 Disk System Management (5702-SC1).
- IBM System/3 Assembler (5702-AS1) or its equivalent.
- IBM System/3 Model 10 Disk System Macros Feature (Feature Number 6020 or 6021).
- IBM System/3 Model 10 Disk System Overlay Linkage Editor (Feature Number 6026 or 6027), unless MLMP programs are written as subroutine to an RPG II program.

B. Model 12

- IBM System/3 Model 12 Disk System Management (5705-SC1). This includes the System Macros and Overlay Linkage Editor.
- IBM System/3 Assembler (5705-AS1) or its equivalent.

C. Model 15

- IBM System/3 Model 15 Disk System Management (5704-SC1 or 5705-SC2). This includes the System Macros and Overlay Linkage Editor.
- IBM System/3 Assembler (5704-AS1 or 5704-AS2) or its equivalent.

MLMP Programming Considerations

- The user must define one EXTRN in every MLMP program: $$BSMS. Other required EXTRNs are generated by the MLMP macro instructions when MLMP programs are assembled.
- MLMP Data Management ($$BSMS), BSC DTFs, MLMP I/O areas, and user logical buffers must be in the root segment. They must not be overlaid. The Allocate, Rollout, and Tape End of Volume functions cannot be performad while BSCA files are open. See *IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561.

If AUTORS–Y is specified in a $DTFB macro instruction, $$BSMD must be in the root segment. If POLRES–Y is specified in a $DTFB macro instruction, $$BSMA, $$BSMB, and $$BSMF must be in the root segment.

- Binary and packed decimal data must be transmitted in transparent mode (EBCDIC only).
- A System/3 RPG II program using the RPG II Telecommunications Feature must not call an assembler subroutine to use a BSCA. (For information on writing an assembler subroutine for an RPG II program, see *IBM System/3 Basic Assembler Reference Manual,* SC2 7509.)

The MLMP user should also be familiar with the unique BSC characteristics of the terminals to be used. Some BSC characteristics are listed by machine in Appendix A. For more information regarding the terminals that can be used with MLMP, see the publicaitons listed in the front of this manual.

IBM 2972 BANKING TERMINAL SYSTEM

- Data received from the 2972 includes the terminal ID and keyboard shift characters.
- Data transmitted to the 2972 must include keyboard shift characters for upper case and lower case as well as NL, HT, and other appropriate commands.
- More than one record can be transmitted to the 2972 before you have to transmit a new line command.
- If you've been transmitting to one 2980, you must transmit EOT before transmitting data to another 2980 (see index entry *terminating transmit files*).

For more information regarding the 2972, see *Component Description: IBM 2972 Models 8 and 11 General Banking Terminal Systems,* GL27-3020.

IBM 3270 INFORMATION DISPLAY SYSTEM

Before writing an application program using a 3270, you must understand the 3270's physical characteristics and capabilities as they are described in *IBM 3270 Information Display System Component Description,* GA27-2749. After reading the 3270 component description, use this section as a guide to coding MLMP macro instructions to control and define I/O for a 3270. Use the 3270 component description to construct data areas (called data stream) to send to a 3270 to display an image or print a line, and to interpret data streams received from a 3270. Data stream formats are shown in this section, but you must have read the component description to understand the terms within the formats.

You must also understand binary synchronous telecommunications procedures as described in *General Information: Binary Synchronous Communications,* GA27-3004.

A sample MLMP program that communicates with a 3270 application is shown in Appendix B.

Polling/Addressing a 3270

There are two kinds of polling for remote 3270 devices:

- General polling. In a general poll a response is sought from any device attached to a particular control unit; the control unit has the responsibility of querying each device in turn for readiness to provide input.
- Specific polling. In a specific poll a particular device attached to a particular control unit is queried for a response.

Addressing (station selection) is always directed to a specific device.

Polling/addressing list entries are derived from the TERMAD operand in $POLB macro instructions (see index entry *$POLB macro instruction*) and are in the following formats:

General Polling:

CU Address	CU Address	7F	7F

Specific Polling and Addressing:

CU Address	CU Address	Device Address	Device Address

The control unit and device addresses are repeated because binary synchronous multipoint communications for the 3270 uses double addressing as a check against intermittent transmission line errors. The hexadecimal values for defining a polling/addressing list depend on which control unit and device are specified and whether the transmission is to be in EBCDIC or ASCII.

Usually, general polling lists are kept separate from specific polling and addressing lists.

Note: The $CANB macro instruction can be used to terminate polling. See index entry *$CANB macro instruction.*

Reading From and Writing To a Remote 3270

I/O control of a remote 3270 is maintained by a combination of $GETB and/or $PUTB macro instructions, $CHK macro instructions, and, in some cases, by special characters in the data stream called an escape command sequence. Details of initiating data transfer to and from the 3270 are discussed under *Read Operations* and *Write Operations,* following, and summarized in Figure 8.

In the discussion that follows of read and write operations, the transmission control characters (STX, ETX, etc.), device control characters, and field definition information are shown in data stream formats; these characters are described in *IBM 3270 Information Display System Component Description,* GA27-2749.

Read Operations

Reading from a remote 3270, you can:

- Read modified fields from a display station buffer after a terminal operator has completed his entry and caused an attention (for instance, by pressing the ENTER key).
- Read from a display station buffer fields modified by an operator without waiting for an attention indication.
- Read only those modified fields beginning at a specified buffer location.
- Read the entire buffer contents, both modified and unmodified data, including attribute characters.
- Read the buffer contents, both modified and unmodified data, including attribute characters, beginning at a specific buffer location.

(Attribute characters and modified and unmodified data are described in the 3270 component description.)

Each of the following five read operations can be terminated by transmitting an RVI to the terminal (see index entry *reverse interrupt).*

Read Modified Fields after Operator Action: The basic means of reading data entered by a terminal operator, this function is performed by issuing an initial GET request ($GETB) and at least one subsequent GET request. The first $GETB initiates a general or specific polling sequence. Data is read by the first and subsequent GET requests when a terminal is encountered at which the operator has done one of the following:

- Pressed one of these keys:

  ENTER
  PF (program function) keys 1-12
  PA (program attention) keys 1-3
  TEST REQUEST
  CLEAR

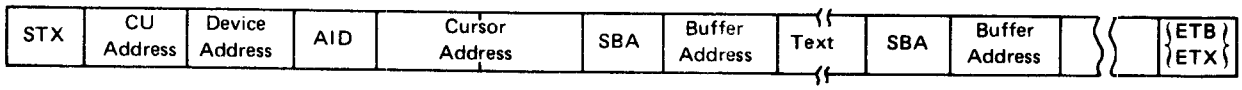
- Selected a detectable field with the selector pen. (See the *Component Description* for establishing a detectable field.)
- Inserted a card in the operator identification card reader.

All modified fields are read from the terminal buffer into the receiver's I/O buffers. A maximum of 256 bytes of data, including control characters, are read for each GET request. By issuing $GETB with OPC=BLK specified (see index entry *$GETB macro instruction*) and monitoring the logical buffer for ETX (meaning that no more message blocks remain to be read), you can determine whether all available data has been read.

After the initial $GETB, at least one more $GETB must be issued. If all data is read on the first $GETB, the next $GETB must be issued to be posted with end-of-file ($BCEOT completion code). The message read by the initial $GETB will be in one of the formats shown in Figure 7. (See the 3270 component description for an explanation of the AID, cursor address, SBA, and other data stream characters illustrated in Figure 7.)

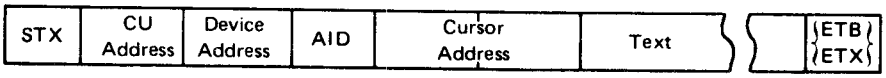
The ID byte in the BSC DTF ($BDIND) identifies the polling list entry of the responding terminal. Either $BDIND or the control unit and device address bytes in the data stream may be used to determine which device responded postively to polling.

If the operator presses the ENTER key, a PF (program function) key, or selects a detectable field with the selector pen, the message read is in this format (assuming the terminal buffer is formatted):

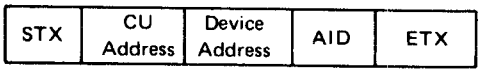


If, in the above case, no fields were modified by the operator (or already set to be modified by the program), the format of the input message is:

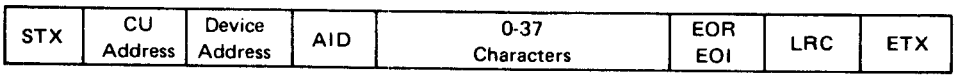
If the terminal buffer is unformatted, the input message is:



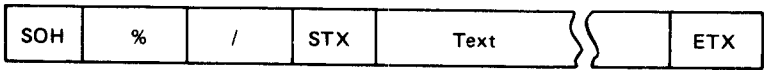
If the operator presses the CLEAR key or presses PA (program attention), the input message is:



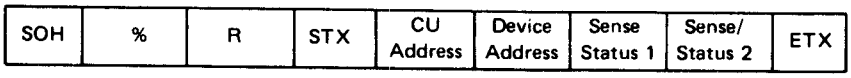
If a card or cards are read by means of the identification card reader, the input message is:



If a test request message is entered, the input message is in this format (although the application is not normally aware of it):



If a status message is read (see Figures 9 and 10), the message received is in the following format:



Note: Your program must be prepared to receive status messages.

In all of the above cases, at least one $GETB is issued to read successive blocks of the message if the message ends with ETB, or to get an EOF completion if the message ends in ETX.

A message block received as the result of subsequent $GETBs has this format (unless it is unformatted):

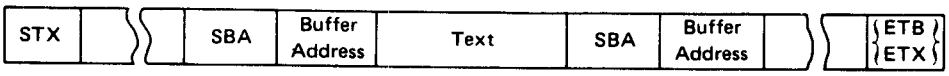


Figure 7. Message Formats Received from the 3270

Read Modified Fields: This function is similar to reading modified fields after operator action except that the operation is directed to a specific device and is performed immediately; it does not depend on an attention-causing action by the terminal operator. The purpose of this process is to read all modified fields in the device buffer.

Read modified fields by issuing:

- A $PUTB macro instruction with OPC–EOB specified (see index entry *$PUTB macro instruction*), and
- One or more $GETB macro instructions with OPC–BLK specified (see index entry *$GETB macro instruction.*

When $PUTB is issued, the logical buffer must contain:

ESC	6

Record length ($BDREL) must be 2, and the DTF must be a conversational DTF (CONV–Y sepcified in $DTFB). If the $BCCRP completion code is posted after checking the PUT request for completion ($CHK), issue $GETB.

Only modified fields are read. Issue $GETBs to read all the modified fields in the terminal buffer. That is, issue $GETB until the $BCEOT completion code is posted.

The input data stream received will be in one of the formats shown in Figure 7.

Read Modified Fields from Position: This function reads all modified fields beginning at a specified position in the device buffer. As the with read modified fields function, no operator attention-causing action is required. The process is directed to a specific device. It can be used in a manner similar to that for reading modified fields except that the program selects only a certain portion of the screen (terminal buffer) to read, even though the terminal operator may have modified other portions of the screen.

Read modified fields from position by issuing:

- A $PUTB macro instruction with OPC–EOB specified (see index entry *$PUTB macro instruction),*
- A second $PUTB macro instruction with OPC–EOB specified, and
- One or more $GETB macro instructions with OPC–BLK specified (see index entry *$GETB macro instruction*).

When the first $PUTB is issued the logical buffer must contain:

ESC	1	WCC	SBA	Buffer Address

The output data stream can also include, following the WCC, data to be written to the terminal. The WCC should be set to inhibit resetting of modified data tags, and the last buffer address should be the position from which the read modified operation is to start.

When the second $PUTB is issued, the logical buffer must contain:

ESC	6

Record length ($BDREL) must be 2, and the DTF must be a conversational DTF (CONV–Y specified in $DTFB). If the $BCCRP completion code is posted after checking the second PUT request for completion ($CHK), issue $GETB.

A maximum of 256 bytes of data, including control characters, will be read by the first $GETB. The data is read from the terminal buffer location established by the first $PUTB. The input data stream will be in one of the formats shown in Figure 7.

Read the remaining message blocks by issuing $GETB until the $BCEOT completion code is posted.

Read Buffer: This function reads the entire contents of a specified terminal buffer, including modified and unmodified fields, attribute characters, and nulls (X'00'). It is intended primarily for diagnostic use.

Read a buffer by issuing:

- A $PUTB macro instruction with OPC–EOB specified (see index entry *$PUTB macro instruction*), and
- One or more $GETB macro instructions with OPC–BLK specified (see index entry *$GETB macro instruction*).

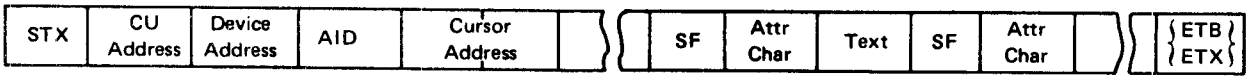
When $PUTB is issued, the logical buffer must contain:

ESC	2

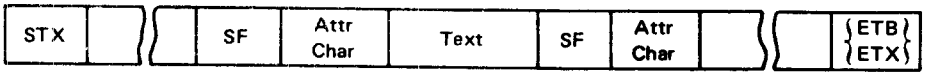
Record length ($BDREL) must be 2, and the DTF must be a conversational DTF (CONV–Y specified in $DTFB). If the $BCCRP completion code is posted after checking the PUT request for completion ($CHK), issue $GETB.

After this message has been written to the device, the $GETB reads the first message block from the terminal buffer (since only a maximum of 256 bytes can be transmitted by one $GETB macro instruction, more read operations will be required to read the entire buffer). Subsequent $GETBs are then issued to read as many remaining blocks of the terminal buffer as the program requires.

All data beginning at location 0 in the terminal buffer is read. In addition, a special character (SF) is inserted by the hardware into the input data stream to indicate the beginning of each field. The input data stream for the first message block, if the terminal buffer is formatted, appears as:



Subsequent message blocks appear as:



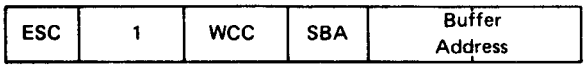
If the terminal buffer is unformatted, no SF characters are inserted since there are no fields. The input following the cursor address would consist of all character locations in the buffer, including nulls.

Read Buffer from Position: This function reads the contents of a specified terminal buffer beginning at a specified buffer position. All fields, modified and unmodified, attribute characters, and nulls (X'00') are read. As with reading a buffer, reading a buffer from position is intended primarily for diagnostic uses.

Read a buffer from position by issuing:

- A $PUTB macro instruction with OPC—EOB specified (see index entry *$PUTB macro instruction*),
- A second $PUTB macro instruction with OPC—EOB specified, and
- One or more $GETB macro instructions with OPC—BLK specified (see index entry *$GETB macro instruction*).

When the first $PUTB is issued the logical buffer must contain:

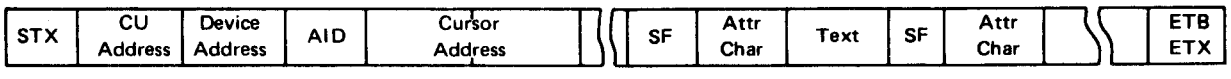


The output data stream could also include, following the WCC, data to be written to the terminal. The WCC should be set to inhibit resetting of modified data tags; and the buffer address should be the position from which the read buffer from position operation is to begin.

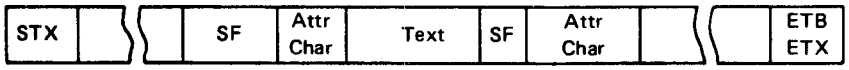
When the second $PUTB is issued the logical buffer must contain:

Record length ($BDREL) must be 2, and the DTF must be a conversational DTF (CONV—Y specified in $DTFB). If the $BCCRP completion code is posted after checking the PUT request for completion ($CHK), issue $GETB.

The $GETB reads the first message block from the terminal buffer beginning at the location specified in the first $PUTB. All data beginning at the specified location is read. In addition, a special character (SF) is inserted into the input data stream to indicate the beginning of each field. The input data stream for the first message block, if the terminal buffer is formatted, appears as:



Subsequent message blocks appear as:



If the terminal buffer from the specified beginning location is unformatted, no SF characters can be inserted, since there are no fields. The input following the cursor address would consist of all character locations in the buffer, including nulls.

Write Operations

Writing to a remote 3270, you can:

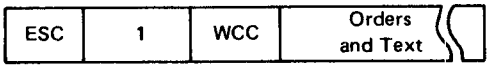
- Write data to any desired position in a display station or printer buffer.
- Erase the data presently at the device buffer (on the screen or in the printer buffer) and write data to any buffer location.
- Erase all unprotected fields in the display or printer buffer. Protected and unprotected data are described in the 3270 component description.
- Copy data in one device buffer to the buffer of a device attached to the same control unit. (For instance, have the contents of a display station buffer printed at a nearby printer.)
- Transmit conversational replies.

Note that if you are transmitting to one terminal attached to a 3271 control unit, you must transmit EOT before you can transmit to a different terminal attached to the same control unit. See index entry *terminating transmit files.* In the description of write operations that follows, consider also that whenever you write to a 3270 printer you must specify $PUTB OPC=EOW for each record, and need not transmit EOT. EOT will be transmitted automatically.

Write: The write function writes a message to a terminal (display station or printer) buffer. To write to a remote 3270:

- Issue a $PUTB macro instruction, and
- Transmit EOT (see index entry *terminating transmit files*).

When $PUTB is issued, the logical buffer must contain:



An SBA order sequence should follow immediately after the WCC, so that the write operation can be retried if an error occurs. See *IBM 3270 Information Display System Component Description,* GA27-2749, for how to write the WCC, SBA, and other data stream characters.

To send the message in blocks instead of in one data stream, additional $PUTBs may be issued with the output area in the format described above. To terminate the process, transmit EOT.

Programming Note: If a terminal operator has made an entry and pressed the ENTER key but no initial $GETB has been issued, an initial $PUTB to the display may nullify the operator input. This situation may be avoided by reserving areas of the display for operator input only (nothing will be written to these areas) and then setting the reset modified data (RMD) bit to zero (meaning do not reset modified data tags) in the write control character (WCC) of the initial $PUTB message. Setting the RMD bit off in the WCC is required because if modified data tags are reset as part of the initial $PUTB the pending attention will not be honored since there will be an indication that no fields have been modified.

Erase and Write: The erase and write function erases the buffer of a selected terminal, and then writes a message to the terminal buffer. The erasure consists of changing each character location in the buffer to X'00'. With the message omitted, the process can be used just to erase the buffer.

Erase and write by:

- Issuing a $PUTB macro instruction, and
- Transmitting EOT (see index entry *terminating transmit files*).

When $PUTB is issued the logical buffer must contain:

ESC	5	WCC	Orders and Text

An SBA order sequence should follow immediately after the WCC so that the write operation can be retried if an error occurs.

To send the message in blocks instead of in one data stream from one large output area, subsequent $PUTBs may be issued with ESC code 1 and WCC specified (see the write function).

To erase and write, the data stream placed in the logical buffer would contain an ESC code 5. To simply erase the buffer, orders and text would be omitted from the data stream.

Erase Unprotected Fields: The erase unprotected fields function sets all unprotected fields in a selected terminal buffer to nulls (X'00'). It also resets the modified data tag (MDT) bits in the attribute characters of unprotected data fields to 0, restores the keyborad, resets the AID, and repositions the cursor to the first character location in the first unprotected field in the buffer. If the buffer is completely protected, the keyboard is restored, and AID reset, the cursor moved to location 0, and no erasure takes place. (See the component description for a description of the attribute and AID characters.)

Erase unprotected fields by:

- Issuing $PUTB with OPC–EOB specified (see index entry *$PUTB macro instruction*), and
- Transmitting EOT (see index entry *terminating transmit files*).

When $PUTB is issued the logical buffer must contain:

ESC	?

Copy: The copy function selects a device and copies into its buffer the contents of the buffer of another device attached to the same 3271 control unit. Copy can be used to transfer the contents of a display station screen to a printer to get a printout of the screen or to copy the contents of one screen onto another.

To copy:

- Issue $PUTB with OPC–EOB (OPC–EOW if you are copying to a 3270 printer) specified (see index entry *$PUTB macro instruction*), and
- Transmit EOT (see index entry *terminating transmit files).*

When $PUTB is issued the logical buffer must contain:

ESC	7	CCC	From Device Address

See the component description for a description of the CCC. The from device address is the one-byte address of the device from which the data is copied.

Reply Conversationally: Any of the 3270 write operations described on previous page can be transmitted as a conversational reply to a specific terminal. Instead of issuing $GETB to receive end of file after you receive a block of text ending with ETX, issue a $PUTB to transmit text to the terminal. The $PUTB must be issued for the same DTF for which you issued the last $GETB, and CONV-Y (as well as FTYP-RCV) must have been specified for the DTF. (See index entry *conversational reply* for a detailed description of conversational techniques.)

Conversational replies save line time because you don't have to receive EOT and initiate a new addressing sequence in order to transmit a response to text received. However, to avoid two-second timeouts that lead to a possible abort situation, conversational replies should not be used if a significant amount of processing must occur on the data received before you will be ready to reply to the terminal from which the data was received. (A significant amount of processing would be, for example, 2 or 3 disk I/O operations.)

To Do This . . .	Use These Macro Instructions[1]	With This ESC Code . . .[3]
Read Modified Fields After Operator Action	$GETB OPC–BLK	None
Read Modified Fields	$PUTB OPC–EOB	6
	Then $GETB = OPC – BLK[4]	No ESC
Read Modified Fields from Position[2]	$PUTB OPC–EOB	1
	Then $PUTB OPC–EOB	6
	Then $GETBs OPC–BLK[4]	No ESC
Read Buffer	$PUTB OPC–EOB	2
	Then $GETBs OPC–BLK[4]	No ESC
Read Buffer from Position[2]	$PUTB OPC–EOB	1
	$PUTB OPC–EOB	2
	Then $GETBs OPC–BLK[4]	No ESC
Write	$PUTBs	1
	$PUTB OPC–EOF[5]	No ESC
Erase and Write	$PUTBs	5
	Then $PUTB OPC–EOF[5]	No ESC
Erase Unprotected Fields	$PUTB OPC–EOB	?
	Then $PUTB OPC–EOF[5]	No ESC
Copy	$PUTB OPC–EOB (OPC–EOW to Copy to a 3270 Printer)	7
	Then $PUTB OPC–EOF[5]	No ESC

[1] A $CHK macro instruction is required with each $GETB or $PUTB macro instruction to determine I/O completion.

[2] In order to effect the read modified fields from position and read buffer from position functions, an initial $PUTB is issued first to establish the screen position by specifying an SBA address, and then a second $PUTB is issued to send the escape command.

[3] The ESC code, in character form, is preceded by the ESC character.

[4] CONV–Y must have been specified in $DTFB for the file, and the $BCCRP completion code must have been posted for the previous $PUTB.

[5] See index entry *terminate transmit files* for other ways to transmit EOT.

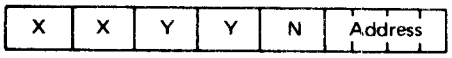
Figure 8. 3270 Read and Write Functions

How to Request an Online Test from a 3270

One 3270 can test another 3270 in the same network (or test itself) by transmitting an online test request to the System/3 control station. To initiate an online test, a 3270 display station operator must:

1. Ensure that the screen is unformatted (one way to do this is to press CLEAR, then RESET).

2. With the cursor at location 0, type in a message with the format:



   where XX is a number from 23 through 34 (see index entry *online test parameter list*) specifying the desired test; YY is a number from 01 through 99 specifying the number of times the test is to be written to the device (if the test is a printer, the test can only be sent one time); N is the number four, indicating the length of Address; where Address is a sequence of four alphameric characters specifying the control unit and device to which the test is to be sent. Alphabetic characters must be typed in upper case. Because double addressing is used, each control unit and device character must be repeated. For example, to send a test message to control unit 0, device 1, in EBCDIC transmission, the operator would press the minus (-) key twice and type two A's.

3. Press TEST REQUEST.

The test should now appear at the selected display station or printer.

After the online test is completed, the 3270 operator must inform the MLMP program that the previous display was erased for the test. (One way to do this is to press CLEAR, providing that the MLMP user's program recognizes the CLEAR key AID sequence. If the sequence is recognized, the MLMP user can issue $PUTBs to refresh the 3270 screen.)

3270 Online Test Considerations

- If System/3 does not recognize the online test request or cannot accept the request, the online test request is passed to the MLMP user as data. Be sure that the requested test number is correct, and that the MLMP I/O buffers are large enough to accommodate an online test request. See index entry *online test.*

- If, in response to a general poll by System/3, you request an online test after one or more stations in your cluster have transmitted data to the System/3, the System/3 will be unable to recognize the online test request, and will pass the request to the System/3 user as data.

- If you respond first to a System/3 general poll by requesting an online test (message type not 00) and another station in your cluster transmits data after you request the test, and before EOT has been transmitted, System/3 aborts transmission and posts the System/3 user with the $BCOLT completion code (see index entry *completion code*). The data transmitted subsequent to your request is lost.

Note: To avoid the last two situations described, try to ensure that your station will be the only one responding to a general poll by System/3 if you want to request an online test in response to a general poll.

Status/Sense Messages

Because the 3270 cannot accept data when status is pending, you must poll the 3270 for status before you can initiate or continue transmission to a 3270 on which status is pending. After you attempt to transmit to a 3270, status may be pending if the $BCERR completion code is posted or if the $BCNEG completion code is posted along with the RVI switch set on. In either case, poll the 3270 for status by issuing $GETB for the terminal for which you issued the unsuccessful $PUTB. (To avoid a polling loop, specify LAST-WRAP in $POLB and a LIMIT of 2 or 3 in $DTFB.) After receiving the status, you must issue a second $GETB to receive EOF ($BCEOT completion code).

If The Status/Sense Bytes Contain . . .	Mnemonic	This Means . . .	Applicable To . . . [1]	See This Action Number In Figure 10
X'4050'	IR	Intervention required for one of these reasons: • A command attempted to start a printer but found it not ready (out of paper, hung, etc.). The printout is suppressed. • The power is off on the printer.	3271, 3275	3A
		• The control unit received a selection addressing sequence or specific polling sequence for a device that is unavailable or went not ready during a printout. (A general poll does not respond to an unavailable or not ready indication and proceeds to the next device.) • The control unit receives a command other than diagnostic read or write, for a device that the CU has logged as unavailable/not ready.	3271	
		• The printer went not ready during a printout.	3275	
X'4060'	CR	Command reject. Receipt of an invalid or illegal 3270 channel command (for example, NOP, Sense, Select, or Copy if not installed).	3271, 3275	4
X'40C1'	OC	Operation check. Any of the following: • An illegal buffer address or an incomplete order sequence received on a write or erase/write command.	3271, 3275	4
		• CCC or from address not received on a copy command.	3271	
		• Invalid command sequence (ESC is not received in second data character position). • An I/O interface overrun is detected. This occurs during a command when a data byte is presented to the control unit by the TCU before the operation required by the previous data byte has completed.	3271, 3275	
X'40C2'	CC	Control check—timeout. A device has failed to respond to control unit communications within a specified period of time.	3271	2A

[1] In analyzing a status/sense message, the programmer may need to determine whether the device is a 3271 or a 3275. One way to do this is to compare the device specified in the status/sense message with a list of all 3275's; if the device is not found in this list, a 3271 can be assumed.

Figure 9 (Part 1 of 3). Analyzing 3270 Status/Sense Messages

If The Status/Sense Bytes Contain . . .	Mnemonic	This Means . . .	Applicable To . . . [1]	See This Action Number In Figure 10
X'40C3'	CC, OC	Control check, operation check. The condition above was detected while the control unit was executing an operation with the from device during a copy command.	3271	1B
X'40C4'	DC	Data check. Either one of the following: • An internal parity check or a cursor check occurred in either the control unit or device buffer.	3271, 3275	2A
		• A transmit parity check occurred on data sent between the device and the control unit.	3271	
X'40C6'	DC, OC	Data check, operation check. A condition above occurred while the control unit was executing an operation with the from device during a copy command.	3271	1B
X'40D1'	IR, OC	Intervention required, operation check. Either of the following: • A copy command contains a from address specifying an unavailable device. • An IR condition (see IR) is detected while the CU is executing an operation with the from device during a copy command.	3271	3B
X'4C40'	DB, US	Device busy, unit specify. The addressed device is presently busy executing an operation or a busy condition was detected previously by a command.	3271, 3275	9
X'4E40'	DB, US, DE	Device busy, unit specify, device end. A busy condition was detected. However, a device end indication means the device is no longer busy and the operation should be retried.	3271, 3275	2A
X'C140'	TC	Detection of a BSC error on the TCU transmission.	3275	11
X'C4C1'	OC, US	Operation check, unit specify. A from address on a copy command specified a device with a locked buffer. (The device was not authorized to be copied from.)	3271	12

[1] In analyzing a status/sense message, the programmer may need to determine whether the device is a 3271 or a 3275. One way to do this is to compare the device specified in the status/sense message with a list of all 3275's; if the device is not found in this list, a 3271 can be assumed.

Figure 9 (Part 2 of 3). Analyzing 3270 Status/Sense Messages

If The Status/Sense Bytes Contain . . .	Mnemonic	This Means . . .	Applicable To . . . [1]	See This Action Number In Figure 10
X'C240'	DE	Device end. Signals that a previously detected busy condition has gone not busy, a not ready device has gone ready, or a not available device is now available.	3271, 3275	N/A
X'C250'	IR, DE	Same as X'4050'.		3A
X'C2C4'	DC, DE	Same as X'40C4'.		2A
X'C2C8'	EC, DE	Equipment check, device end. A mechanical hang or a character generator read out error on the printer.	3275	6
X'C2D8'	IR, EC, DE	Same as above.	3275	6
X'C4C4'	DC, US	Same as X'40C4'.		2A
X'C4C5'	DC, OC, US	Same as X'40C4' occurring while the CU was executing an operation with the from device on a copy command.	3271	2B
X'C6C4'	DC, US, DE	Same as X'40C4'.	3271	7
X'C6C8'	EC, US, DE	Same as X'C2C8'.	3271	6
X'C6D8'	IR, EC, US, DE	Same as X'C2C8'.	3271	6
X'C840'	DB	The addressed device is presently busy executing an operation or a busy condition was detected previously by a command. The device is or was busy executing a printout, accepting data from an identification card reader, or performing keyboard functions. Set under either of these conditions: • A command is addressed to a busy device. • A specific poll sequence makes a status poll to a device and finds it busy.	3271, 3275	8
X'C8C1'	DB, OC	The same as X'C840' and the CU was executing an operation with the from device busy during a copy command.	3271	10

[1] In analyzing a status/sense message, the programmer may need to determine whether the device is a 3271 or a 3275. One way to do this is to compare the device specified in the status/sense message with a list of all 3275's; if the device is not found in this list, a 3271 can be assumed.

Figure 9 (Part 3 of 3). Analyzing 3270 Status/Sense Messages

Action Number	Programmer Action
1A	Execute a new address selection sequence and retransmit the message starting with the command sequence which was being executed when the error occurred. If the operation is not successful after two retries, consider this an unrecoverable error and follow procedure 5A.
1B	Same as 1A except follow procedure 5B after two retries.
1C	Same as 1A except retransmit the entire failing chain of commands.
2A	It is suggested that the user reconstruct the entire screen buffer image if this is possible and retry the failing chain of commands (within the BSC sequence of operations). If the information in the screen buffer is such that it cannot or need not be reconstructed, the operation may still be retried. If the operation is not successful after three retries, consider this an unrecoverable error and follow procedure 5A.
2B	The error occurred during the execution of a copy command. Execute procedure 2A except that it is the buffer of the from device specified by the copy command that should be reconstructed. After three retries, execute procedure 5B.
3A	The error indicates that the printer is out of paper, has the cover open, has the print mechanism hung, or the device is unavailable. Wait for the display operator or system operator to intervene and mechanically ready the printer. Then retry the printout by issuing a $PUTB with the WCC and no data stream. (There is no data error and the data is still intact in the device buffer and can be sensed.) Otherwise, execute procedure 2A.
3B	The error indicates that the from device specified in a copy command is unavailable. The device address associated with the error status/sense information is not the one requiring readying. The device requiring corrective action is the from device specified in the copy command. This from device should be determined and made ready. Then execute procedure 1B.
4	An unrecoverable programming error has occurred. Examine the data stream to locate the problem.
5A	Request maintenance on the device giving the trouble. After repair, attempt to reconstruct the screen buffer image if possible, starting with an erase/write command in order to correct a missing or multiple cursor situation in the device buffer. Retry the failing operations as done in the procedure previous to 5A.
5B	The from device specified by the copy command in the failing operation is malfunctioning. The from device should be determined from the data stream information, and maintenance should be requested on the device. After repair, reconstruct the screen buffer image, if possible. The sequence of commands used to reconstruct the image should start with an erase/write command to correct a missing or multiple cursor situation in the device buffer. Retry failing operations as done in the procedure previous to 5B.
6	The error occurred during a printing and indicates either a character generator readout error or a print mechanism hang. There is no data error. The proper error recovery procedure is application dependent, since the user may or may not want a new printout. If a new printout is required, follow procedure 3A.
7	A data error occurred in the device buffer during printing; follow procedure 2A.

Figure 10 (Part 1 of 2). Suggested Actions Based on 3270 Status/Sense Messages

Action Number	Programmer Action
8	A specific poll detected that the addressed device is busy. Periodically issue a specific poll to pick up the device end status/sense bit which is sent by the device to the TCU when the device becomes not busy (unless this status change is detected on a selection addressing sequence).
9	A command was erroneously addressed to a busy device. Periodically issue an initial $GETB with a specific poll to pick up the device end status/sense bit, which is sent by the device to the TCU when the device becomes not busy; then follow procedure 2A.
10	This error indicates that in attempting to execute a copy command the from device was found to be busy. Execute procedure 1A when the from device is not busy. (A specific poll read picks up the device end status/sense bit.) The device address associated with the status/sense message is the address of the to device and not that of the busy from device.
11	A BSC error was detected during a text transmission from the TCU. Follow procedure 2A if the failing command is a write command which has a data stream of more than 1 byte or if it is in a chain of commands and one of the previous commands in the chain is a write command without an SBA order immediately following the WCC character. In all other cases, follow procedure 1C. If, after following the above retry procedure, the problem is not corrected, follow procedure 5A.
12	An unauthorized attempt was made to read data. An effort was made to execute a copy command but access to the from device data was not authorized. The device address associated with the error status/sense bits is that of the to device.

Figure 10 (Part 2 of 2). Suggested Actions Based on 3270 Status/Sense Messages

Polling/Addressing a 3270 via the Display Adapter

The display adapter is supported by Models 8, 12, and 15 only. User assembler programs for the display adapter are coded similar to programs for the BSC adapter with the following exceptions:

- The display adapter requires another link-edit of your R-module using a new INCLUDE card:

  // INCLUDE NAME-$$BSID,UNIT-xx
  (xx = the unit name R1, F1, R2, F2)

- The display adapter emulates both the System/3 BSCA line 2 attachment and the 3271 control unit of the 3270 system.

- The display adapter provides a continuous poll function which polls the specified devices in the order given. (Devices can be repeated more than once to set up priorities.)

- The poll/address line buffer must contain valid device addresses or you will get a unit check (see index entry *$POLB macro instruction).*

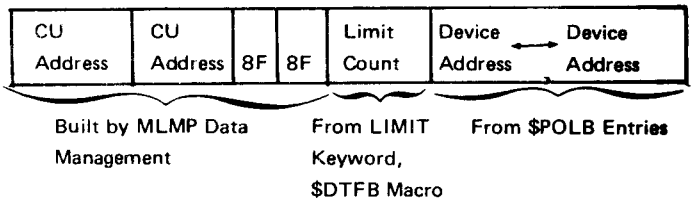
A maximum of 12 unique device addresses can be specified for the Models 8 and 12, and a maximum of 30 can be specified for the Model 15.

- 255 total polling entries can be specified in the user program (see index entry *$POLB macro instruction).*

- The display adapter is supported on line 2 only.

- The display adapter addressing (station selection) is always directed to a specified device.

- Display adapter continuous polling line buffer:



- Address and polling characters for the display adapter are 6060xxxx and 4040xxxx respectively (xxxx are the device address characters).

See Appendix B for samples.

IBM 3735 PROGRAMMABLE TERMINAL

Form Descriptor Convert Routine ($$BSCN)

The Form Descriptor Convert routine (FDP/Convert) is provided with MLMP to convert form descriptor programs (FDPs) generated on OS or DOS to a format suitable for transmission from System/3 to an IBM 3735 Programmable Terminal. FDP/Convert converts to a System/3 file an FDP generated by OS or DOS FDP utility programs and punched into cards by IEBPTPCH or an equivalent punch routine. The System/3 file can then be transmitted to a 3735 by an MLMP program. (For information on how to generate FDPs on System/3, see *IBM Systems 3735 Support Program Coding Manual,* GC21-5096.)

Whatever the punch routine used to generate the input deck for FDP/Convert, the data must be punched in the following format:

- Six cards: data in columns 1-72, sequence number in columns 73-80;
- Every seventh card: data in columns 1-44, sequence number in columns 73-80.

That is, seven cards of input for FDP/Convert must contain a maximum of 476 bytes, and be equivalent to one OS or DOS FDP disk record. (OS FDP disk records always contain a maximum of 476 bytes. DOS FDP disk records, however, can contain a maximum of 486 bytes. The first 10 bytes of each DOS FDP disk record contain an 8-byte name and a 2-byte sequence number. If you are punching FDP records generated on DOS, do not punch the first 10 bytes of the disk records.)

Note: FDP/Convert checks the sequence of input decks. If an input card is out of sequence, FDP/Convert issues the 'P9' halt. See the appropriate halt/messages manual listed in the *Preface* for a complete description of the 'P9' halt. For information regarding IEBPTPCH, see *IBM System/360 Operating System Utilities,* GC28-6586.

FDP/Convert can convert:

- 80-column FDP cards to 96-column expanded format cards.
- 80-column FDP cards to a user-specified consecutive 5444 disk file.
- 96-column expanded format FDP cards to a user-specified consecutive 5444 disk file.

Note: Only the disk files are formatted for transmission to a 3735.

80-Column to 96-Column

For each 80-column input card, FDP/Convert generates a maximum of two 96-column cards. The information in the input card, except null characters, is expanded for printability and punched in columns 1-86 of a 96-column card. (Null characters [X'4070'] are inserted into FDP records by OS and DOS to pad a record to 476 bytes.)

As an example of the expansion for printability, the 80-column input:

Column	1	2	3	4
Hex value	4	7	4	7
	3	2	E	0

Is converted by FDP/Convert to:

Column	1	2	3	4	5	6	7	8
Hex value	F	F	F	F	F	C	F	F
	4	3	7	2	4	5	7	0
Character	4	3	7	2	4	E	7	0

To generate a deck of 96-column FDP cards from a deck of 80-column cards:

1. Place, as shown, the following OCL statements and data in the 1442 Card Reader/Punch:

   ```
   // LOAD $$BSCN,unit
   // RUN
   // CONVERT TO-MFCU,FDPNUM-nnn
   80-column FDP deck(s)
   /*
   ```

2. Place blank 96-column cards in MFCU2.
3. Begin reading from the 1442.

The FDP number specified in the // CONVERT statement (FDPNUM) is punched in columns 89-91 of each 96-column card. Each 96-column card also contains a sequence number punched in columns 92-96.

80-Column to Consecutive 5444 Disk

FDP/Convert builds a maximum of one 476-byte disk record for every seven 80-column FDP cards. Null characters are deleted.

To generate from an 80-column card deck a consecutive 5444 disk file containing one or more FDPs:

1. Place, as shown, the following OCL statements and data in the 1442 Card Reader/Punch:

```
// LOAD $$BSCN,unit
// FILE NAME–$WORK,UNIT–xx,PACK–pack,
   LABEL–your filename,RECORDS–number,
   RETAIN–{P}
          {T}
          {S}
// RUN
// CONVERT TO–DISK
80-column FDP deck(s)
/*
```

2. Begin reading from the 1442.

RECORDS-number in the // FILE OCL statement specifies the number of records required to contain the FDP file. The maximum number of records required is n divided by 7, where n = the number of cards in the 80-column input deck.

Expanded 96-Column to Consecutive 5444 Disk

FDP/Convert repacks expanded format 96-column FDP cards, then builds 476-byte disk records from the repacked data.

To generate from a 96-column card deck a consecutive 5444 disk file containing one or more FDPs:

1. Place, as shown, the following OCL statements and data in MFCU1:

```
// LOAD $$BSCN,unit
// FILE NAME–$WORK,UNIT–xx,PACK–pack,
   LABEL–your filename,RECORDS–number,
   RETAIN–{P}
          {T}
          {S}
// RUN
// CONVERT TO–DISK      (The // CONVERT
                         statement is optional.)
96-column FDP deck(s)
/*
```

2. Begin reading from the MFCU.

RECORDS-number on the // FILE OCL statement specifies the number of records required to contain the FDP file. The number of records required is n divided by 952, where n = total number of columns, excluding columns 89-96, punched in the 96-column input deck. The number of records required is approximately one disk record for every eleven input cards.

FDP/Convert Considerations

- FDP/Convert requires a 1442 Card Reader/Punch to convert 80-column FDP cards. If you don't have a 1442, you can use an IBM Business Systems Center or another customer installation to run FDP/Convert. Arrangements for using an IBM Business Systems Center can be made through your IBM representative or the IBM branch office serving your locality.
- All FDPs must be generated initially by System/3 Model 10 Disk System, OS, or DOS. You can have your FDPs generated at an IBM Business Systems Center or at a customer installation. Arrangements for using an IBM Business Systems Center can be made through your IBM representative or the IBM branch office serving your locality.
- To avoid billable maintenance by IBM, be sure your FDPs are generated on the current version/modification level of System/3, OS, or DOS. IBM may charge you to fix an FDP problem if the FDP was generated on an old level of System/3 OS, or DOS, and the problem is fixed on the current level.

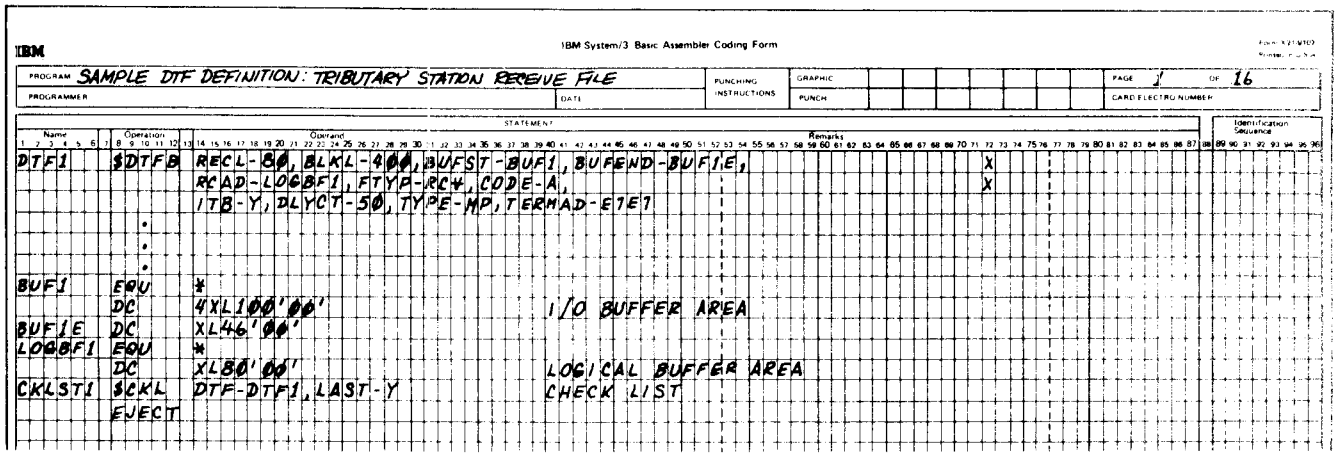
Additional 3735 Considerations

The 3735 has unique data formatting requirements as described in *IBM 3735 Programmer's Guide,* GC30-3001. You must be familiar with these requirements before you attempt to communicate with the 3735.

This appendix contains examples of MLMP macro instructions, coded with a minimum of associated assembler statements, and a complete sample program written to communicate with the IBM 3270 Information Display System.

Sample MLMP Macro Instructions

In pages 8-16 of the following sample MLMP macro instructions, the label $DTF is equated to XR2 by the $COMN macro instruction, and other labels used are equated by the $DTOB macro instruction.

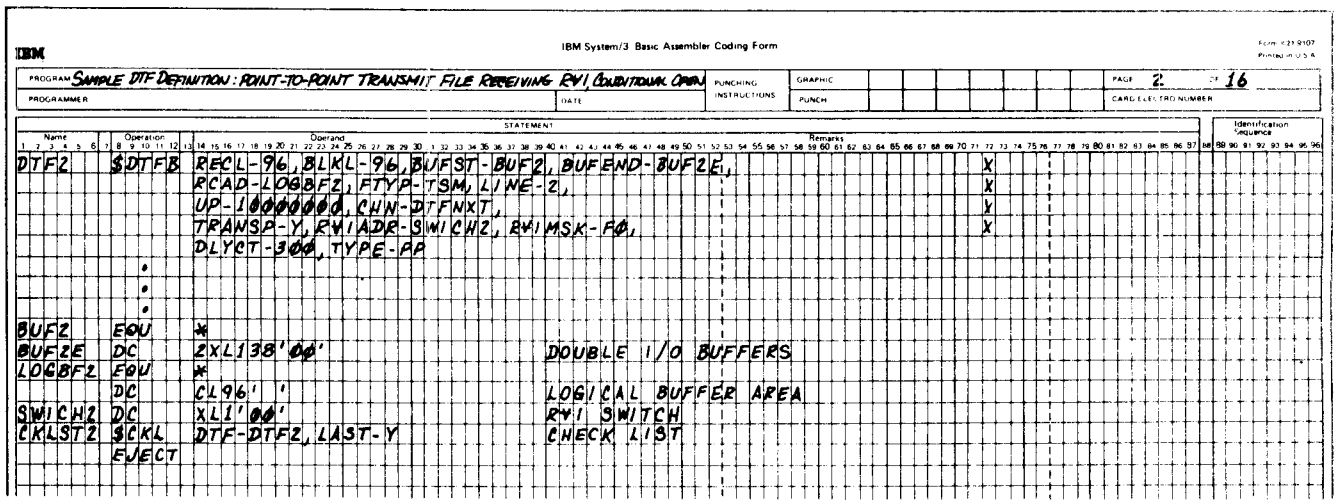


IBM System/3 Basic Assembler Coding Form

PROGRAM SAMPLE DTF DEFINITION: TRIBUTARY STATION RECEIVE FILE — PUNCHING INSTRUCTIONS — GRAPHIC — PUNCH — PAGE 1 OF 16

PROGRAMMER — DATE — CARD ELECTRO NUMBER

Name	Operation	Operand	Remarks	Identification Sequence
DTF1	$DTFB	RECL-80,BLKL-400,BUFST-BUF1,BUFEND-BUF1E,	X	
		RCAD-LOGBF1,FTYP-RCV,CODE-A,	X	
		ITB-Y,DLYCT-50,TYPE-MP,TERMAD-E7E7		
		.		
		.		
		.		
BUF1	EQU	*		
	DC	4XL100'00'	I/O BUFFER AREA	
BUF1E	DC	XL46'00'		
LOGBF1	EQU	*		
	DC	XL80'00'	LOGICAL BUFFER AREA	
CKLST1	$CKL	DTF-DTF1,LAST-Y	CHECK LIST	
	EJECT			

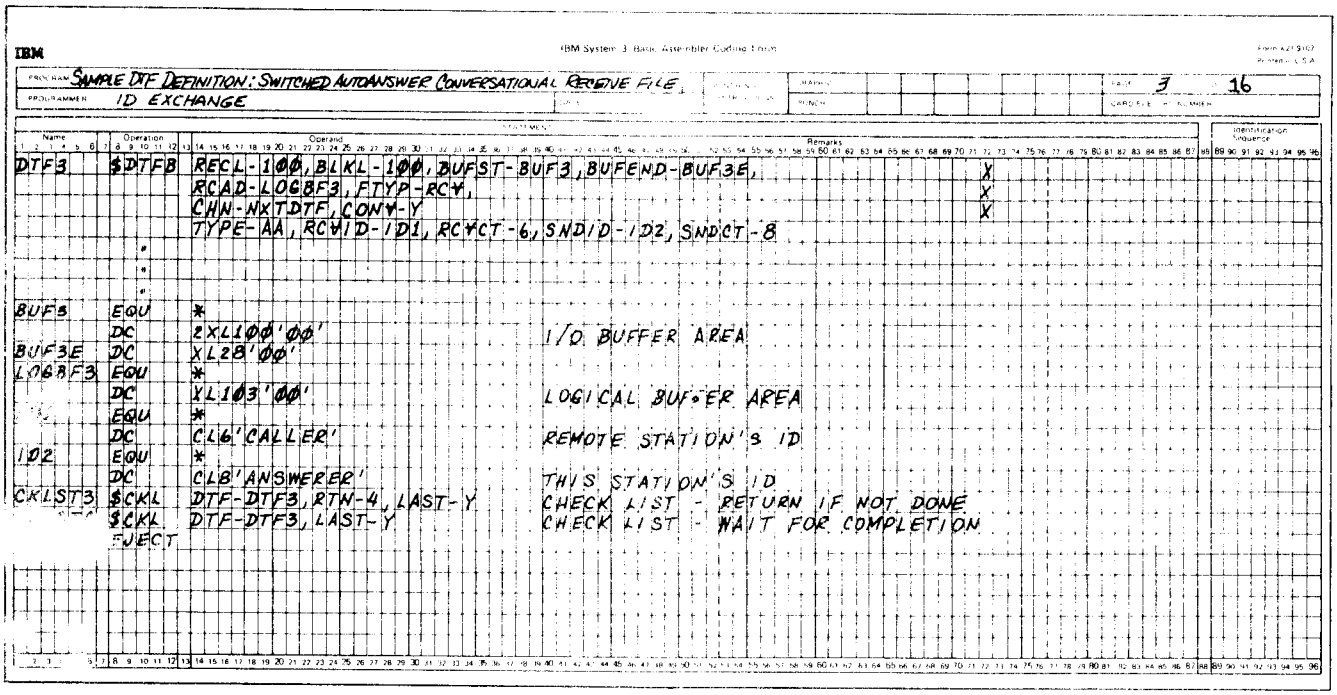


IBM System/3 Basic Assembler Coding Form

PROGRAM SAMPLE DTF DEFINITION: POINT-TO-POINT TRANSMIT FILE RECEIVING RVI, CONDITIONAL OPEN — PUNCHING INSTRUCTIONS — GRAPHIC — PUNCH — PAGE 2 OF 16

PROGRAMMER — DATE — CARD ELECTRO NUMBER

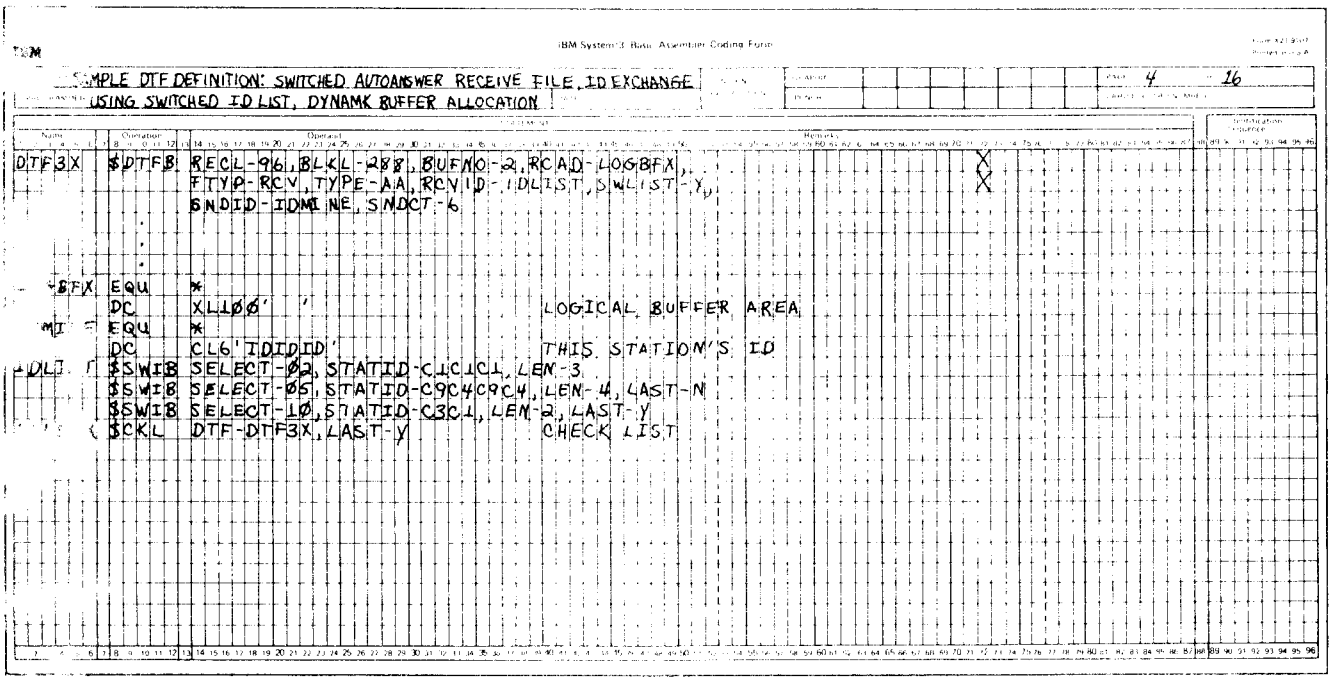
Name	Operation	Operand	Remarks	Identification Sequence
DTF2	$DTFB	RECL-96,BLKL-96,BUFST-BUF2,BUFEND-BUF2E,	X	
		RCAD-LOGBF2,FTYP-TSM,LINE-2,	X	
		UP-10000000,CHN-DTFNXT,	X	
		TRANSP-Y,RVIADR-SWICH2,RVIMSK-F0,	X	
		DLYCT-300,TYPE-AP		
		.		
		.		
		.		
BUF2	EQU	*		
BUF2E	DC	2XL138'00'	DOUBLE I/O BUFFERS	
LOGBF2	EQU	*		
	DC	CL96' '	LOGICAL BUFFER AREA	
SWICH2	DC	XL1'00'	RVI SWITCH	
CKLST2	$CKL	DTF-DTF2,LAST-Y	CHECK LIST	
	EJECT			



IBM System/3 Basic Assembler Coding Form

PROGRAM: SAMPLE DTF DEFINITION: SWITCHED AUTOANSWER CONVERSATIONAL RECEIVE FILE
PROGRAMMER: ID EXCHANGE
PAGE 3 16

Name	Operation	Operand	Remarks
DTF3	$DTFB	RECL-100,BLKL-100,BUFST-BUF3,BUFEND-BUF3E,	
		RCAD-LOGBF3,FTYP-RCV,	
		CHN-NXTDTF,CONV-Y	
		TYPE-AA,RCVID-ID1,RCVCT-6,SNDID-ID2,SNDCT-8	
BUF3	EQU	*	
	DC	2XL100'00'	I/O BUFFER AREA
BUF3E	DC	XL28'00'	
LOGBF3	EQU	*	
	DC	XL103'00'	LOGICAL BUFFER AREA
	EQU	*	
	DC	CL6'CALLER'	REMOTE STATION'S ID
ID2	EQU	*	
	DC	CL8'ANSWERER'	THIS STATION'S ID
CKLST3	$CKL	DTF-DTF3,RTN-4,LAST-Y	CHECK LIST - RETURN IF NOT DONE
	$CKL	DTF-DTF3,LAST-Y	CHECK LIST - WAIT FOR COMPLETION
	END		



IBM System/3 Basic Assembler Coding Form

SAMPLE DTF DEFINITION: SWITCHED AUTOANSWER RECEIVE FILE, ID EXCHANGE
USING SWITCHED ID LIST, DYNAMK BUFFER ALLOCATION
PAGE 4 16

Name	Operation	Operand	Remarks
DTF3X	$DTFB	RECL-96,BLKL-288,BUFNO-2,RCAD-LOGBFX,	
		FTYP-RCV,TYPE-AA,RCVID-IDLIST,SWLIST-Y,	
		SNDID-IDMINE,SNDCT-6	
LOGBFX	EQU	*	
	DC	XL100''	LOGICAL BUFFER AREA
MI	EQU	*	
	DC	CL6'IDIDID'	THIS STATION'S ID
IDLIST	$SWIB	SELECT-02,STATID-C1C1C1,LEN-3	
	$SWIB	SELECT-05,STATID-C9C4C9C4,LEN-4,LAST-N	
	$SWIB	SELECT-10,STATID-C3C1,LEN-2,LAST-Y	
	$CKL	DTF-DTF3X,LAST-Y	CHECK LIST
	END		

IBM — IBM System/3 Basic Assembler Coding Form

PROGRAM: SAMPLE DTF DEFINITION: SWITCHED AUTOCALL RECEIVE FILE, ID EXCHANGE — PAGE 5 OF 16

```
DTF4     $DTFB  RECL-476,BLKL-476,BUFNO-1,                               X
                RCAD-LOGBF4,FTYP-RCV,CODE-E,                             X
                ITB-N,TRANSP-N,                                          X
                TYPE-AC,DIAL-PHONE,DIALCT-7,                             X
                RCVID-ID3,RCVCT-5,SNDID-ID4,SNDCT-5,                     X
                SPAN-Y,RECSEP-1E
            .
            .
            .
LOGBF4   EQU    *
         DC     CL132' '                  LOGICAL BUFFER AREA
PHONE    EQU    *
         DC     CL7'7654321'              PHONE NO FOR AUTOCALL
ID3      EQU    *
         DC     CL5'Ø1234'                REMOTE STATION'S ID
ID4      EQU    *
         DC     CL5'56789'                THIS STATION'S ID
CKLST4   $CKL   DTF-DTF4,LAST-Y           CHECK LIST
         EJECT
```

IBM — IBM System/3 Basic Assembler Coding Form

PROGRAM: SAMPLE DTF DEFINITION: CONTROL STATION RECEIVE FILE, TRANSMITTING RVI — PAGE 6 OF 16

```
DTF5     $DTFB  RECL-256,BLKL-256,BUFNO-2,                               X
                RCAD-LOGBF5,FTYP-RCV,RVIADR-SWICH5,RVIMSK-Ø1,            X
                TYPE-CS,LISTAD-POLIST,ERRLOG-LOG5
            .
            .
            .
LOGBF5   EQU    *
         DC     CL256' '                  LOGICAL BUFFER AREA
SWICH5   DC     XL1'ØØ'                   RVI SWITCH
LOG5     $LOGB  NUM-3,LEN-2               *STATISTICS LOG AREA
POLIST   $POLB  ID-ØØ,TERMAD-C2C2,LEN-2   POLLING LIST, POLLING S/3 BB
         $POLB  ID-Ø1,TERMAD-C3C3,LEN-2                             CC
         $POLB  ID-Ø2,TERMAD-C4C4,LEN-2,LAST-WRAP                   DD
CKLST5   $CKL   DTF-DTF5,LAST-Y           CHECK LIST
         EJECT
```

IBM System/3 Basic Assembler Coding Form

SAMPLE DTF DEFINITION: CONTROL STATION TRANSMIT FILE CAN RECEIVE RY1

PAGE 7 OF 16

```
DTF6     $DTFB  RECL-48,BLKL-48,BUFST-BUF6,BUFEND-BUF6E,.          X
                RCAD-LOGBF6,FTYP-TSM,CODE-E,LINE-2,               X
                CHN-NEXT1,RY1ADR-SW1CH6,RY1MSK-FF,                X
                TYPE-CS,LISTAD-ADRLST,ERRLOG-LOG6,LIMIT-7
*
*
*
BUF6     EQU    *
BUF6E    DC     XL90'00'                  I/O BUFFER AREA
LOGBF6   EQU    *
         DC     CL48' '                   LOGICAL BUFFER AREA
SW1CH6   DC     XL1'00'                   RY1 SWITCH
LOG6     EQU    *                         *STATISTICS LOG AREA
         DC     ALZ(13)                   .
         DC     XL13'00'                  .
ADRLST   $VOLB  ID-01,TERMAD-E2E2,LEN-2   ADDRESSING LIST, FOR S/3 SS
LASTG    $CKL   DTF-DTF6,LAST-Y           CHECK LIST
         EJECT
```


IBM System/3 Basic Assembler Coding Form

SAMPLE MLMP: RECEIVE DTF1 (SEE PAGE 1)

PAGE 8 OF 16

```
RCV1     $GETB  DTF-DTF1,REJECT-ERR1A      GET AN 80 BYTE RECORD
CHK1     $CHK   CKL-CKLST1                 WAIT FOR COMPLETION
         CLI    $BDCMP(,$DTF),$BCDNE       GOOD COMPLETION?
         JNE    ERR1B                      NO. CHECK COMPLETION
*        PROCESS THE 80 BYTE ASCII RECORD MOVED TO LOGBF1
         B      RCV1                       GET ANOTHER RECORD
*
ERR1A    EQU    *                          REQUEST NOT ACCEPTED
*        DETERMINE WHY THE REQUEST WAS REJECTED AND
*        TAKE AN APPROPRIATE ACTION
*
ERR1B    EQU    *                          CHECK COMPLETION
         CLI    $BDCMP(,DTF),$BCEOT        END-OF-FILE?
         BNE    ERR1C                      NO. CHECK FOR OTHER COMPLETION
*        END-OF-FILE RECEIVED
*        DO ADDITIONAL PROCESSING AS REQUIRED
*
ERR1C    EQU    *
*        DETERMINE THE ERROR CONDITION AND
*        TAKE AN APPROPRIATE ACTION
*
         EJECT
```

IBM System/3 Basic Assembler Coding Form

PROGRAM: SAMPLE MLMP : TRANSMIT DTF2 (SEE PAGE 2) — PAGE 9 / 16

PROGRAMMER — DATE — PUNCHING INSTRUCTIONS — GRAPHIC PUNCH

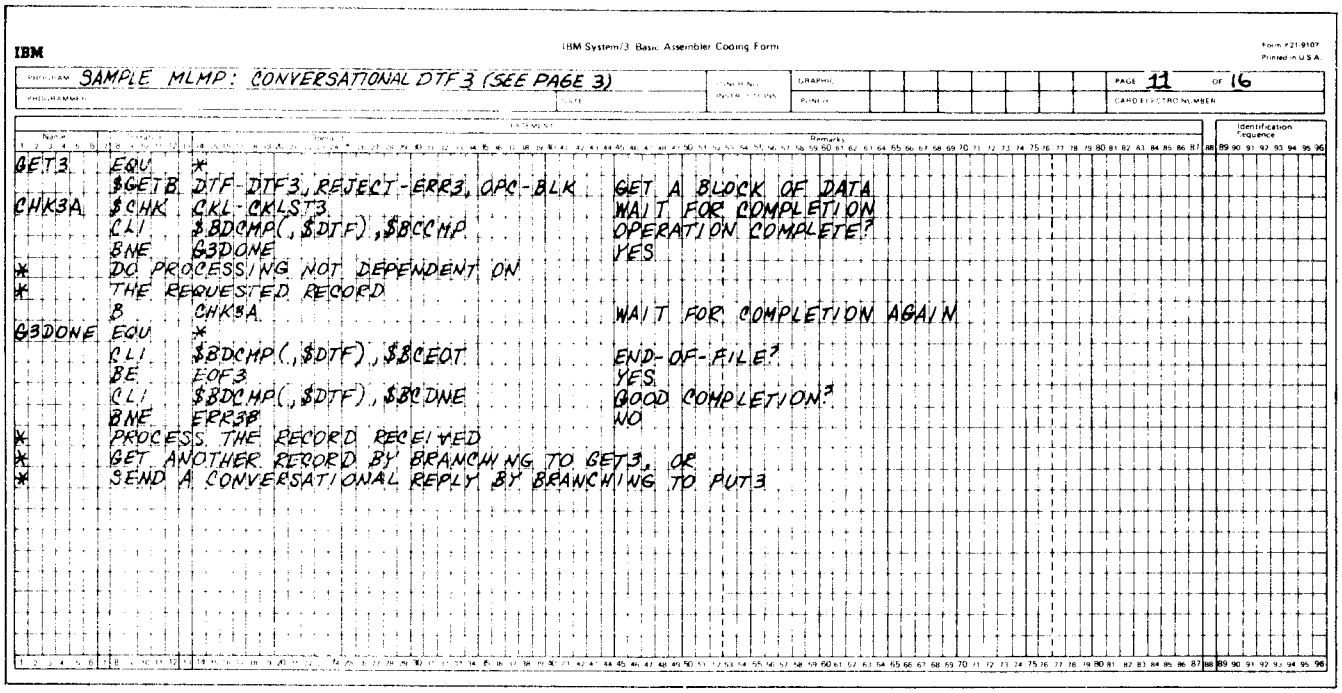
```
TSM2    EQU   *
*       PREPARE A 96 BYTE RECORD FOR TRANSMISSION AND MOVE TO LOGBF2
*       IF NO MORE RECORDS GO TO EOF2
          .
        $PUTB DTF-DTF2                    TRANSMIT A RECORD
        CLI   $BDCMP(,$DTF),$BCREQ        REQUEST ACCEPTED?
        BNE   ERR2A                       NO. CHECK COMPLETION
        $CHK  CKL-CKLST2                  WAIT FOR COMPLETION OF PUT
        CLI   $BDCMP(,$DTF),$BCDNE        GOOD COMPLETION?
        BNE   ERR2B                       NO. CHECK OTHER COMPLETIONS
        CLI   SWICH2,X'F0'                RVI RECEIVED?
        BNE   TSM2                        NO. SEND NEXT RECORD
*       PROCESS RVI RECEIVED
          .
EOF2    $PUTB DTF-DTF2,REJECT-ERR2A,OPC-EOF   SEND END-OF-FILE
        $CHK  CKL-CKLST2                  WAIT FOR COMPLETION
        CLI   $BDCMP(,$DTF),$BCDNE        GOOD COMPLETION?
        BNE   ERR2B                       NO.
*       CONTINUE PROCESSING
          .
ERR2A   EQU   *                           REQUEST REJECTED
*       DETERMINE WHY THE REQUEST WAS REJECTED AND
*       TAKE AN APPROPRIATE ACTION
          .
```

IBM System/3 Basic Assembler Coding Form

PROGRAM: SAMPLE MLMP; TRANSMIT DTF2 (CONT) — PAGE 10 / 16

PROGRAMMER — DATE — PUNCHING INSTRUCTIONS — GRAPHIC PUNCH

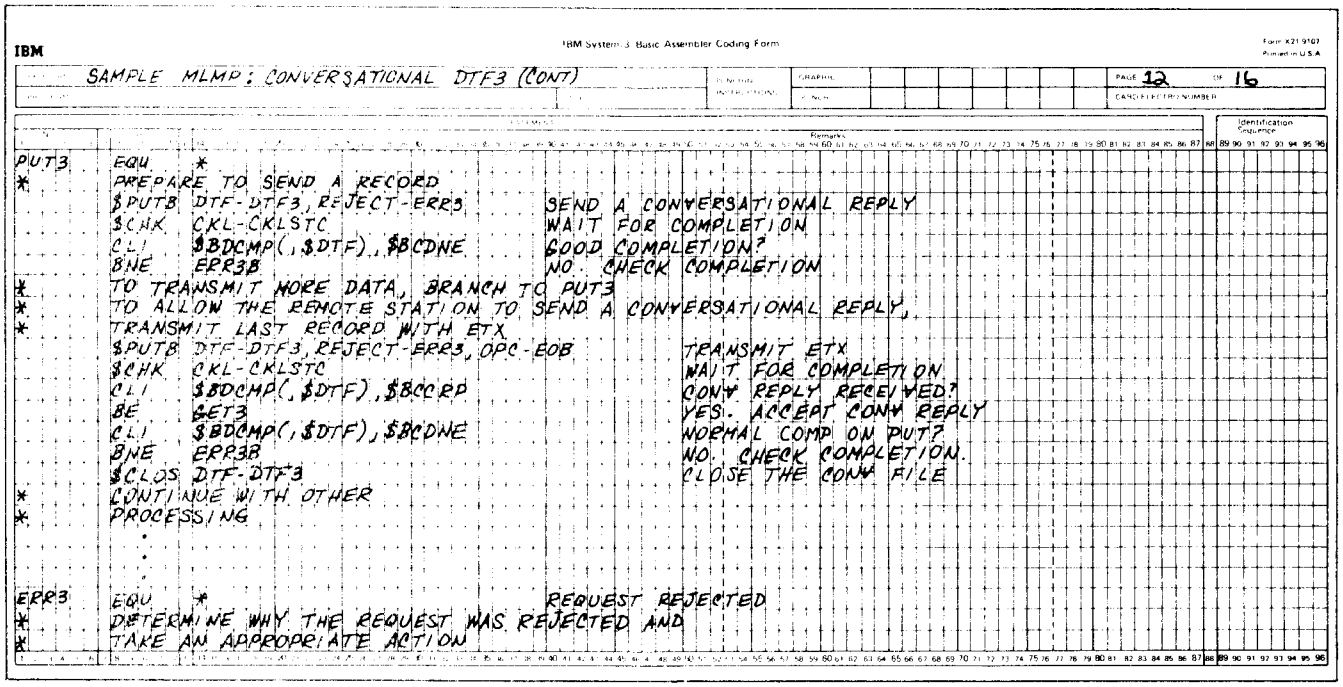
```
ERR2B   EQU   *
*       DETERMINE THE ERROR CONDITION AND
*       TAKE AN APPROPRIATE ACTION
          .
        EJECT
```



IBM System/3 Basic Assembler Coding Form — Form X21-9107 Printed in U.S.A.

PROGRAM: SAMPLE MLMP: CONVERSATIONAL DTF3 (SEE PAGE 3) — PAGE 11 OF 16

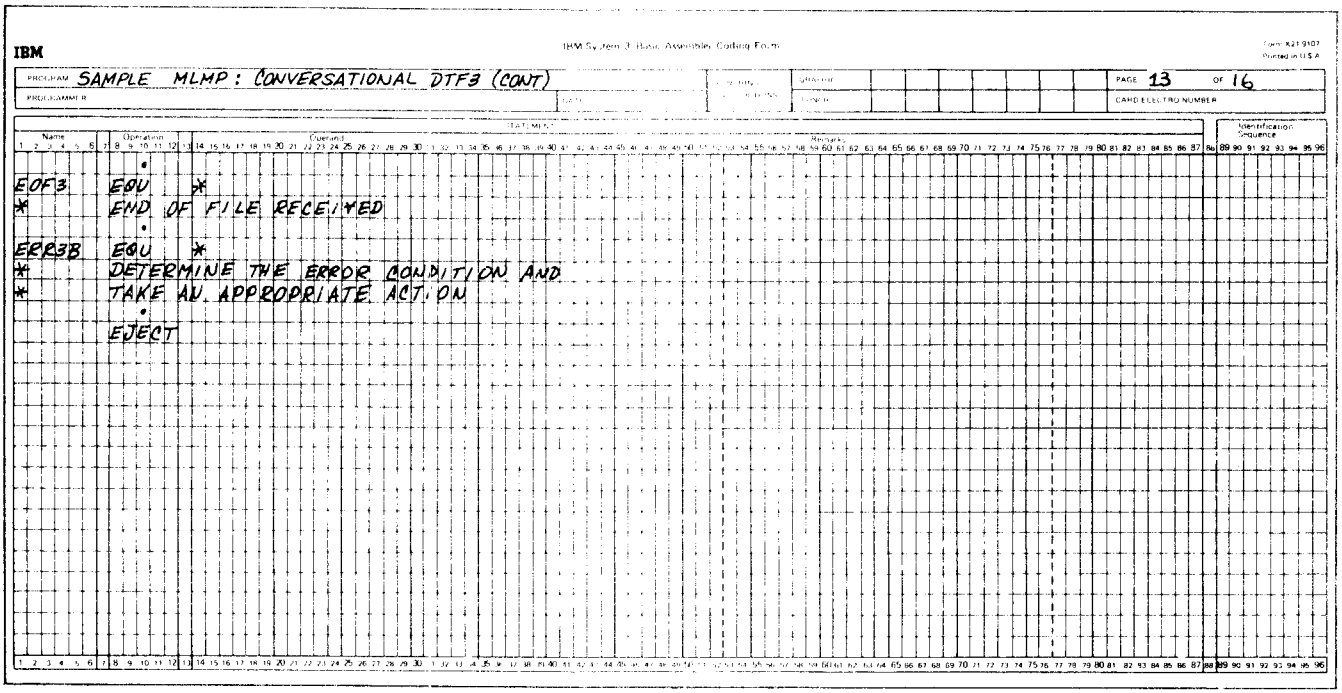
```
GET3    EQU   *
        $GETB DTF-DTF3,REJECT-ERR3,OPC-BLK    GET A BLOCK OF DATA
CHK3A   $CHK  CKL-CKLST3                      WAIT FOR COMPLETION
        CLI   $BDCMP(,$DTF),$BCCMP            OPERATION COMPLETE?
        BNE   G3DONE                          YES
*       DO PROCESSING NOT DEPENDENT ON
*       THE REQUESTED RECORD
        B     CHK3A                           WAIT FOR COMPLETION AGAIN
G3DONE  EQU   *
        CLI   $BDCMP(,$DTF),$BCEOT            END-OF-FILE?
        BE    EOF3                            YES
        CLI   $BDCMP(,$DTF),$BCDNE            GOOD COMPLETION?
        BNE   ERR3B                           NO
*       PROCESS THE RECORD RECEIVED
*       GET ANOTHER RERORD BY BRANCHING TO GET3, OR
*       SEND A CONVERSATIONAL REPLY BY BRANCHING TO PUT3
```



IBM System/3 Basic Assembler Coding Form — Form X21-9107 Printed in U.S.A.

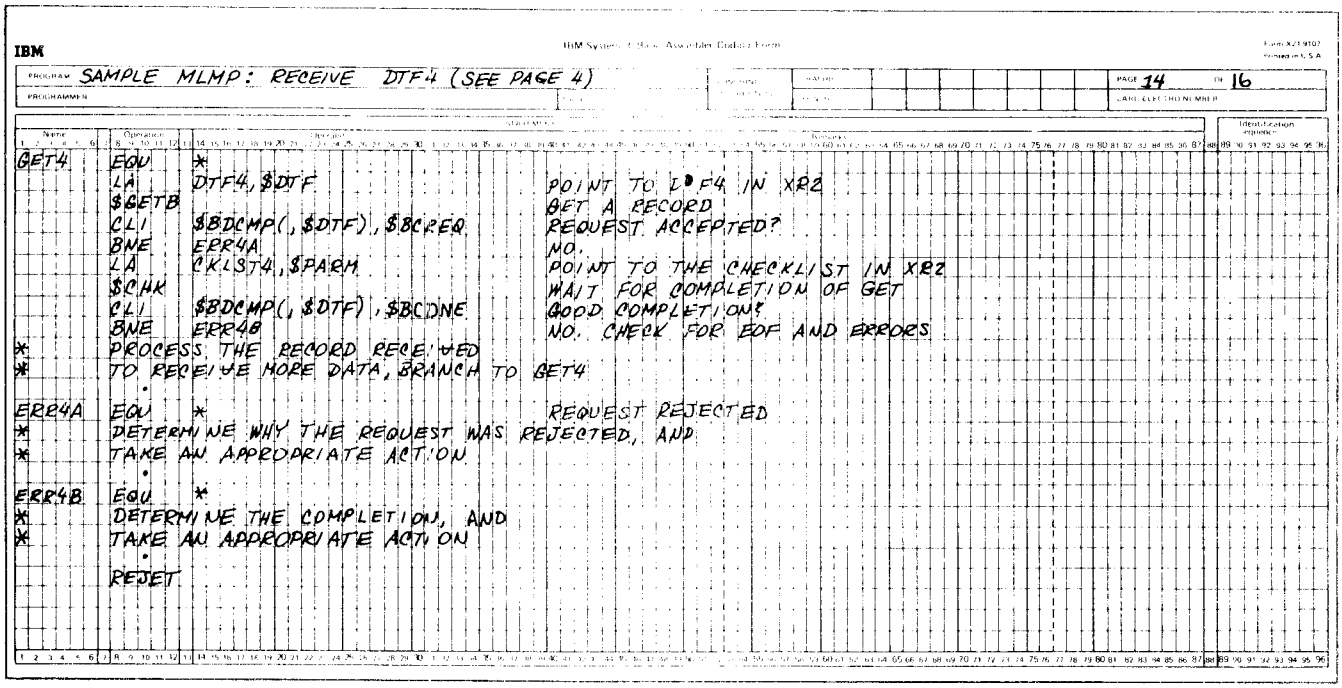
SAMPLE MLMP: CONVERSATIONAL DTF3 (CONT) — PAGE 12 OF 16

```
PUT3    EQU   *
*       PREPARE TO SEND A RECORD
        $PUTB DTF-DTF3,REJECT-ERR3            SEND A CONVERSATIONAL REPLY
        $CHK  CKL-CKLSTC                      WAIT FOR COMPLETION
        CLI   $BDCMP(,$DTF),$BCDNE            GOOD COMPLETION?
        BNE   ERR3B                           NO. CHECK COMPLETION
*       TO TRANSMIT MORE DATA, BRANCH TO PUT3
*       TO ALLOW THE REMOTE STATION TO SEND A CONVERSATIONAL REPLY,
*       TRANSMIT LAST RECORD WITH ETX
        $PUTB DTF-DTF3,REJECT-ERR3,OPC-EOB    TRANSMIT ETX
        $CHK  CKL-CKLSTC                      WAIT FOR COMPLETION
        CLI   $BDCMP(,$DTF),$BCCRP            CONV REPLY RECEIVED?
        BE    GET3                            YES. ACCEPT CONV REPLY
        CLI   $BDCMP(,$DTF),$BCDNE            NORMAL COMP ON PUT?
        BNE   ERR3B                           NO. CHECK COMPLETION.
        $CLOS DTF-DTF3                        CLOSE THE CONV FILE
*       CONTINUE WITH OTHER
*       PROCESSING
          .
          .
          .
ERR3    EQU   *                               REQUEST REJECTED
*       DETERMINE WHY THE REQUEST WAS REJECTED AND
*       TAKE AN APPROPRIATE ACTION
```



```
PROGRAM SAMPLE MLMP: CONVERSATIONAL DTF3 (CONT)                PAGE 13 OF 16

EOF3     EQU   *
*        END OF FILE RECEIVED
         .
ERR3B    EQU   *
*        DETERMINE THE ERROR CONDITION AND
*        TAKE AN APPROPRIATE ACTION
         .
         EJECT
```



```
PROGRAM SAMPLE MLMP: RECEIVE DTF4 (SEE PAGE 4)                 PAGE 14 OF 16

GET4     EQU   *
         LA    DTF4,$DTF                  POINT TO DTF4 IN XR2
         $GETB                            GET A RECORD
         CLI   $BDCMP(,$DTF),$BCREQ       REQUEST ACCEPTED?
         BNE   ERR4A                      NO.
         LA    CKLST4,$PARM               POINT TO THE CHECKLIST IN XR2
         $CHK                             WAIT FOR COMPLETION OF GET
         CLI   $BDCMP(,$DTF),$BCDNE       GOOD COMPLETION?
         BNE   ERR4B                      NO. CHECK FOR EOF AND ERRORS
*        PROCESS THE RECORD RECEIVED
*        TO RECEIVE MORE DATA, BRANCH TO GET4
         .
ERR4A    EQU   *                          REQUEST REJECTED
*        DETERMINE WHY THE REQUEST WAS REJECTED, AND
*        TAKE AN APPROPRIATE ACTION
         .
ERR4B    EQU   *
*        DETERMINE THE COMPLETION, AND
*        TAKE AN APPROPRIATE ACTION
         .
         REJET
```

IBM System/3 Basic Assembler Coding Form

PROGRAM: SAMPLE MLMP: RECEIVE DTF5 (SEE PAGE 5) — PAGE 15 OF 16

```
GET5     EQU   *
         $GETB DTF-DTF5,REJECT-ERR5,OPC-BLK   GET A BLOCK OF DATA
         $CHK  CKL-CKLST5                     WAIT FOR COMPLETION
         CLI   $BDCMP(,$DTF),$BCDNE           GOOD COMPLETION?
         BNE   ERR5B                          NO. CHECK FOR EOP AND ERRS
*        PROCESS THE BLOCK OF DATA RECEIVED
*        TO RECEIVE MORE DATA, BRANCH TO GET5
*        TO SEND RVI
         MVI   SWICH5,X'01'                   SET ON RVI MSK TO SEND RVI
         B     GET5                           GET NEXT BLOCK
         .
ERR5     EQU   *                              REQUEST REJECTED
*        DETERMINE WHY THE REQUEST WAS REJECTED, AND
*        TAKE AN APPROPRIATE ACTION
         .
ERR5B    EQU   *
         MVI   SWICH5,X'00'                   SET RVI MSK OFF
*        DETERMINE THE COMPLETION, AND
*        TAKE AN APPROPRIATE ACTION
         .
         EJECT
```


IBM System/3 Basic Assembler Coding Form

PROGRAM: SAMPLE MLMP: TRANSMIT DTF6 (SEE PAGE 6) — PAGE 16 OF 16

```
PUT6     EQU   *
*        PREPARE A 48 BYTE RECORD FOR TRANSMISSION AND MOVE
*        TO LOGBF6
SEND6    $PUTB DTF-DTF6,REJECT-ERR6,OPC-EOM   TRANSMIT A RECORD
         $CHK  CKL-CKLST6                     WAIT FOR COMPLETION
         CLI   $BDCMP(,$DTF),$BCDNE           GOOD COMPLETION?
         BE    CONTNU                         YES. CONTINUE
         CLI   $BDCMP(,$DTF),$BCNEG           NEGATIVE RESPONSE?
         BNE   ERR6B                          NO. CHECK COMPLETION
         TBN   SWICH6,X'FF'                   RVI RECEIVED?
         SBF   SWICH6,X'FF'                   SET RVI MSK OFF
         BT    GET6                           YES. POLL TO RECEIVE DATA
         B     SEND6                          NO. CONTINUE POLLING
         .
ERR6     EQU   *
*        DETERMINE WHY THE REQUEST WAS REJECTED, AND
*        TAKE AN APPROPRIATE ACTION
         .
ERR6B    EQU   *
*        DETERMINE THE ERROR CONDITION, AND
*        TAKE AN APPROPRIATE ACTION
         END
```

Model 10 and Model 12 Sample Program: Communicating with the 3270

```
S3270                          EXTERNAL SYMBOL LIST

SYMBOL    TYPE                                          VER 14, MOD 00  08/26/77  PAGE    1

S3270     MODULE
$$BSMS    EXTRN
$$BSTT    EXTRN
$$BMCH    EXTRN
$$BSLG    EXTRN



S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE    2

    0000                        2 S3270  START 0
                          0001  3        EXTRN $$BSMS
                          0002  4        EXTRN $$BSTT


                                6 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                                7 *                                                                             *
                                8 *                                                                             *
                                9 *   S3270 IS A SAMPLE MLMP PROGRAM WRITTEN TO ILLUSTRATE THE                  *
                               10 *   FOLLOWING -                                                               *
                               11 *                                                                             *
                               12 *         1. HOW TO CODE SYSTEM/3 SYSTEM MACROS.                              *
                               13 *         2. HOW TO CODE SYSTEM/3 MLMP MACROS.                                *
                               14 *         3. HOW TO READ AND WRITE WITH A 3270 INFORMATIONAL DISPLAY          *
                               15 *         SYSTEM.                                                             *
                               16 *                                                                             *
                               17 *   S3270 WILL DISPLAY A NAME AND ADDRESS MESSAGE ON THE SCREEN. THE          *
                               18 *   OPERATOR THEN MAY KEY IN THE NAME AND ADDRESS FIELDS. THE ENTER           *
                               19 *   KEY SHOULD BE DEPRESSED TO INDICATE THE END OF THE FIELDS.                *
                               20 *   THE DATA FIELDS ENTERED BY THE OPERATOR VIA THE DISPLAY KEYBOARD          *
                               21 *   WILL BE PRINTED ON THE SYSTEM LOGGING DEVICE. THE CLEAR KEY OR THE        *
                               22 *   ONLINE  TEST REQUEST KEY MAY ALSO BE USED. PRESSING THE ENTER KEY         *
                               23 *   WITHOUT ANY DATA WILL CAUSE END OF JOB. USE OF ANY OTHER KEYS             *
                               24 *   WILL CAUSE AN ERROR MESSAGE TO BE PRINTED AND THE SCREEN TO BE            *
                               25 *   REFRESHED WITH THE NAME AND ADDRESS DISPLAY.                              *
                               26 *                                                                             *
                               27 *                                                                             *
                               28 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE   3

                                30 *       $ALOC DTF-DTF1                    ALLOCATE AND OPEN THE DTFS
                                31+*     DEVICE ALLOCATE LINKAGE             RELEASE-7
    0000 C2 02 01DF             32+        LA     DTF1,$DTF                  SET ADDRESS OF DTF IN REGISTER 2
    0004 C0 87 0004             33+        B      4                          BRANCH TO GENERAL ENTRY
    0008 88                0008 34+        DC     XL1'88'                    RIB - ALLOCATE SPACE
                                35 *       $OPEN DTF-DTF1
                                36+*     DEVICE OPEN LINKAGE                 RELEASE-7
    0009 C2 02 01DF             37+        LA     DTF1,$DTF                  LOAD DTF ADDRESS IN REGISTER 2
    000D C0 87 0004             38+        B      4                          GO TO GENERAL ENTRY
    0011 82                0011 39+        DC     XL1'82'                    RIB - OPEN
                                40 *
                                41 *  THIS ROUTINE WILL SET THE DISPLAY SCREEN TO NULLS,  MOVE THE NAME  *
                                42 *   AND ADDRESS MESSAGES TO THE SCREEN DISPLAY AND SET THE CURSOR AND *
                                43 *   BUFFER ADDRESS POINTERS TO THE FIRST OPERATOR ENTERED FIELD.      *
                                44 *
                           0012 45 SETDIS EQU   *
                                46 *       $PUTB DTF-DTF1,REJECT-BADRET
                                47+*       BSCA PUT LINKAGE                  RELEASE-7  D

    0012 C2 02 01DF             49+        LA     DTF1,$DTF                  LOAD XR2 WITH DTF ADDR.
    0016 BD 00 0E               50+        CLI    $BDCMP(,$DTF),$BCREQ       LAST OP DONE ?
    0019 F2 81 30               51+        JE     $E1003                     NO-GO POST REQUEST IGNORED.
    001C B8 01 03               52+        TBN    $BDATR(,$DTF),$BCOPN       OPENED ?
    001F B8 40 02               53+        TBN    $BDATT(,$DTF),$BCOUT       PUT FILE ?
    0022 F2 90 2D               54+        JE     $E2003                     NO-GO POST PERM'T ERROR.
    0025 B8 C0 02               55+        TBN    $BDATT(,$DTF),$BCCNV       CONVERSATIONAL
    0028 BD 46 0E               56+        CLI    $BDCMP(,$DTF),$BCCRP       REPLY PENDING ?
    002B F2 16 2A               57+        JC     $E3003,$TRU+$HI+$LO        YES-GO POST INVALID CALL.
    002E BC 40 0E               58+        MVI    $BDOPC(,$DTF),$BOPUT       SET REQUESTED PUT OPERATION.
    0031 BC 00 0E               59+        MVI    $BDCMP(,$DTF),$BCREQ       SET TO OP ACCEPTED.
    0034 C0 87 0001             60+        B      $$BSMS                     GO TO BSCA DATA MANAGEMENT.
    0038 34 01 0045             61+        ST     $S1003+3,$BBAC1            SAVE XR1.
    003C B5 01 23               62+        L      $BDIOB(,$DTF),$IOB         LOAD IOB ADDR IN XR1.
    003F 70 4D 07               63+        CLI    $BICMP(,$IOB),$BCCAL       Q INVALID CALL POSTED IN IOB?
    0042 C2 01 0000          64+$S1003 LA     *-*,$BBAC1                 RESTORE XR1.
    0046 F2 81 0F               65+        JE     $E3003                     YES-GO TO POST INVALID REQUEST.
    0049 F2 87 13               66+        J      $XT003                     GO EXIT.
    004C BC 4A 0E               67+$E1003  MVI    $BDCMP(,$DTF),$BCIGN       SET REQUEST IGNORED.
    004F F2 87 09               68+        J      $RJ003                     GO EXIT.
    0052 BC 41 0E               69+$E2003  MVI    $BDCMP(,$DTF),$BCUER       SET USER ERROR.
    0055 F2 87 03               70+        J      $RJ003                     GO EXIT.
    0058 BC 4D 0E               71+$E3003  MVI    $BDCMP(,$DTF),$BCCAL       SET INVALID CALL.
                           005B 72+$RJ003  EQU    *
    005B C0 87 019F             73+        B      BADRET                     GO HANDLE REJECTED COMMAND.
                           005F 74+$XT003  EQU    *
                                75 *       $CHK   CKL-LIST1
                                76+*     GENERATE A WAIT ON LIST CALL        RELEASE-7  A
    005F C2 02 061F             77+        LA     LIST1,$PARM                LOAD ADDR OF WAIT LIST IN XR2.
    0063 C0 87 0003             78+        B      $$BMCH                     CALL COMMON WAIT.

                           0003 80+        EXTRN  $$BMCH                     EXTRN FOR $$BMCH.

    0067 BD 44 0E               82         CLI    $BDCMP(,$DTF),$BCNEG       NEGATIVE RESPONSE ?
    006A F2 01 0B               83         JNE    CKGOOD                     JUMP IF NOT NEGATIVE RESPONSE
    006D 38 80 071B             84         TBN    MASK,RVI                   RVI PENDING ?
    0071 3B 80 071B             85         SBF    MASK,RVI                   SET OFF RVI INDICATOR
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE   4

     0075 F2 10 07                 86         JT    RESPON                      GET STATUS IF RVI RESPONSE
                           0078    87 CKGOOD  EQU   *
     0078 BD 40 0F                 88         CLI   $BDCMP(,$DTF),$BCDNE        GOOD OP COMPLETION ?
     007B C0 01 019F               89         BNE   BADRET                      GO TO ERROR PRINT IF NOT GOOD
                                   90 *
                                   91 *   THIS ROUTINE WILL POLL THE 3270 SYSTEM FOR THE OPERATOR RESPONSE. *
                                   92 *   THE DATA FIELD RECEIVED FROM THE TERMINAL IS ANALYZED FOR THE     *
                                   93 *   CLEAR KEY, STATUS MESSAGE, OR ENTER WITH OR WITHOUT DATA. ANY     *
                                   94 *   OTHER KEY WILL CAUSE AN ERROR MESSAGE. IF ENTER IS PRESENT WITH   *
                                   95 *   NO DATA THE END OF JOB ROUTINE IS CALLED. THE STATUS MESSAGE WILL *
                                   96 *   BE PRINTED OUT. THE OPERATOR KEYED DATA FIELDS WILL BE PRINTED.   *
                                   97 *
                           007F    98 RESPON  EQU   *
                           007F    99 MORE    EQU   *
                                  100 *       $GETB DTF-DTF2,OPC-BLK,REJECT-BADRET
                                  101**       BSCA GET LINKAGE                  RELEASE-7  D

     007F C2 02 021D              103+        LA    DTF2,$DTF                   LOAD XR2 WITH DTF ADDR.
     0083 BD 00 0E                104+        CLI   $BDCMP(,$DTF),$BCREQ       LAST OP DONE ?
     0086 F2 81 41                105+        JE    $E1005                      NO-GO POST REQUEST IGNORED.
     0089 B8 01 03                106+        TBN   $BDATR(,$DTF),$BCOPN       OPENED ?
     008C B8 80 02                107+        TBN   $BDATT(,$DTF),$BCINP       GET FILE ?
     008F F2 90 3E                108+        JE    $E2005                      NO-GO POST PERMIT ERROR.
     0092 34 01 009F              109+        ST    $SV005+3,$BBAC1             SAVE REGISTER.
     0096 B5 01 23                110+        L     $BDIOB(,$DTF),$IOB          LOAD THE IOB REGISTER.
     0099 79 04 05                111+        TBF   $BIFLA(,$IOB),$BIEST        ERROR FREE, ----------------|
     009C C2 01 0000              112+$SV005  LA    *-*,$BBAC1                  RELOAD REGISTER.            |
     00A0 B8 C0 02                113+        TBN   $BDATT(,$DTF),$BCCNV        AND CONVERSATIONAL ? <------|
     00A3 F2 90 09                114+        JE    $GT005                      NO-GO PROCESS THE GET.
     00A6 B8 40 0E                115+        TBN   $BDOPC(,$DTF),$BOPUT        LAST OPERATION A PUT,
     00A9 BD 46 0E                116+        CLI   $BDCMP(,$DTF),$BCCRP        AND NO CONV. REPLY PENDING ?
     00AC F2 11 27                117+        JC    $E3005,$TRU+$EQ             YES-GO POST INVALID CALL.
     00AF BC 81 0E                118+$GT005  MVI   $BDOPC(,$DTF),$BOGBK        SET OP CODE FOR GET FUNCTION.
     00B2 C0 87 0001              119+        B     $$BSMS                      GO TO BSCA DATA MANAG MENT.
     00B6 34 01 00C3              120+        ST    $S1005+3,$BBAC1             SAVE XR1.
     00BA B5 01 23                121+        L     $BDIOB(,$DTF),$IOB          LOAD IOB ADDR IN SR1.
     00BD 7D 4D 07                122+        CLI   $BICMP(,$IOB),$BCCAL        Q INVALID CALL POSTED IN IOB?
     00C0 C2 01 0000              123+$S1005  LA    *-*,$BBAC1                  RESTORE XR1.
     00C4 F2 81 0E                124+        JE    $E3005                      YES-GO TO SET INVALID REQUEST.
     00C7 F2 87 13                125+        J     $XT005                      GO EXIT.
     00CA BC 4A 0E                126+$E1005  MVI   $BDCMP(,$DTF),$BCIGN        SET REQUEST IGNORED.
     00CD F2 87 09                127+        J     $RJ005                      GO EXIT.
     00D0 BC 41 0E                128+$E2005  MVI   $BDCMP(,$DTF),$BCUER        SET USER ERROR.
     00D3 F2 87 03                129+        J     $RJ005                      GO EXIT.
     00D6 BC 4D 0E                130+$E3005  MVI   $BDCMP(,$DTF),$BCCAL        SET INVALID REQUEST.
                           00D9   131+$RJ005  EQU   *
     00D9 C0 87 C19F              132+        B     BADRET                      GO HANDLE REJECTED COMMAND.
                           00DD   133+$XT005  EQU   *
                                  134 *       $CHK  CKL-LIST2
                                  135+*      GENERATE A WAIT ON LIST CALL       RELEASE-7  A
     00DD C2 02 0622              136+        LA    LIST2,$PARM                 LOAD ADDR OF WAIT LIST IN XR2.
     00E1 C0 87 00C3              137+        B     $$BMCH                      CALL COMMON WAIT.

     00E5 BD 42 0F                139         CLI   $BDCMP(,$DTF),$BCEOT        END OF FILE RECEIVED ?
     00E8 C0 81 0012              140         BE    SETDIS                      GO TO PUT IF EOF RETURN
     00EC BD 40 0F                141         CLI   $BDCMP(,$DTF),$BCDNE        GOOD OP COMPLETION ?
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE   5

     00EF C0 01 019F              142         BNE   BADRET                      PRINT BAD RETURN IF NOT EOF
     00F3 3D 6D 04AF              143         CLI   WORK2+3,CLEAR               WAS CLEAR KEY USED ?
     00F7 C0 81 007F              144         BE    MORE                        LOOP TO GET END OF FILE
     00FB 0D 02 04AD 0627         145         CLC   WORK2+2(3),STATUS           STATUS MESSAGE RECEIVED ?
     0101 C0 81 011A              146         BE    PSTAT                       GO TO PRINT THE STATUS MESSAGE
     0105 3D 7D 04AE              147         CLI   WORK2+3,ENTER               ENTER KEY USED ?
     0109 C0 81 0175              148         BE    DATA                        CHECK FOR DATA IF ENTER
     010D C2 02 05E6              149         LA    ERROR,XR2                   SET ERROR PARM POINTER
                                  150 *       $SVC  RIB-HALT
                                  151+*     SUPERVISOR CALL LINKAGE             RELEASE-7
     0111 C0 87 0004              152+        B     4                           BRANCH TO GENERAL ENTRY
     0115 85                0115  153+        DC    XL1'85'                     RIB - 85

     0116 C0 87 007F              155         B     MORE                        LOOP BACK TO GET EOF
```

```
S3270  ALMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE   6

                                157 *
                                158 *   THIS ROUTINE WILL PRINT THE STATUS MESSAGES
                                159 *
                          011A  160 PSTAT   EQU   *
    011A 3C F0 071A             161         MVI   STAT,ZERO
    011E 0C 10 0719 071A        162         MVC   STAT-1(17),STAT          INITIALIZE THE PRINT AREA
    0124 3C 09 06E8             163         MVI   SCOUNT,NINE              INITIALIZE MOVE COUNTER
    0128 C2 01 0709             164         LA    STATB,XR1                SET PRINT POINTER
    012C C2 02 04AB             165         LA    WORK2,XR2                SET DATA POINTER
                          0130  166 MOVST   EQU   *
    0130 68 02 00 00            167         MNZ   0(,XR1),0(,XR2)          SAVE THE STATUS BYTES
    0134 68 03 01 00            168         MNN   1(,XR1),0(,XR2)
    0138 C2 01 02               169         LA    2(,XR1),XR1              UPDATE THE PRINT AREA POINTER
    013B E2 02 01               170         LA    1(,XR2),XR2              UPDATE THE STATUS POINTER
    013E CF 00 06E8 06E9        171         SLC   SCOUNT(1),ONE            UPDATE THE MOVE COUNTER
    0144 C0 01 0130             172         BNZ   MOVST                    LOOP TO MOVE IF NOT DONE
    0148 C2 01 0709             173         LA    STATB,XR1                RESET STATUS MESSAGE POINTER
    014C 3C 12 06F8             174         MVI   SCOUNT,EIGTEN            RESET PROCESS COUNT
                          0150  175 SETPRT  EQU   *
    0150 7D F9 00               176         CLI   0(,XR1),NONPRT           PRINTABLE CHARACTER ?
    0153 F2 04 C5               177         JNH   NOSB39                   JUMP IF PRINTABLE
    0156 4F 00 00 06FA          178         SLC   0(1,XR1),SB39            SET TO PRINTABLE
                          0158  179 NOSB39  EQU   *
    0158 C2 01 01               180         LA    1(,XR1),XR1              UPDATE THE PRINT AREA POINTER
    015E CF 00 06E8 06E9        181         SLC   SCOUNT(1),ONE            UPDATE PROCESS COUNTER
    0164 C0 01 0150             182         BNZ   SETPRT                   LOOP IF NOT END OF PROCESS
    0168 C2 02 05EB             183         LA    SMESSG,XR2               SET PRINT PARM POINTER
                                184 *       $SVC  RIB-HALT
                                185+*     SUPERVISOR CALL LINKAGE          RELEASE-7
    016C C0 87 0004             186+        B     4                        BRANCH TO GENERAL ENTRY
    0170 85                     187+        DC    XL1'85'                  RIB - 85

    0171 C0 87 007F             189         B     MORE                     GO TO GET EOF
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE   7

                                191 *
                                192 *   THIS ROUTINE IS USED TO PROCESS THE DATA MESSAGES
                                193 *
                          0175  194 DATA   EQU   *
    0175 3D 03 04B1             195        CLI   WORK2+6,ETX            ENTER KEY ONLY ?
    0179 C0 81 01D1             196        BE    CLOSE                  GO TO CLOSE IF ENTER ONLY
    017D 0C 1D 06B7 04D6        197        MVC   NEND(30),NFIELD        MOVE NAME DATA TO PRINT
    0183 C2 02 05DC             198        LA    NAME,XR2               SET NAME PRINT PARM POINTER
                                199 *      $SVC  RIB-HALT
                                200+*    SUPERVISOR CALL LINKAGE        RELEASE-7
    0187 C0 87 0004             201+       B     4                      BRANCH TO GENERAL ENTRY
    018B 85                     202+       DC    XL1'85'                RIB - 85

    018C 0C 1D 06E7 04F2        204        MVC   AEND(30),AFIELD        MOVE ADDRESS TO PRINT AREA
    0192 C2 02 05E1             205        LA    ADDR,XR2               SET ADDRESS PARM POINTER
                                206 *      $SVC  RIB-HALT
                                207+*    SUPERVISOR CALL LINKAGE        RELEASE-7
    0196 C0 87 0004             208+       B     4                      BRANCH TO GENERAL ENTRY
    019A 85                     209+       DC    XL1'85'                RIB - 85

    019B C0 87 007F             211        B     MORE                   LOOP TO RECEIVE EOF
                                212 *
                                213 *   THIS ROUTINE CHECKS THE COMPLETION CODE FOR THE GET REQUEST
                                214 *
                          019F  215 BADRET EQU   *
    019F 07 01 0647 0647        216        SZ    RET(2),RET(2)          INITIALIZE PRINT AREA
    01A5 28 02 0646 0E          217        MNZ   RET-1,$BDCMP(,$DTF)    SAVE THE RETURN CODE
    01AA 28 03 0647 0E          218        MNN   RET,$BDCMP(,$DTF)
    01AF 3D F9 0647             219        CLI   RET,NONPRT             PRINTABLE CHARACTER ?
    01B3 F2 04 06               220        JNH   NOSET                  JUMP IF PRINTABLE ?
    01B6 CF 00 0647 06EA        221        SLC   RET(1),SB39            SET TO PRINTABLE CHARACTER
                          01BC  222 NOSET  EQU   *
    01BC C2 02 05D7             223        LA    RETURN,XR2             SET PRINT PARM POINTER
                                224 *      $SVC  RIB-HALT
                                225+*    SUPERVISOR CALL LINKAGE        RELEASE-7
    01C0 C0 87 0004             226+       B     4                      BRANCH TO GENERAL ENTRY
    01C4 85                     227+       DC    XL1'85'                RIB - 85

    01C5 C2 02 05F0             229        LA    HLMESS,XR2             SET HALT PARM POINTER
                                230 *      $SVC  RIB-HALT
                                231+*    SUPERVISOR CALL LINKAGE        RELEASE-7
    01C9 C0 87 0004             232+       B     4                      BRANCH TO GENERAL ENTRY
    01CD 85                     233+       DC    XL1'85'                RIB - 85

    01CE F0 FF FF               235        HPL   X'FF',X'FF'            HALT TO ALLOW CORE DUMP
                          01D1  236 CLOSE  EQU   *
                                237 *      $CLOS DTF-DTF1               CLOSE THE BSCA FILES
                                238+*    DEVICE CLOSE LINKAGE           RELEASE-7
    01D1 C2 02 01DF             239+       LA    DTF1,$DTF              SET ADDR OF DTF IN REGISTER 2.
    01D5 C0 87 0004             240+       B     4                      BRANCH TO GENERAL ENTRY
    01D9 83                     241+       DC    XL1'83'                RIB - CLOSE
                                242 *      $EOJ                         CALL EOJ ROUTINES
                                243+*     END OF JOB LINKAGE            RELEASE-7
    01DA C0 87 0004             244+       B     4                      BRANCH TO GENERAL ENTRY
    01DE 84                     245+       DC    XL1'84'                END OF JOB
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE   8

                          247 *TF1    $DTFB RECL-104,BLKL-104,BUFST-BUF1,BUFEND-B1END,RCAD-WORK1,      X
                          248 *               FTYP-TSM,LINE-1,CODE-E,TYPE-CS,LISTAD-SELECT,ERRLOG-LOG,  X
                          249 *               CHN-DTF2,RVIADR-MASK,RVIMSK-80,LIMIT-10
                          250+*       BSCA DTF.                          RELEASE-8
                    0004  251+        EXTRN $$BSLG                       EXTRN FOR TERM LOG ROUTINE.
                    01DF  252+DTF1    EQU   *
    01DF 80         01DF  253+        DC    XL1'80'                      DEVICE ID.
    01E0 00         01E0  254+        DC    XL1'00'                      DEFAULT UPSI OF ZERO.
    01E1 41         01E1  255+        DC    AL1(00+00+00+00+9*00+65)     ATTR BYTE 1.
    01E2 88         01E2  256+        DC    XL1'88'                      ATTR BYTE 2.
    01E3 0000       01E4  257+        DC    XL2'00'                      POST OPEN DTF CHAIN PTR.
    01E5 C21D       01E6  258+        DC    AL2(DTF2)                    DTF CHAIN PTR.
    01E7 C0000000   01EA  259+        DC    XL4'00'                      SYSTEM SAVE AREA.
    01EB 0443       01EC  260+        DC    AL2(WORK1)                   ADDR OF USER LOGICAL BUFFER.
    01ED 00         01ED  261+        DC    XL1'00'                      COMPLETION CODE.
    01EE 00         01EE  262+        DC    XL1'00'                      OPERATION CODE.
    01EF 0000       01F0  263+        DC    XL2'00'                      BSCA WORK AREA.
    01F1 00         01F1  264+        DC    XL1'00'                      INDICATORS.
    01F2 060F       01F3  265+        DC    AL2(SELECT)                  ADDR OF POLL/ADDR LIST.
    01F4 F1         01F4  266+        DC    XL1'F1'                      POLL/ADDR ID WHEN CTRL STAT.
    01F5 C0         01F5  267+        DC    XL1'00'                      RESERVED.
    01F6 CA         01F6  268+        DC    AL1(10)                      WRAP LIST COUNT.
    01F7 00         01F7  269+        DC    XL1'00'                      RESERVED.
    01F8 000000     01FA  270+        DC    XL3'00'                      RESERVED.
    01FB 00B4       01FC  271+        DC    XL2'00B4'                    DELAY COUNT.
    01FD 0068       01FE  272+        DC    AL2(104)                     RECORD LENGTH.
    01FF C068       0200  273+        DC    AL2(104)                     BLOCK LENGTH.
    0201 025B       0202  274+        DC    AL2(BUF1)                    ADDR OF START OF I/O AREA.
    0203 02EC       0204  275+        DC    AL2(B1END)                   ADDR OF END OF I/O AREA.
    0205 0000000000 0209  276+        DC    XL5'00'                      BSCA WORK AREA.
    020A 80         020A  277+        DC    XL1'80'                      RVI MASK.
    020B 071B       020C  278+        DC    AL2(MASK)                    RVI MASK ADDR.
    020D 000000000000 0212 279+       DC    XL6'00'                      BSCA WORK AREA.
    0213 08         0213  280+        DC    AL1(C8+00+00)                TERM ATTR.
    0214 C0         0214  281+        DC    XL1'00'                      RESERVED.
    0215 000000000000 021A 282+       DC    XL6'00'                      BSCA WORK AREA.
    021B 05F5       021C  283+        DC    AL2(LOG)                     ADDR OF TERM LOG AREA.


S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE   9

                          285 *TF2    $DTFB RECL-300,BLKL-300,BUFST-BUF2,BUFEND-B2END,RCAD-WORK2,      X
                          286 *               FTYP-RCV,LINE-1,CODE-E,TYPE-CS,LISTAD-POLL,ERRLOG-LOG
                          287+*       BSCA DTF.                          RELEASE-8
                    021D  288+DTF2    EQU   *
    021D 80         021D  289+        DC    XL1'80'                      DEVICE ID.
    021E 00         021E  290+        DC    XL1'00'                      DEFAULT UPSI OF ZERO.
    021F 89         021F  291+        DC    AL1(00+00+00+0C+9*08+65)     ATTR BYTE 1.
    0220 88         0220  292+        DC    XL1'88'                      ATTR BYTE 2.
    0221 C000       0222  293+        DC    XL2'00'                      POST OPEN DTF CHAIN PTR.
    0223 FFFF       0224  294+        DC    XL2'FFFF'                    DTF CHAIN PTR.
    0225 C0000000   0228  295+        DC    XL4'00'                      SYSTEM SAVE AREA.
    0229 C4AB       022A  296+        DC    AL2(WORK2)                   ADDR OF USER LOGICAL BUFFER.
    022B 00         022B  297+        DC    XL1'00'                      COMPLETION CODE.
    022C C0         022C  298+        DC    XL1'00'                      OPERATION CODE.
    022D 0000       022E  299+        DC    XL2'00'                      BSCA WORK AREA.
    022F 00         022F  300+        DC    XL1'00'                      INDICATORS.
    0230 0617       0231  301+        DC    AL2(POLL)                    ADDR OF POLL/ADDR LIST.
    0232 F1         0232  302+        DC    XL1'F1'                      POLL/ADDR ID WHEN CTRL STAT.
    0233 C0         0233  303+        DC    XL1'00'                      RESERVED.
    0234 FF         0234  304+        DC    XL1'FF'                      DEFAULT WRAP LIST COUNT.
    0235 00         0235  305+        DC    XL1'00'                      RESERVED.
    0236 000000     0238  306+        DC    XL3'00'                      RESERVED.
    0239 00B4       023A  307+        DC    XL2'00B4'                    DELAY COUNT.
    023B 012C       023C  308+        DC    AL2(300)                     RECORD LENGTH.
    023D C12C       023E  309+        DC    AL2(300)                     BLOCK LENGTH.
    023F 02ED       0240  310+        DC    AL2(BUF2)                    ADDR OF START OF I/O AREA.
    0241 0442       0242  311+        DC    AL2(B2END)                   ADDR OF END OF I/O AREA.
    0243 0000000000 0247  312+        DC    XL5'00'                      BSCA WORK AREA.
    0248 000000     024A  313+        DC    XL3'00'                      DEFAULT-NO RVI.
    024B 000000000000 0250 314+       DC    XL6'00'                      BSCA WORK AREA.
    0251 C0         0251  315+        DC    AL1(00+00)                   TERM ATTR.
    0252 00         0252  316+        DC    XL1'00'                      RESERVED.
    0253 C00000000000 0258 317+       DC    XL6'00'                      BSCA WORK AREA.
    0259 05F5       025A  318+        DC    AL2(LOG)                     ADDR OF TERM LOG AREA.
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE       ADDR  STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE  10

                                  320 *    BUFFER AREAS DATA AREAS HALT/SYSLOG PARAMETERS


                           025B   322 BUF1   EQU   *
 025B C000000000000000     02EC   323 B1END  DC    XL146'00'                BUFFER FOR DTF 1
 0263 C000000000000000           323
 026B 0000000000000000           323
 0273 0000000000000000           323
 027B 0000000000000000           323
 0283 0000000000000000           323
 028B 0000000000000000           323
 0293 0000000000000000           323
 029B 0000000000000000           323
 02A3 0000000000000000           323
 02AB 0000000000000000           323
 02B3 C000000000000000           323
 02BB 0000000000000000           323
 02C3 0000000000000000           323
 02CB C000000000000000           323
 02D3 0000000000000000           323
 02DB 0000000000000000           323
 02E3 0000000000000000           323
 02EB C000                       323
                           02ED   324 BUF2   EQU   *
 02ED 0000000000000000     0442   325 B2END  DC    342XL1'00'               BUFFER FOR DTF 2
 02F5 C000000000000000           325
 02FD 0000000000000000           325
 0305 0000000000000000           325
 030D C000000000000000           325
 0315 0000000000000000           325
 031D C000000000000000           325
 0325 0000000000000000           325
 032D C000000000000000           325
 0335 C000000000000000           325
 033D C000000000000000           325
 0345 0000000000000000           325
 034D 0000000000000000           325
 0355 C000000000000000           325
 035D 0000000000000000           325
 0365 0000000000000000           325
 036D C000000000000000           325
 0375 0000000000000000           325
 037D 0000000000000000           325
 0385 C000000000000000           325
 038D 0000000000000000           325
 0395 0000000000000000           325
 039D C000000000000000           325
 03A5 C000000000000000           325
 03AD 0000000000000000           325
 03B5 0000000000000000           325
 03BD C000000000000000           325
 03C5 0000000000000000           325
 03CD C000000000000000           325
 03D5 C000000000000000           325
 03DD 0000000000000000           325
 03E5 C000000000000000           325
```

```
S3270  ML MP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  CBJECT CODE       ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE  11

     03ED C000000000000000        325
     03F5 0000000000000000        325
     03FD 0000000000000000        325
     0405 0000000000000000        325
     040D 0000000000000000        325
     0415 C000000000000000        325
     041D C000000C00000000        325
     0425 0000000000000000        325
     042D C000000000000000        325
     0435 0000C00000000000        325
     043D C00000000000            325
                                  326 * COMMAND AND ORDER CHAIN FOR 3270 DISPLAY
                          0443    327 WORK1  EQU   *
     0443 27F5C71140C41DF0 044A   328        DC    XL8'27F5C71140C41DF0'
     044B D5C1D4C5406040   0451   329        DC    CL7'NAME -'
     0452 1D4013           0454   330        DC    XL3'1D4013'
     0455 4040404040404040 0474   331        DC    CL32' '                    ASSIGNED NAME FIELD
     045D 4040404040404040        331
     0465 4040404040404040        331
     046D 4040404040404040        331
     0475 1DF011C1D41DF0   047B   332        DC    XL7'1DF011C1D41DF0'
     047C C1C4C4D9C5E2E240 0485   333        DC    CL10'ADDRESS - '
     0484 6040                    333
     0486 1D40             0487   334        DC    XL2'1D40'
     0488 4040404040404040 04A5   335        DC    CL30' '                    ASSIGNED ADDRESS FIELD
     0490 4040404040404040        335
     0498 4040404040404040        335
     04A0 40404040404040          335
     04A6 1DF011404D       04AA   336        DC    XL5'1DF011404D'            SET END OF FIELD AND BUFFER ADD
                          04AB    337 WORK2  EQU   *
     04AB 0000000000000000 05D6   338        DC    300XL1'00'                 OPERATOR RESPONSE BUFFER
     04B3 0000000C00000000        338
     04BB 0000000000000000        338
     04C3 0000000000000000        338
     04CB 0000000C00000000        338
     04D3 0000000000000000        338
     04DB 0000000000000000        338
     04E3 0000000000000000        338
     04EB 0000000000000000        338
     04F3 0000000000000000        338
     04FB 0000000C00000000        338
     0503 0000000000000000        338
     050B 0000000000000000        338
     0513 0000000C00000000        338
     051B C000000000000000        338
     0523 0000000000000000        338
     052B C000000000000000        338
     0533 C000000000000000        338
     053B 0000000000000000        338
     0543 C000000000000000        338
     054B 0000000000000000        338
     0553 0000000000000000        338
     055B C000000000000000        338
     0563 C000000C00000000        338
     056B C000000000000000        338
     0573 0000000000000000        338
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE       ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE  12

     057B C000000C00000000        338
     0583 0000000000000000        338
     058B 0000000000000000        338
     0593 0000000000000000        338
     059B 0000000000000000        338
     05A3 0000000000000000        338
     05AB 0000000000000000        338
     05B3 0000000000000000        338
     05BB 0000000000000000        338
     05C3 0000000000000000        338
     05CB 0000000000000000        338
     05D3 C0000000                338


                                  340 *   HALT / SYSLOG PARAMETERS FOR PRINTING
                             05D7 341 RETURN EQU   *
     05D7 200020             05D9 342        DC    XL3'2C0020'
     05DA C628               05DB 343        DC    AL2(RETRN)
                             05DC 344 NAME   EQU   *
     05DC 200030             05DE 345        DC    XL3'200030'
     05DF 0688               05E0 346        DC    AL2(NBEGIN)
                             05E1 347 ADDR   EQU   *
     05E1 200030             05E3 348        DC    XL3'200030'
     05E4 C688               05E5 349        DC    AL2(ABEGIN)
                             05E6 350 ERROR  EQU   *
     05E6 200020             05E8 351        DC    XL3'200020'
     05E9 C648               05EA 352        DC    AL2(ERMESS)
                             05EB 353 SMESSG EQU   *
     05EB 200030             05ED 354        DC    XL3'200030'
     05EE C6EB               05EF 355        DC    AL2(SMSG)
                             05F0 356 HLMESS EQU   *
     05F0 200020             05F2 357        DC    XL3'200020'
     05F3 0668               05F4 358        DC    AL2(HMESS)
                             05F5 359 LOG    EQU   *
     05F5 001A               05F6 360        DC    XL2'001A'
     05F7 C000000C00000000   060E 361        DC    XL24'00'                  ERROR LOGGING TABLE
     05FF 0000000000000000        361
     0607 C000000000000000        361
                                  362 *ELECT $POLB ID-20,TERMAD-60604040,LEN-4,LAST-WRAP
                                  363+*      GENERATE A POLL/ADDR LIST.     RELEASE-7  A
                             060F 364+SELECT EQU   *
     060F 20                 060F 365+       DC    XL1'20'
     0610 04                 0610 366+       DC    AL1(4)                    LENGTH OF POLL/ADDR.
     0611 60604040           0614 367+       DC    XL4'60604040'             POLL/ADDR CHARS.
     0615 00                 0615 368+       DC    XL1'00'                   STATUS.
     0616 FF                 0616 369+       DC    XL1'FF'                   END OF LIST.
                                  370 *OLL   $POLB ID-20,TERMAD-40407F7F,LEN-4,LAST-WRAP
                                  371+*      GENERATE A POLL/ADDR LIST.     RELEASE-7  A
                             0617 372+POLL   EQU   *
     0617 20                 0617 373+       DC    XL1'20'
     0618 04                 0618 374+       DC    AL1(4)                    LENGTH OF POLL/ADDR.
     0619 40407F7F           061C 375+       DC    XL4'40407F7F'             POLL/ADDR CHARS.
     061D C0                 061D 376+       DC    XL1'00'                   STATUS.
     061E FF                 061E 377+       DC    XL1'FF'                   END OF LIST.
                                  378 *IST1  $CKL   DTF-DTF1,LAST-Y
                                  379+*      CHECK LIST FOR COMMON WAIT     RELEASE-7  B
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE       ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE  13

    061E 20                061E  380+LIST1  DC    AL1(0+0+32+0+0)          ENTRY STATUS BYTE.
    0620 01DF              0621  381+        DC    AL2(DTF1)                DTF ADDRESS.

                                 383 *IST2   $CKL   DTF-DTF2,LAST-Y
    0622 20                0622  384+LIST2   DC    AL1(0+0+32+0+0)          ENTRY STATUS BYTE.
    0623 021D              0624  385+        DC    AL2(DTF2)                DTF ADDRESS.

    0625 016CD9            0627  387 STATUS  DC    XL3'016CD9'
                           04D6  388 NFIELD  EQU   WORK2+43
                           04F2  389 AFIELD  EQU   WORK2+71
                           0001  390 XR1     EQU   1
                           0002  391 XR2     EQU   2
                           0003  392 ETX     EQU   3
                           00F0  393 ZERO    EQU   X'F0'
                           006D  394 CLEAR   EQU   X'6D'
                           007D  395 ENTER   EQU   X'7D'
                           0080  396 RVI     EQU   X'80'
                           0009  397 NINE    EQU   9
                           0012  398 EIGTEN  EQU   18
                           00F9  399 NONPRT  EQU   X'F9'
                           0628  400 RETRN   EQU   *
    0628 5BC2C4C3D4D740C3  0647  401 RET     DC    CL32'$BDCMP COMPLETION CODE IS -'
    0630 D6D4D7D3C5E3C9D6        401
    0638 E540C3D6C4C540C9        401
    0640 E240604040404040        401
                           0648  402 ERMESS  EQU   *
    0648 C9D5E5C1D3C9C440  0667  403         DC    CL32'INVALID KEY HAS BEEN USED'
    0650 D2C5E840C8C1E240        403
    0658 C2C5C5D540F4F2C5        403
    0660 C4404040404040          403
                           0668  404 HMESS   EQU   *
    0668 C3C5C6C540C3D6D9  0687  405         DC    CL32'CEFE CORE DUMP MAY NOW BE TAKEN'
    0670 C540C4E4D4D740D4        405
    0678 C1E840D5D6E640C2        405
    0680 C540E3C1D2C5D540        405
                           0688  406 NBEGIN  EQU   *
    0688 D5C1D4C540D2C5E8  0699  407         DC    CL18'NAME KEYED -'
    0690 C5C4406040404040        407
    0698 4040                    407
    069A 4040404040404040  06B7  408 NEND    DC    CL30' '
    06A2 4040404040404040        408
    06AA 4040404040404040        408
    06B2 40404040404040          408
                           06B8  409 ABEGIN  EQU   *
    06B8 C1C4C4D9C5E2E240  06C9  410         DC    CL18'ADDRESS KEYED -'
    06C0 D2C5E8C5C4406040        410
    06C8 4040                    410
    06CA 4040404040404040  06E7  411 AEND    DC    CL30' '
    06D2 4040404040404040        411
    06DA 4040404040404040        411
    06E2 40404040404040          411
    06E8 00                06E8  412 SCOUNT  DC    XL1'00'
    06E9 01                06E9  413 ONE     DC    XL1'01'
    06EA 39                06EA  414 SB39    DC    XL1'39'
                           06EB  415 SMSG    EQU   *
    06EB E2E3C1E3E4E240D4  0708  416         DC    CL30'STATUS MESSAGE RECEIVED IS -'


S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE       ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE  14

    06F3 C5E2E2C1C7C540D9        416
    06FB C5C3C5C9E5C5C440        416
    0703 C9E2406C4040            416
                           0709  417 STATB   EQU   *
    0709 4040404040404040  071A  418 STAT    DC    CL18' '
    0711 4040404C40404040        418
    0719 4040                    418
    071B 00                071B  419 MASK    DC    XL1'00'                  RVI MASK HOLD AREA
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE  15

                           421 *       $DTOB
                           422+*       BSCA EQUATES.                  RELEASE-13

                           424+*       EQUATES IN IOB
                     0005  425+$BIFLA  EQU   5                        FLAG A.
                     0004  426+$BIFST  EQU   X'04'                    FIRST IN FLAG A.
                     0002  427+$BIOBQ  EQU   X'02'                    OFFSET TO IOB OPERATION.
                     0083  428+$BIRVI  EQU   X'83'                    RCVI OPERATION.
                     0007  429+$BICMP  EQU   X'07'                    OFFSET TO IOB COMPLETION.

                           431+*       EQUATES FOR WORK AREA.
                     0001  432+$BWK    EQU   1                        WORK AREA REG.
                     001D  433+$BWFG3  EQU   29                       FLAG 3.
                     0023  434+$BWLGD  EQU   35                       FLAG BYTE.
                     0010  435+$BPATV  EQU   X'10'                    LINE ACTIVE INDICATOR.
                     0008  436+$BWRFT  EQU   X'08'                    RFT IND. IN FLAG 3.
                     0017  437+$BPOLD  EQU   X'17'                    OFFSET TO POLL INDICS IN WKA.
                     0004  438+$BPCNC  EQU   X'04'                    CANCEL POST INDIC.
                     0010  439+$BPRES  EQU   X'10'                    RESET POLL INDIC.
                     0059  440+$BWKMC  EQU   X'59'                    OFFSET FOR LDA INDICS.
                     0006  441+$B2SEC  EQU   X'06'                    TWO SEC TIME FOR LDA.

                           443+*       EQUATES FOR $BCPL AND $BCSW MACROS.
                     0000  444+$BPATR  EQU   0                        CHANGE LIST ATTRIBUTE OFFSET.
                     0080  445+$BPACT  EQU   X'80'                    OFF-ACTIVATE;ON-DEACTIVATE.
                     0040  446+$BPEXT  EQU   X'40'                    OFF-EXACT,ON-FIRST N CHARS.
                     0002  447+$BPDTF  EQU   2                        CHANGE LIST DTF ADDR OFFSET.
                     0003  448+$BPNUM  EQU   3                        CHANGE LIST OFFSET TO LENGTH.
                     00FE  449+$BPEND  EQU   X'FE'                    END OF POLL/ADDR OR SW ID LIST.
                     0000  450+$BPNOP  EQU   X'00'                    NO-OP JUMP INSTR.
                     0080  451+$BPENA  EQU   X'80'                    ON-ACTIVE;OFF-INACT. LIST ATTR.
                     0001  452+$BPRM1  EQU   1                        REG EQU FOR MACRO PARM LIST.
                     0002  453+$BLST2  EQU   2                        REG EQU FOR POLL OR ID LIST- XR2
                     0002  454+$BLIST  EQU   2                        REG EQU FOR PTR TO LIST IN XR2.

                           456+*       EQUATES FOR $RFT MACRO
                     0003  457+$BRCNT  EQU   3                        COUNT OF NUMBER OF TRANSMISSIONS
                     000F  458+$BHXOF  EQU   X'0F'                    MASK TO CHECK FOR DECIMAL NUMBER

                           460+*       GENERAL EQUATES.
                     0080  461+$BDISA  EQU   X'80'                    ENABLE BSCA.
                     00C0  462+$BENAB  EQU   X'C0'                    DISABLE BSCA.
                     00FF  463+$BFOX   EQU   X'FF'                    EQUATE FOR 'FF'.
                     0001  464+$BBAC1  EQU   1                        USER REGISTER SAVE (REG 1).
                     0002  465+$BPRS2  EQU   2                        PARAMETER REGISTER SAVE (REG 2).


                           467+*       EQUATES FOR $CANB MACRO
                     0008  468+$BLIN2  EQU   X'08'                    LINE-2.
                     0016  469+$BTREQ  EQU   X'16'                    TRUE AND EQUAL
                     0010  470+$BDAON  EQU   X'10'                    D.A. SUPPORTED
                     0088  471+$BTOSC  EQU   X'88'                    TWO SEC TIME OUT
                     0011  472+$BTRNQ  EQU   X'11'                    TRUE AND NOT EQUAL
                     0002  473+$BDTF   EQU   2                        DTF REG.
                     0001  474+$BIOB   EQU   1                        IOB REG.
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE  16

                          0001  475+$BONE   EQU   1                        OFFSET FOR CONSTANT ONE
                          0003  476+$BTRE   EQU   3                        OFFSET FOR CONSTANT THREE


                                478+*               OFFSETS FOR BSCA DTF.
                          0000  479+$BDDEV  EQU   0                        DEVICE ID.
                          0001  480+$BDUPS  EQU   1                        UPSI.
                          0002  481+$BDATT  EQU   2                        ATTRIBUTE BYTE 1.
                          0080  482+$BCINP  EQU   X'80'                    INPUT FILE.
                          0040  483+$BCOUT  EQU   X'40'                    OUPUT FILE.
                          00C0  484+$BCCNV  EQU   X'C0'                    CONVERSATIONAL FILE.
                          0020  485+$BCITB  EQU   X'20'                    ITB MODE.
                          0010  486+$BCRAN  EQU   X'10'                    TRANSPARENCY.
                          0008  487+$BCGET  EQU   X'08'                    GET FILE.
                          0004  488+$BCASK  EQU   X'04'                    ON-ASCII; OFF-EBCDIC.
                          0001  489+$BCASM  EQU   X'01'                    ASSEM DTF.
                          0003  490+$BDATR  EQU   3                        ATTRIBUTE BYTE 2.
                          0088  491+$BCMCN  EQU   X'88'                    MULTIPOINT CONTROL STATION.
                          0080  492+$BCMPT  EQU   X'80'                    MULTIPOINT TRIBUTARY.
                          0020  493+$BCMAN  EQU   X'20'                    MANUAL LINE.
                          0010  494+$BCANS  EQU   X'10'                    ANSWER LINE.
                          0008  495+$BCSWI  EQU   X'08'                    SWITCHED LINE.
                          0004  496+$BCUSD  EQU   X'04'                    FILE USED.
                          0002  497+$BCACT  EQU   X'02'                    FILE ACTIVE.
                          0001  498+$BCOPN  EQU   X'01'                    FILE OPENED.
                          0005  499+$BDCHN  EQU   5                        POST OPEN DTF CHAINING PTR.
                          0007  500+$BDNXT  EQU   7                        DTF CHAINING POINTER.
                          0009  501+$BDWK1  EQU   9                        WORK AREA.
                          000B  502+$BDWK2  EQU   11                       WORK AREA.
                          000D  503+$BDWKB  EQU   13                       ADDRESS OF USER'S LOGICAL BUFF.
                          000E  504+$BDCMP  EQU   14                       COMPLETION CODE.
                          0000  505+$BCREQ  EQU   X'00'                    REQUEST ACCEPTED.
                          0040  506+$BCONE  EQU   X'40'                    NORMAL COMPLETION.
                          0041  507+$BCUER  EQU   X'41'                    USER ERROR.
                          0042  508+$BCEOT  EQU   X'42'                    END OF FILE.
                          0043  509+$BCBID  EQU   X'43'                    INVALID ID.
                          0044  510+$BCNEG  EQU   X'44'                    NEGATIVE RESPONSE TO POLL/ADDR.
                          0045  511+$BCNON  EQU   X'45'                    NO RESPONSE TO POLL/ADDR.
                          0046  512+$BCCRP  EQU   X'46'                    CONV REPLY PENDING.
                          0047  513+$BCNDT  EQU   X'47'                    NO DATA FOR CONV GET.
                          0048  514+$BCOLT  EQU   X'48'                    INVALID RET REQUEST.
                          0049  515+$BCNAC  EQU   X'49'                    NO ACT ENTRY IN POLL LIST.
                          004A  516+$BCIGN  EQU   X'4A'                    REQUEST IGNORED.
                          004B  517+$BCASC  EQU   X'4B'                    INVALID ASCII CHARACTER.
                          004C  518+$BCNCN  EQU   X'4C'                    NO-CONNECTION.
                          004D  519+$BCCAL  EQU   X'4D'                    INVALID REQUEST.
                          004E  520+$BCLST  EQU   X'4E'                    DELAY COUNT EXCEEDED.
                          004F  521+$BCERR  EQU   X'4F'                    PERM ERROR.
                          0050  522+$BCTIM  EQU   X'50'                    NO RESP FROM REMOTE DEV.
                          0051  523+$BCDAT  EQU   X'51'                    DATA CHECK.
                          0052  524+$BCLOS  EQU   X'52'                    LOST DATA.
                          0053  525+$BCCON  EQU   X'53'                    LOST CONNECTION.
                          0054  526+$BCRSP  EQU   X'54'                    INVALID RESP FROM REMOTE DEV.
                          0055  527+$BCADP  EQU   X'55'                    ADAPTER CHECK.
                          0056  528+$BCCMP  EQU   X'56'                    NO COMPLETIONS IN CHECK LIST.
```

S3270 4LMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC CBJECT CCDE ADDR STMT SOURCE STATEMENT VER 14, MOD 00 08/26/77 PAGE 17

```
0057  529+$BCACD EQU   X'57'                    NO ACTIVE DTFS IN CHECK LIST.
0058  530+$BCRLE EQU   X'58'                    MAXIMUM RECORD LENGTH EXCEEDED.
000F  531+$BDOPC EQU   15                       OPERATION CODE.
0080  532+$BDGET EQU   X'80'                    GET.
0081  533+$BDGBK EQU   X'81'                    GET-BLOCK.
0040  534+$BDPUT EQU   X'40'                    PUT.
0041  535+$BDPEB EQU   X'41'                    PUT END OF BLOCK.
0042  536+$BDPEF EQU   X'42'                    PUT END OF FILE.
0044  537+$BDPEW EQU   X'44'                    PUT EOT TO WACK RESPONSE.
0011  538+$BDMRL EQU   17                       MAXIMUM RECORD LENGTH.
0012  539+$BDADD EQU   18                       SPECIAL USE INDICATORS
0001  540+$BCAA1 EQU   X'01'                    ADD ON AREA ON DTF
0002  541+$BCPOL EQU   X'02'                    POLLING MODULES RESIDENT
0004  542+$BCOFL EQU   X'04'                    TRUNCATE RECORD INDICATOR.
0008  543+$BCRCL EQU   X'08'                    SPAN INDICATOR FOR RECORD LENGTH
0010  544+$BCTWO EQU   X'10'                    END OF BLOCK INDICATOR.
0080  545+$BCSWD EQU   X'80'                    ID LIST FOR SWICHED LINE
0014  546+$BDDCH EQU   20                       *ADDRESS OF DIAL NUMBER OR
0014  547+$BDPSC EQU   20                       *POLL/ADDR CHARACTERS OR
0014  548+$BDLST EQU   20                       *ADDRESS OF POLL/ADDR LIST.
0015  549+$BDDCC EQU   21                       *LENGTH OF DIAL NUMBER OR
0015  550+$BDIND EQU   21                       *POLLING/OR ADDRESSING ID.
0017  551+$BDRID EQU   23                       *ADDR OF RCV ID OR ID LIST OR
0017  552+$BDCNT EQU   23                       *LIST COUNT.
0018  553+$BDRLN EQU   24                       LEN OF RCV ID OR ENTRY SELECTOR.
0018  554+$BDLID EQU   24                       LAST ID OR POLL/ADDR FUNCTION.
001A  555+$BDSID EQU   26                       ADDRESS OF SEND ID.
001B  556+$BDSLN EQU   27                       LENGTH OF SEND ID.
001D  557+$BDDLY EQU   29                       DELAY COUNT.
001F  558+$BDREL EQU   31                       RECORD LENGTH.
0021  559+$BDBKL EQU   33                       BLOCK LENGTH.
0023  560+$BDIOB EQU   35                       ADDRESS OF IOB IN PROCESS.
0025  561+$BDBKX EQU   37                       POINT TO DATA IN BSCA BUFFER.
0027  562+$BDITB EQU   39                       ITB CHARACTER COUNT.
002A  563+$BDPRM EQU   42                       RESERVED.
002D  564+$BDRVI EQU   45                       RVI MASK AND DISPLACEMENT.
002E  565+$BDNDX EQU   46                       INDEX FOR LINE INITIALIZATION.
0030  566+$BDWKA EQU   48                       ADDRESS OF BSCA WORK AREA.
0032  567+$BDINT EQU   50                       DISK ADDR OF LINE INIT MODULE.
0033  568+$BDDED EQU   51                       WORK AREA.
0034  569+$BDATI EQU   52                       ATTRIBUTE BYTE FOR TERMINALS.
0001  570+$BCSEP EQU   X'01'                    RECORD SEPARATOR.
0002  571+$BCSPN EQU   X'02'                    SPANNING RECORD.
0004  572+$BCNOW EQU   X'04'                    SPAN IN PROCESS.
0008  573+$BCPUT EQU   X'08'                    PUT SPAN FILE.
0010  574+$BCRES EQU   X'10'                    SPAN RESTORE NECESSARY.
0040  575+$BCPLR EQU   X'40'                    POLLING RESIDENT.
0035  576+$BDSEP EQU   53                       RECORD SEPARATOR.
0037  577+$BDSBF EQU   55                       SAVE AREA FOR USER BUFFER ADDR.
0039  578+$BDSRL EQU   57                       SAVE AREA FOR RECORD LENGTH.
003B  579+$BDRFT EQU   59                       SAVE AREA FOR OLT PARM.
003D  580+$BDTSA EQU   61                       ADDR OF TERM LCG AREA.
      581+**           ADD ON AREA OF DTF
003F  582+$BDRLO EQU   63                       ADDR OF RESIDENT LO.
0041  583+$BDRCL EQU   65                       ADDR OF RESIDENT CLOSE.
0043  584+$BDARA EQU   67                       AUTO RESPONSE MODULE.
```

```
S3270  MLMP 3270 INFORMATION DISPLAY SYSTEM SAMPLE PROGRAM

ERR LOC  OBJECT CODE      ADDR STMT SOURCE STATEMENT          VER 14, MOD 00  08/26/77  PAGE  18

                          0044  585+$BDERR EQU   68                        RETRY COUNT.
                          0046  586+$BDT1A EQU   70                        SAVE ADDRESS OF OLT CS.
                          0049  587+$BD375 EQU   73                        3735 CCP EOT INDIC.
                                588 *      $COMN
                                589+*     COMMON LABELS FOR SYSTEM MACROS   RELEASE-7
                          0001  590+$IOB   EQU   1                         IOB ADDRESS
                          0002  591+$DTF   EQU   2                         DTF ADDRESS
                          0002  592+$LDADR EQU   2                         LOAD ADDRESS
                          0002  593+$PARM  EQU   2                         PARAMETER LIST ADDRESS
                          0008  594+$ARR   EQU   8                         ADDRESS RECALL REGISTER
                          0010  595+$IAR   EQU   16                        INSTRUCTION ADDRESS REGISTER
                          0010  596+$TRU   EQU   X'10'                     TEST TRUE
                          0090  597+$FLS   EQU   X'90'                     TEST FALSE
                          0004  598+$HI    EQU   X'04'                     TEST HIGH
                          0002  599+$LO    EQU   X'02'                     TEST LOW
                          0001  600+$EQ    EQU   X'01'                     TEST EQUAL
                          0020  601+$OVFB  EQU   X'20'                     BINARY OVERFLOW
                          0008  602+$OVFD  EQU   X'08'                     DECIMAL OVERFLOW
                          0000  603        END   S3270

TOTAL STATEMENTS IN ERROR IN THIS ASSEMBLY =      0
```

S3270 CROSS REFERENCE

SYMBOL LEN VALUE DEFN REFERENCES VER 14, MOD 00 08/26/77 PAGE 19

```
$$BMCH  001  0003 0080   0078  0137
$$BSLG  001  0004 0251
$$BSMS  001  0001 0003   0060  0119
$$BSTT  001  0002 0004
$ARR    001  0008 0594
$BBAC1  001  0001 0464   0061  0064* 0109  0112* 0120  C123*
$BCAA1  001  0001 0540
$BCACD  001  0057 0529
$BCACT  001  0002 0497
$BCADP  001  0055 0527
$BCANS  001  0010 0494
$BCASC  001  004B 0517
$BCASK  001  0004 0488
$BCASM  001  0001 0489
$BCBID  001  0043 0509
$BCCAL  001  004D 0519   0063  0071  0122  0130
$BCCMP  001  0056 0528
$BCCNV  001  00C0 0484   0055  0113
$BCCON  001  0053 0525
$BCCRP  001  0046 0512   0056  0116
$BCDAT  001  0051 0523
$BCDNE  001  0040 0506   0088  0141
$BCEOT  001  0042 0508   0139
$BCERR  001  004F 0521
$BCGET  001  0008 0487
$BCIGN  001  004A 0516   0067  0126
$BCINP  001  0080 0482   0107
$BCITB  001  0020 0485
$BCLOS  001  0052 0524
$BCLST  001  004E 0520
$BCMAN  001  0020 0493
$BCMCN  001  0088 0491
$BCMPT  001  0080 0492
$BCNAC  001  0049 0515
$BCNCN  001  004C 0518
$BCNDT  001  0047 0513
$BCNEG  001  0044 0510   0082
$BCNON  001  0045 0511
$BCNOW  001  0004 0572
$BCOFL  001  0004 0542
$BCOLT  001  0048 0514
$BCOPN  001  0001 0498   0052  0106
$BCOUT  001  0040 0483   0053
$BCPLR  001  0040 0575
$BCPOL  001  0002 0541
$BCPUT  001  0008 0573
$BCRAN  001  0010 0486
$BCRCL  001  0008 0543
$BCREQ  001  0000 0505   0050  0059  0104
$BCRES  001  0010 0574
$BCRLE  001  0058 0530
$BCRSP  001  0054 0526
$BCSEP  001  0001 0570
$BCSPN  001  0002 0571
$BCSWD  001  0080 0545
$BCSWI  001  0008 0495
```

S3270 CROSS REFERENCE

SYMBOL LEN VALUE DEFN REFERENCES VER 14, MOD 00 08/26/77 PAGE 20

```
$BCTIM   001  0050 0522
$BCTWO   001  0010 0544
$BCUER   001  0041 0507   0069  0128
$BCUSD   001  0004 0496
$BDADD   001  0012 0539
$BDAON   001  0010 0470
$BDARA   001  0043 0584
$BDATR   001  0003 0490   0052  0106
$BDATT   001  0002 0481   0053  0055  0107  0113
$BDAT1   001  0034 0565
$BDBKL   001  0021 0559
$BDBKX   001  0025 0561
$BDCHN   001  0005 0499
$BDCMP   001  000E 0504   0050  0056  0059* 0067* 0069* 0071* 0082  0088  0104  0116  0126* 0128*
                          0130* 0139  0141  0217  0218
$BDCNT   001  0017 0552
$BDDCC   001  0015 0549
$BDDCH   001  0014 0546
$BDDED   001  0033 0568
$BDDEV   001  0000 0479
$BDDLY   001  001D 0557
$BDERR   001  0044 0585
$BDIND   001  0015 0550
$BDINT   001  0032 0567
$BDIOB   001  0023 0560   0062  0110  0121
$BDISA   001  0080 0461
$BDITB   001  0027 0562
$BDLID   001  0018 0554
$BDLST   001  0014 0548
$BDMRL   001  0011 0538
$BDNDX   001  002E 0565
$BDNXT   001  0007 0500
$BDOPC   001  000F 0531   0058* 0115  0118*
$BDPRM   001  002A 0563
$BDPSC   001  0014 0547
$BDRCL   001  0041 0583
$BDREL   001  001F 0558
$BDRET   001  003B 0575
$BDRID   001  0017 0551
$BDRLN   001  0018 0553
$BDRLO   001  003F 0582
$BDRVI   001  002D 0564
$BDSBF   001  0037 0577
$BDSEP   001  0035 0576
$BDSID   001  001A 0555
$BDSLN   001  0018 0556
$BDSRL   001  0039 0578
$BDTF    001  0002 0473
$BDTSA   001  003D 0580
$BDT1A   001  0046 0586
$BDUPS   001  0001 0480
$BDWKA   001  0030 0566
$BDWKB   001  000D 0503
$BDWK1   001  0009 0501
$BDWK2   001  0008 0502
$BD375   001  0049 0587
```

S3270 CROSS REFERENCE

SYMBOL LEN VALUE DEFN REFERENCES VER 14, MOD 00 08/26/77 PAGE 21

```
$BENAB  0C1  00C0 0462
$BFOX   0C1  00FF 0463
$BHXOF  0C1  000F 0458
$BICMP  001  0007 0429   0063  0122
$BIFLA  0C1  00C5 0425   0111
$BIFST  001  0004 0426   0111
$BIOB   0C1  0001 0474
$BIOBQ  0C1  0002 0427
$BIRVI  0C1  0083 0428
$BLIN2  0C1  0008 0468
$BLIST  001  0002 0454
$BLST2  001  0002 0453
$BOGBK  001  0081 0533   0118
$BOGET  0C1  0080 0532
$BONE   0C1  0001 0475
$BOPEB  0C1  0041 0535
$BOPEF  0C1  0042 0536
$BOPEW  0C1  0044 0537
$BOPUT  001  0040 0534   0058  0115
$BPACT  001  0080 0445
$BPATR  0C1  0000 0444
$BPATV  0C1  0010 0435
$BPCNC  001  0004 0438
$BPDTF  0C1  0002 0447
$BPENA  0C1  0080 0451
$BPEND  0C1  00FE 0449
$BPEXT  0C1  0040 0446
$BPNOP  0C1  0000 0450
$BPNUM  001  0003 0448
$BPOLD  0C1  0017 0437
$BPRES  0C1  0010 0439
$BPRM1  0C1  0001 0452
$BPRS2  0C1  0002 0465
$BRCNT  001  0003 0457
$BTOSC  001  0088 0471
$BTRE   0C1  0003 0476
$BTREQ  001  0016 0469
$BTRNQ  0C1  0011 0472
$BWFG3  001  001D 0433
$BWK    001  0001 0432
$BWKMC  0C1  0059 0440
$BWLGD  0C1  0023 0434
$BWRFT  001  0008 0436
$B2SEC  0C1  0006 0441
$DTF    001  0002 0591   0032* 0037* 0049* 0050  0052  0053  0055  0056  0058  0059  0062  0067
                         0069  0071  0082  0088  0103* 0104  0106  0107  0110  0113  0115  0116
                         0118  0121  0126  0128  0130  0139  0141  0217  0218  0239*
$EQ     001  0001 0600   0117
$E1003  0C3  004C 0067   0051
$E1005  0C3  00CA 0126   0105
$E2003  003  0052 0069   0054
$E2005  0C3  00D0 0128   0108
$E3003  0C3  0058 0071   0057  0065
$E3005  003  00D6 0130   0117  0124
$FLS    0C1  0090 0597
$GT005  0C3  00AF 0118   0114
$HI     001  0004 0598   0057
$IAR    0C1  0010 0595
$IOB    0C1  0001 0590   0062* 0063  0110* 0111  0121* 0122
$LDADR  001  0002 0592
$LD     0C1  0002 0599   0057
$OVFB   0C1  0020 0601
$OVFD   001  0008 0602
$PARM   0C1  0002 0593   0077* 0136*
$RJ003  0C1  005B 0072   0068  0070
$RJ005  001  00D9 0131   0127  0129
$SV005  0C4  009C 0112   0109*
$S1003  0C4  C042 0064   0061*
$S1005  0C4  00C0 0123   0120*
$TRU    0C1  0010 0596   0057  0117
$XT003  001  005F 0074   0066
$XT005  0C1  00DD 0133   0125
```

```
S3270                              CROSS REFERENCE

SYMBOL   LEN VALUE DEFN   REFERENCES                          VER 14, MOD 00  08/26/77  PAGE  22

ABEGIN   001  06B8 0409   0349
ADDR     001  05E1 0347   0205
AEND     030  06E7 0411   0204*
AFIELD   001  04F2 0389   0204
BADRET   001  019F 0215   0073  0089  0132  0142
BUF1     001  025B 0322   0274
BUF2     001  02FD 0324   0310
B1END    146  02EC 0323   0275
B2END    001  0442 0325   0311
CKGOOD   001  0078 0087   0083
CLEAR    001  006D 0394   0143
CLOSE    001  01D1 0236   0196
DATA     001  0175 0194   0148
DTF1     001  01DF 0252   0032  0037  0049  0239  0381
DTF2     001  021D 0288   0103  0258  0385
EIGTEN   001  0012 0398   0174
ENTER    001  007D 0395   0147
ERMESS   001  0648 0402   0352
ERROR    001  05E6 0350   0149
ETX      001  0003 0392   0195
HLMESS   001  05F0 0356   0229
HMESS    001  0668 0404   0358
LIST1    001  061F 0380   0077
LIST2    001  0622 0384   0136
LOG      001  05F5 0359   0283  0318
MASK     001  071B 0419   0084  0085* 0278
MORE     001  007F 0099   0144  0155  0189  0211
MOVST    001  0130 0166   0172
NAME     001  05DC 0344   0198
NBEGIN   001  0688 0406   0346
NEND     030  06B7 0408   0197*
NFIELD   001  04D6 0388   0197
NINE     001  0009 0397   0163
NONPRT   001  00F9 0399   0176  0219
NOS839   001  0158 0179   0177
NOSET    001  01BC 0222   0220
ONE      001  06E9 0413   0171  0181
POLL     001  0617 0372   0301
PSTAT    001  011A 0160   0146
RESPON   001  007F 0098   0086
RET      032  0647 0401   0216  0216* 0217* 0218* 0219  0221*
RETRN    001  0628 0400   0343
RETURN   001  05D7 0341   0223
RVI      001  0080 0396   0084  0085
S839     001  06EA 0414   0178  0221
SCOUNT   001  06E8 0412   0163* 0171* 0174* 0181*
SELECT   001  060F 0364   0265
SETDIS   001  0012 0045   0140
SETPRT   001  0150 0175   0182
SMESSG   001  05EB 0353   0183
SMSG     001  06EB 0415   0355
STAT     018  071A 0418   0161* 0162  0162*
STATB    001  0709 0417   0164  0173
STATUS   003  0627 0387   0145
S3270    001  0000 0002   0603
WORK1    001  0443 0327   0260
WORK2    001  04AB 0337   0143  0145  0147  0165  0195  0296  0388  0389
XR1      001  0001 0390   0164* 0167  0168  0169  0169* 0173* 0176  0178  0180  0180*
XR2      001  0002 0391   0149* 0165* 0167  0168  0170  0170* 0183* 0198* 0205* 0223* 0229*
ZERO     001  00F0 0393   0161

TOTAL STATEMENTS IN ERROR IN THIS ASSEMBLY =      0

OL105 I  THE CODE LENGTH OF S3270  IS  1820 DECIMAL.
OL103 I  TOTAL NUMBER OF LIBRARY SECTORS REQUIRED IS    9
         NAME-S3270 ,PACK-R1R1R1,UNIT-R1,RETAIN-T,LIBRARY-R,CATEGORY-000
```

BSC DTF

Displacement Hex	Displacement Decimal	Label	Length in Bytes	Description
0	0	$BDDEV	1	Device ID: X'80' for BSCA 1 X'88' for BSCA 2
1	1	$BDUPS	1	UPSI: U1-U8, customer controlled program switches
2	2	$BDATT	1	Attribute byte 1 *Value* *Label* *Description* X'01' $BCASM Assembler DTF X'04' $BCASK On–ASCII file Off–EBCDIC file X'08' $BCGET GET file X'10' $BCRAN Transparent mode X'20' $BCITB ITB mode X'40' $BCOUT Output file X'80' $BCINP Input file X'C0' $BCCNV Conversational file
3	3	$BDATR	1	Attribute byte 2 *Value* *Label* *Description* X'01' $BCOPN File opened X'02' $BCACT File active X'04' $BCUSD File used X'08' $BCSWI Switched line X'10' $BCANS Answer line X'20' $BCMAN Manual line X'80' $BCMPT Tributary station X'88' $BCMCN Control station
5	5	$BDCHN	2	Chaining pointer to post open DTFs
7	7	$BDNXT	2	Chaining pointer to next DTF, preopen or post open
9	9	$BDWK1	2	Work area for MLMP routines
B	11	$BDWK2	2	Work area for MLMP routines
D	13	$BDWKB	2	Address of user's logical buffer

Figure 11 (Part 1 of 4). BSC DTF

Displacement Hex	Displacement Decimal	Label	Length in Bytes	Description
E	14	$BDCMP	1	Completion codes

Value	*Label*	*Description*
X'00'	$BCREQ	Request accepted
X'40'	$BCDNE	Normal completion
X'41'	$BCUER	User error
X'42'	$BCEOT	End of file
X'43'	$BCBID	Invalid ID
X'44'	$BCNEG	Negative response to poll/address
X'45'	$BCNON	No response to poll/address
X'46'	$BCCRP	Conversational reply pending
X'47'	$BCNDT	No data for conversational GET (null message received)
X'48'	$BCOLT	Invalid $RFT (request for online test)
X'49'	$BCNAC	No active entry in polling list
X'4A'	$BCIGN	Request ignored
X'4B'	$BCASC	Invalid ASCII character
X'4C'	$BCNCN	No connection
X'4D'	$BCCAL	Invalid request
X'4E'	$BCLST	Delay count (DLYCT in $DTFB) exceeded
X'4F'	$BCERR	Permanent error
X'50'	$BCTIM	No response from remote device
X'51'	$BCDAT	Data check
X'52'	$BCLOS	Lost data or no RECSEP character within 2 contiguous blocks of spanned records
X'53'	$BCCON	Lost connection or DISC received
X'54'	$BCRSP	Invalid response received
X'55'	$BCADP	Adapter check
X'56'	$BCCMP	No completion in check list
X'57'	$BCACD	No active DTFs in check list
X'58'	$BCRLE	Maximum record length exceeded

F	15	$BDOPC	1	Operation codes

Value	*Label*	*Description*
X'40'	$BOPUT	PUT
X'41'	$BOPEB	PUT end-of-block
X'42'	$BOPEF	PUT end-of-file
X'44'	$BOPEW	PUT EOT to WACK response
X'80'	$BOGET	GET
X'81'	$BOGBK	GET a block

11	17	$BDMRL	2	Maximum record length

Figure 11 (Part 2 of 4). BSC DTF

Displacement Hex	Displacement Decimal	Label	Length in Bytes	Description
12	18	$BDADD	1	Attribute byte *Value* *Label* *Description* X'01' $BCAA1 Add on area on DTF X'02' $BCPOL Polling modules resident X'04' $BCOFL Truncate record indicator X'08' $BCRCL Span indicator for record length X'10' $BCTWO End of block indicator X'20' $BCOPD File has been opened X'80' $BCSWD Switched ID list in use
14	20	$BDDCH	2	Address of dial number if this is a switched line, or
		$BDPSC	2	Polling/addressing characters if this is for a tributary station, or
		$BDLST	2	Address of polling/addressing list if this is for a control station
15	21	$BDDCC	1	Length of dial number if this is for a switched line
		$BDIND	1	Polling/addressing list entry ID if this is for a control station
17	23	$BDRID	2	Address of receive ID field or address of switched ID list if this is for a switched line
		$BDCNT	2	List count (number of times to go through a polling list when all responses are negative) if this is for a control station
18	24	$BDRLN	1	Receive ID field length or switched ID list selection characters
		$BDLID	1	Last polling/address ID or last polling/addressing function (X'F0' or X'F1')
1A	26	$BDSID	2	Address of send ID field
1B	27	$BDSLN	1	Length of send ID field
1D	29	$BDDLY	2	Delay count (DLYCT in $DTFB)
1F	31	$BDREL	2	Record length
21	33	$BDBKL	2	Block length
23	35	$BDIOB	2	Address of IOB in process
25	37	$BDBKX	2	Pointer to data in BSCA buffer
27	39	$BDITB	2	ITB character count
2A	42	$BDPRM	3	Reserved

Figure 11 (Part 3 of 4). BSC DTF

Displacement Hex	Displacement Decimal	Label	Length in Bytes	Description
2D	45	$BDRVI	3	RVI (reverse interrupt request) mask and displacement. First byte is mask; next two bytes are address. Address must be valid. Mask must be zero if not used.
2E	46	$BDNDX	1	Index for line initialization
30	48	$BDWKA	2	Address of BSC work area
32	50	$BDINT	2	Disk address of line initialization module
33	51	$BDDED	1	Work area for MLMP routines
34	52	$BDAT1	1	Terminal attribute byte *Value* *Label* *Description* X'01' $BCSEP Record separator used X'02' $BCSPN Spanned records used X'04' $BCNOW Spanning in process X'08' $BCPUT PUT span file X'10' $BCRES Restore after spanning X'40' $BCPLR Polling resident
35	53	$BDSEP	1	Record separator character
37	55	$BDSBF	2	Save area for user's logical buffer address
39	57	$BDSRL	2	Save area for record length
3B	59	$BDRFT	2	Save area for address of online test parameter list
3D	61	$BDTSA	2	Address of terminal statistics logging area
				The following are in the DTF only if core resident polling, auto response, or user error retry count is present:
3F	63	$BDRL0	2	Address of $$BSMA (For resident polling (POLRES–Y in $DTFB))
41	65	$BDRCL	2	Address of $$BSMC (For resident polling (POLRES–Y in $DTFB))
43	67	$BDARA	2	Address of $$BSMD (For resident polling (POLRES–Y in $DTFB))
44	68	$BDERR	1	Error retry count
46	70	$BDT1A	2	Disk address of online test module
48	72	$BDEX@	2	CCP user exit address (CCP only)
49	73	—	3	Reserved (This field is present only if AUTORS–Y, POLRES–Y, or ERRCT–decdig was specified in $DTFB)

Figure 11 (Part 4 of 4). BSC DTF

MLMP I/O Area

The MLMP I/O area for a file is defined by the BUFST and BUFEND operands of $DTFB, or by the BUFNO operand of $DTFB. When the file is opened, MLMP formats the allocated I/O area into as many IOBs and buffers as the area can contain (see Figure 12).

Note: Conversational files (CONV–Y in $DTFB) are permitted only one IOB and buffer.

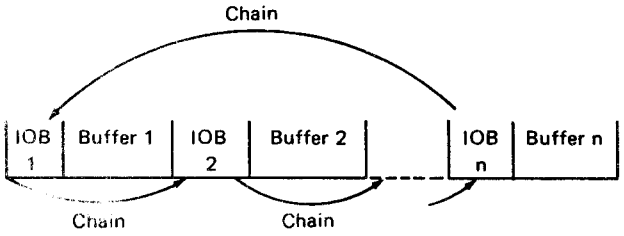


Figure 12. MLMP Multiple Buffers

The BSC IOB is 21 bytes long. The IOB controls the flow of data to and from the associated buffer. Each buffer contains one block of data as described in $DTFB (see Figure 13).

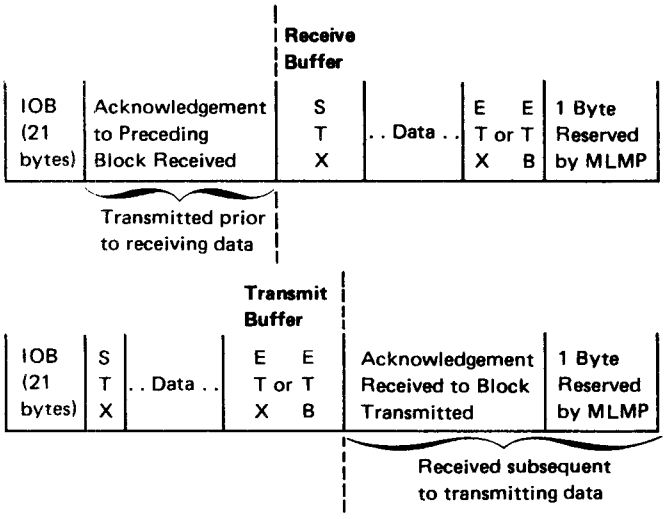


Figure 13. Sample MLMP Buffers

The length of each buffer, including the IOB, is calculated as follows:

- Conversational mode:

  Buffer length = (record length x 2) + 28

- Non-conversational mode:

  Buffer length = (record length x blocking factors) + C
  + number of characters needed for ITB

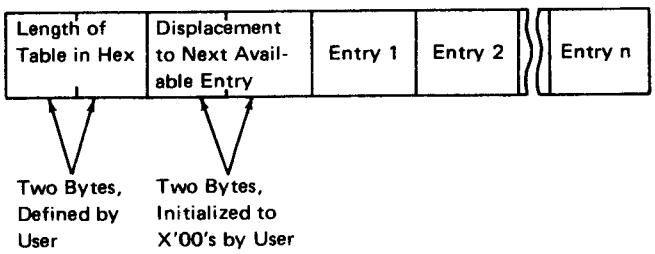
  Blocking factor = number of records per block
  C = 44 for ITB transparent PUT; 42 in all other cases
  Number of characters needed for ITB = (blocking factor - 1) x ITB count
  ITB count = 0 for non-ITB
  1 for ITB non-transparent
  3 for ITB transparent GET
  5 for ITB transparent PUT

If you want multiple buffers, double the computed buffer length to have double buffers, triple the computed buffer length to have triple buffers, etc. Multiple buffering uses a telecommunications line more efficiently than single buffering: you can be moving data to or from one buffer while data is being transmitted from or to another.

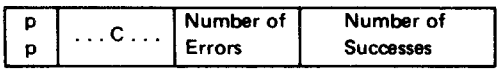
Note: A conversational file (CONV–Y in $DTFB) cannot have multiple buffers.

Terminal Statistics Logging Area

The Terminal Statistics Logging Area is a table in main storage used by the terminal statistics logging routine to record terminal statistics for multipoint control stations. The table is addressed by the ERRLOG operand (required if you specify TYPE–CS) of $DTFB. You can use the $LOGB macro instruction to allocate the Terminal Statistics Logging Area. The format of the area is:



Each entry in the table is formatted by the terminal statistics logging routine:



pp = Number of polling or addressing characters (1 byte).

C = Polling or addressing characters, as many as 7.

Errors = Transmissions containing I/O errors (2 bytes).

Successes = Successful transmissions (4 bytes).

Space in main storage must be available for the table. The number of bytes required by the table (and entered as the length of the table in the first two bytes) is: 4 + p(7 + n) where:

p = Number of unique sets of polling and addressing characters referenced in this program by this DTF. (If one terminal has both a set of polling characters and a set of addressing characters, two entries are required for the terminal.)

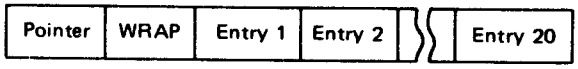
n = Maximum number of polling or addressing characters per entry.

If you define space in main storage for the Terminal Statistics Logging Area but you don't use $LOGB, specify the length of the entire table in the first two bytes of the table, and initialize the rest of the table to X'00'.

Note: If the area you define is not large enough to contain all the terminal statistics, MLMP will issue the Y6 halt. For a complete description of the Y6 halt, see the appropriate halt/messages manual listed in the *Preface.*

Trace Table

The trace table contains I/O information recorded by the trace module ($$BSTT). The trace table is 323 bytes long, beginning at symbolic address MTBSML and ending at symbolic address MTBSMM. The format of the trace table is:



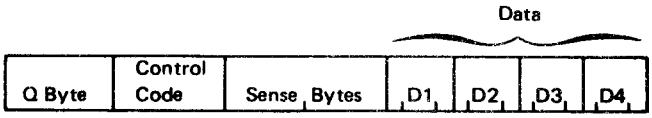
Pointer = Address of the first byte of the last entry in the table used by Trace (2 bytes).

WRAP = Status byte:

X'00'—No more than 20 entries have been written to the the table.

X'01'—Each entry has been filled at least once, and entries are now being overlaid, beginning with entry entry 1.

Entry = 16 bytes. The format of each entry is:



Q Byte—From the BSCA SIO instruction initiating the event recorded.

Control Code—From the SIO instruction initiating the event recorded; 1 byte.

Sense/Status Bytes:

Byte	Bit	Meaning When Set to 1
1	0	Timeout status. a. A receive timeout occurred during a receive operation with the adapter in the busy state. b. An autocall operation was terminated by an abandon call and retry signal from the autocalling unit, (ACU), indicating that a connection was not established.
1	1	Data check during receive operation. a. A BCC compare check occurred (EBCDIC). b. A VRC check occurred (ASCII). *(Note:* Characters having VRC checks are distinguished by a high-order bit in core storage. These characters are never recognized as control characters by the BSCA.)
1	2	Adapter check during transmit operation. a. DBI register parity check. b. I/O cycle steal overrun.

Byte	Bit	Meaning When Set to 1
		c. LSR or shift register parity check. d. Transmit control register check.
1	3	Adapter check during receive operation. a. DBI register parity check. b. I/O cycle steal overrun. c. LSR or shift register parity check.
1	4	Invalid ASCII character. (A byte fetched from core by an adapter using ASCII code contained a 1-bit in the high order bit position.)
1	5	Abortive disconnect. Indicates BSCA on switched network was enabled, then the data set became ready, then not ready. This indicates the connection has been released and causes data terminal ready to turn off.
1	6	Disconnect timeout. Indicates disconnect timeout occurred on a switched network. Disconnect timeout causes data terminal ready to turn off. (May not apply to systems using the IBM remote job entry program.) *Note:* The program must perform a disconnect operation.
1	7	Not assigned.
2	0	Not assigned.
2	1	
2	2	
2	3	
2	4	
2	5	
2	6	Data set ready. This indicates that the data set is ready to operate and that the BSCA has been enabled.
2	7	Data line occupied. This bit is used on a switched network when the BSCA is equipped with the autocall feature. This bit indicates that the data receive initial instruction will be rejected.

Note: Byte 1 equals leftmost byte; byte 2 equals rightmost byte.

Data =

D1—Contents, at the time the I/O operation was started, of the byte addressed by the current address register (CAR) and the two bytes that follow.

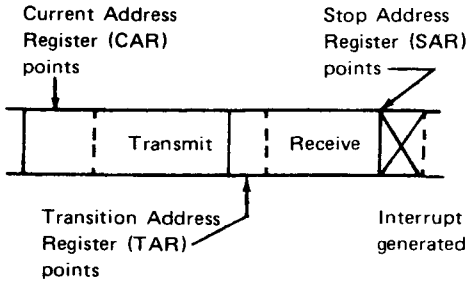
D2—Contents, at the time the I/O operation was started, of the three bytes preceding the byte addressed by the transition address register (TAR).

D3—Contents, at the time the I/O operation was completed, of byte addressed by the TAR and the two bytes follow.

D4—Contents, at the time the I/O operation was completed, of the three bytes preceding the byte addressed by the CAR.

Note: When a 2-second timeout occurs, D1-D4 are set to X'FF's. When a receive timeout occurs, D3 and D4 are set to X'FF'. When the I/O operation is receive-initial (RCVI), receive only (RCVO), or autocall, D2 and D3 are set to X'FF'.

BSC I/O Registers



Transmission proceeds until CAR equals TAR, when receive mode is entered. An interrupt is generated and the operation is terminated if CAR equals SAR, if a line turnaround sequence is received, or if a timeout occurs.

For a complete description of the BSCA instruction set and the BSCA registers, see the appropriate components reference manual listed in the *Preface.*

Checklist

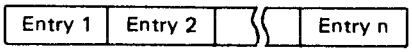
The format of the checklist entries created by the $CKL macro instruction is:

Byte 1 = Status byte:
- X'80'— Skip this entry
- X'40'— This is a printer-keyboard request DTF
- X'20'— This is the last entry in the checklist

X'10'– Return control to user if no completed events (significant in the first checklist entry only)
X'04'– Program function key not available

Bytes 2-3 = Symbolic address of the DTF for this entry

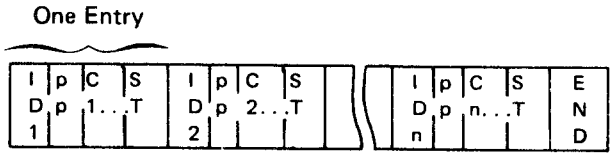
The format of the checklist is:



See also index entry *generate a checklist ($CKL).*

Polling/Addressing List

The format of the entries generated by the $POLB macro instruction is:



IDn = Terminal identification, X'00' through X'EF'

pp = Number of polling or addressing characters

Cn = Polling or addressing characters, as many as 7

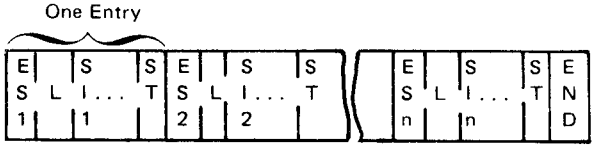
ST = Status byte:
X'80'–Inactive entry
X'00'–Active entry

END = End byte:
X'FE'–Open list
X'FF'–Wrap list

See also index entry *generate a polling or addressing list ($POLB).*

Switched ID List

The format of the entries generated by the $SWIB macro instruction is:



ESn = Entry selection characters, X'00' through X'EF'

L = Number of station identification characters

SIn = Station identification characters, as many as 15

ST = Status byte:
X'80'–Inactive entry
X'00'–Active entry

END = End byte: X'FF'

Parameter List for Changing a Polling List or Switched ID List

The format of the parameter list generated by the $CHGB macro instruction is:

Type

Bit 0: Off–activate selected entries
On–deactivate selected entries

Bit 1: Off–select only those entries whose characters match the characters specified in the Chars field (below)
On–select only those entries whose first N characters match the first N characters specified in the Chars field (below), where N is the number specified in the N byte (below)

DTF Address: Address of the DTF whose polling list or switched ID list is to be changed.

N: Hexadecimal number of characters in the Chars field (below) to be compared to characters in polling list or switched ID list entries.

Chars: Hexadecimal representation of characters to be compared to characters in polling list or switched ID list entries.

See also index entries *$BCPL macro instruction* and *$CHGB macro instruction.*

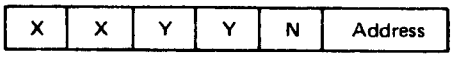
Translate Parameter List

The translate parameter list is used by the $TRAN macro instruction to translate data. See also index entry *generate an interface to the translate routine ($TRAN).*

Field Length	Field Description
2	Address of translate table generated by the $TRTB macro instruction (see index entry *$TRTB macro instruction)*
2	FROM field for translation
2	TO field for translation
2	Number of bytes to translate
1	Return Code X'00'—translation complete, no errors X'FF'—invalid character detected

Online Test Parameter List

The format of the online test parameter list required by the $RFT macro instruction is:



XX = Decimal number specifying test type. Of the following test types, System/3 can only request types 00, 01, 06, and 14. (When requesting an online test, be sure to specify a test the remote terminal can accept.) System/3 can accept all of the following test types. (Types 23 through 34 are only accepted from a 3270.)

Test Type	Description
00	Receive and acknowledge the test message the number of times specified in bytes YY. The formatted test request must not be more than 300 characters long. See index entry *online test requests.*
01	Transmit the test message the number of times specified in bytes YY. The formatted test request must not be more than 300 characters long. See index entry *online test requests.*
06	Transmit 36 alphameric characters, A-Z and 0-9, the number of times specified in bytes YY. Transmit the characters in ASCII (ASCII adapter only).
14	Transmit 36 alphameric characters, A-Z and 0-9, the number of times specified in bytes YY. Transmit the characters in EBCDIC (EBCDIC adapter only).
23	3270 basic EBCDIC test message: This test checks all alphameric characters at a display station or printer. It also checks the use of the WCC to sound the audible alarm and allows attribute field specifications to be checked at a display station. It starts a printer, printing only 40 characters to a line.
24	3270 Model 1 align EBCDIC test pattern: This test checks position alignment for the 480-character display station. It also checks the WCC to sound the audible alarm. It starts a printer, printing 40 characters to a line.
25	3270 Model 2 align EBCDIC test pattern: This test checks position alignment for the 1920-character display station. It also checks the WCC to sound the audible alarm. It starts a printer, printing 80 characters to a line.
26	3270 orders EBCDIC test message: Tests 3270 orders (SF, SBA, etc.), checks the WCC to sound the audible alarm, and uses display and intensified brightness. It starts the printer, printing 64 characters to a line.
27	3270 EBCDIC universal character set test pattern: This test uses the erase/write command, displaying the universal character set in EBCDIC. It checks the WCC to start the printer, sounds the audible alarm (on a display), and prints 132 characters per line on the printer. NL and EM are also tested on a printer. Display intensity is used. The SF, NL, EM, and IC orders are used.

Test Type	Description
28	3270 NL/EM EBCDIC test pattern: This test is mainly intended to test the end of message (EM) order and multiple new line (NL) orders on the printer. The WCC is checked to start the printer, sound the alarm (on a display), and print 132 characters to a line on the printer.
29-34	3270 ASCII test patterns: These tests correspond to tests 23-28 except that transmission is in ASCII.

Y Y = Decimal number specifying the number of times to transmit or receive the test message.

N = Decimal number specifying the number of addressing characters (0-7). N equals 0 except when a tributary station requests a test other than type 00.

Address = Addressing characters to be used, not more than 7.

Note: Decimal numbers and addressing characters must be given in the code used by the BSCA (EBCDIC or ASCII).

The message type specified determines the buffer space required to transmit or receive an online test request. In all cases, block length specified for System/3 (BLKL in $DTFB) must be equal to or greater than the test message length (LEN in $RFT), whether System/3 is sending or receiving the online test request.

Where m = test message length, including framing characters (see index entry *online test requests*), and
n = number of addressing characters,
the minimum BLKL required to send or receive an online test request is:

Message type 00

BLKL = 9 or m, whichever is greater

Message type 01

Point-to-point nonswitched and switched lines:

BLKL = 7 + m

Multipoint lines:

Tributary station:
BLKL = 7 + m + n

Control station:
BLKL = 11 + m + 2n

Message types 06 and 14

BLKL must be greater than or equal to 40.

Message types 23-34

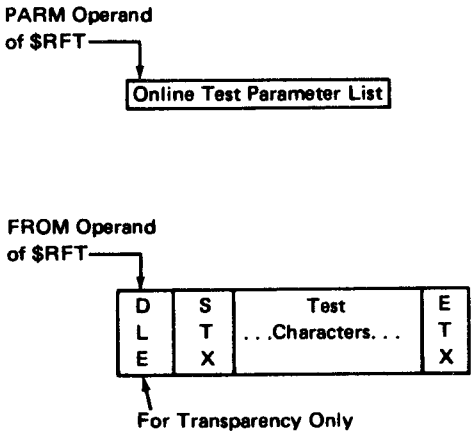
BLKL must be greater than or equal to 300.

Online Test Requests

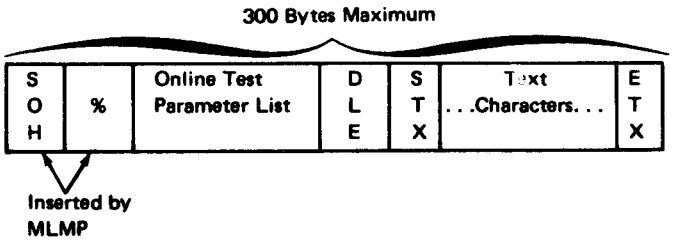
Online test requests are written as follows:

- Test type 01

Given the online test parameter list and a test message,

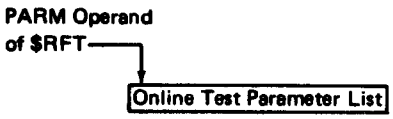


MLMP constructs and transmits the following online test request:

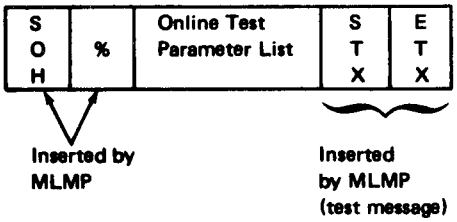


- All test types other than 01

  Given the online test parameter list,



MLMP constructs and transmits the following online test request:

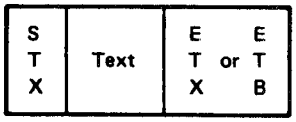


Note: The parameter list and the test message must both be given in the appropriate line code, EBCDIC or ASCII (translation may be required—see index entries *generate a translate parameter list ($TRL)* and *generate an interface to the translate routine ($TRAN)*).

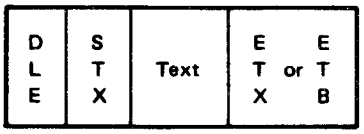
MLMP Message Formats

MLMP builds and recognizes the following message formats:

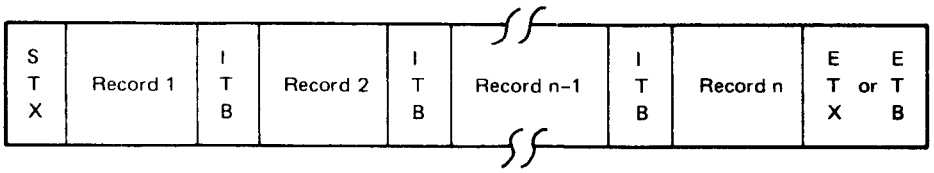
- Non-ITB, non-transparent



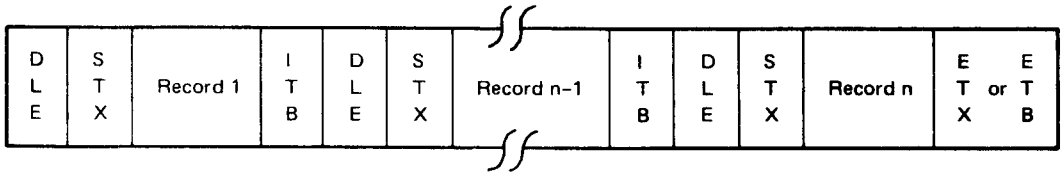
- Non-ITB, text transparency



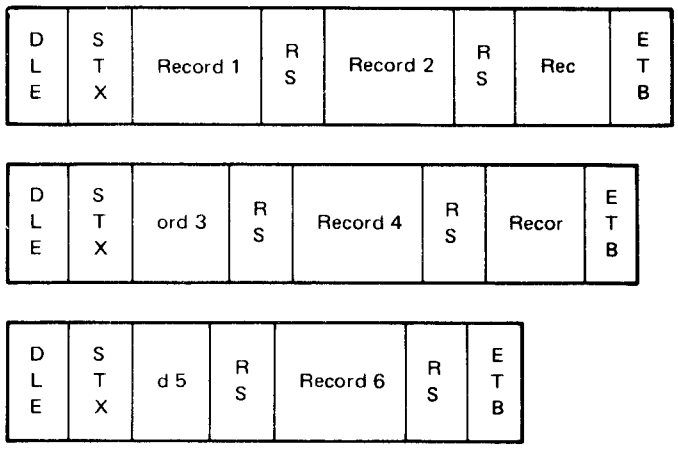
- ITB, non-transparent



- ITB, text transparency



Spanned records can be used with or without ITB, and can be transparent or non-transparent. The following format shows spanned records--non-ITB, text transparency:



RS = Record separator

EBCDIC

Main Storage Bit Positions 4, 5, 6, 7		Main Storage Bit Positions 0, 1, 2, 3															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	DLE	DS		SP	&	-						{	}	\	0
0001	1	SOH	DC1	SOS				/		a	j	~		A	J		1
0010	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	3	ETX	DC3							c	l	t		C	L	T	3
0100	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	5	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	6	LC	BS	EOB / ETB	UC					f	o	w		F	O	W	6
0111	7	DEL	IL	PRE / ESC	EOT					g	p	x		G	P	X	7
1000	8		CAN							h	q	y		H	Q	Y	8
1001	9	RLF	EM						\	i	r	z		I	R	Z	9
1010	A	SMM	CC	SM		¢	!	¦	:								
1011	B	VT				.	$	,	#								
1100	C	FF	IFS		DC4	<	*	%	@								
1101	D	CR	IGS	ENQ	NAK	(	)	_	'								
1110	E	SO	IRS	ACK		+	;	>	=								
1111	F	SI	IUS	BEL	SUB	¦	¬	?	"								

Duplicate Assignment

ASCII

Main Storage Bit Positions 4, 5, 6, 7		Main Storage Bit Positions 0, 1, 2, 3															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	DLE	SP	0	@	P	`	p								
0001	1	SOH	DC1	!	1	A	Q	a	q								
0010	2	STX	DC2	"	2	B	R	b	r								
0011	3	ETX	DC3	#	3	C	S	c	s								
0100	4	EOT	DC4	$	4	D	T	d	t								
0101	5	ENQ	NAK	%	5	E	U	e	u								
0110	6	ACK	SYN	&	6	F	V	f	v								
0111	7	BEL	ETB	'	7	G	W	g	w								
1000	8	BS	CAN	(	8	H	X	h	x								
1001	9	HT	EM	)	9	I	Y	i	y								
1010	A	LF	SUB	*	:	J	Z	j	z								
1011	B	VT	ESC	+	;	K	[	k	{								
1100	C	FF	FS	,	<	L	\	l	|							
1101	D	CR	GS	-	=	M	]	m	}								
1110	E	SO	RS	.	>	N	¬	n	~								
1111	F	SI	US	/	?	O	_	o	DEL								

Hexadecimal Representations

BSC control characters are represented by the following values:

	EBCDIC	*ASCII*
ACK0	X'1070'	X'1030'
ACK1	X'1061'	X'1031'
DISC	X'1037'	X'1004'
DLE	X'10'	X'10'
ENQ	X'2D'	X'05'
EOT	X'37'	X'04'
ETB	X'26'	X'17'
ETX	X'03'	X'03'
ITB	X'1F'	X'1F'
NAK	X'3D'	X'15'
RVI	X'107C'	X'103C'
SOH	X'01'	X'01'
STX	X'02'	X'02'
SYN	X'32'	X'16'
TTD	X'022D'	X'0205'
WACK	X'106B'	X'103B'

Tributary System/3 Polling and Addressing Characters

Polling and addressing characters must be used in pairs: that is, once a polling character is selected, the complementary addressing character is determined; once an addressing character is selected, the complementary polling character is determined.

EBCDIC

Polling Character	Hexadecimal Representation	Addressing Character	Hexadecimal Representation
BB	C2C2	SS	E2E2
CC	C3C3	TT	E3E3
DD	C4C4	UU	E4E4
EE	C5C5	VV	E5E5
FF	C6C6	WW	E6E6
GG	C7C7	XX	E7E7
HH	C8C8	YY	E8E8
II	C9C9	ZZ	E9E9
JJ	D1D1	11	F1F1
KK	D2D2	22	F2F2
LL	D3D3	33	F3F3
MM	D4D4	44	F4F4
NN	D5D5	55	F5F5
OO	D6D6	66	F6F6
PP	D7D7	77	F7F7
QQ	D8D8	88	F8F8
RR	D9D9	99	F9F9

ASCII

Polling Character	Hexadecimal Representation	Addressing Character	Hexadecimal Representation
AA	4141	aa	6161
BB	4242	bb	6262
CC	4343	cc	6363
DD	4444	dd	6464
EE	4545	ee	6565
FF	4646	ff	6666
GG	4747	gg	6767
HH	4848	hh	6868
II	4949	ii	6969
JJ	4A4A	jj	6A6A
KK	4B4B	kk	6B6B
LL	4C4C	ll	6C6C
MM	4D4D	mm	6D6D
NN	4E4E	nn	6E6E
OO	4F4F	oo	6F6F
PP	5050	pp	7070
QQ	5151	qq	7171
RR	5252	rr	7272
SS	5353	ss	7373
TT	5454	tt	7474
UU	5555	uu	7575
VV	5656	vv	7676
WW	5757	ww	7777
XX	5858	xx	7878
YY	5959	yy	7979
ZZ	5A5A	zz	7A7A

Macro Instruction	Format	Function	Approximate Number of Instructions Generated, Excluding Mnotes	Approximate Number of Bytes Required, Excluding Mnotes
$ALOC	[name] $ALOC [DTF–name]	Allocates files.	3	9
$BCPL	[name] $BCPL [PARM–address]	Changes entries in polling lists.	39	109
$BCSW	[name] $BCSW [PARM–address]	Changes entries in switched ID lists.	43	113
$CANB	[name] $CANB [DTF–address]	Cancels an initial GET request.	25	82
$CHGB	[name] $CHGB TYPE–AM/AN/DM/DN,DTF–address, NUM–hex,CHARS–hex	Generates a parameter list for changing polling or switched ID lists.	(6 lines)	6-12
$CHK	[name] $CHK [CKL–address]	Checks for I/O completion.	2	8
$CKL (Model 10)	[name] $CKL DTF–address [,SKIP–Y/N] [,REQK–Y/N or ,CONS–Y/N] [,RTN–Y/N] [,LAST–Y/N]	Creates entries in the check list.	(3 lines)	3
$CKL (Model 15)	[name] $CKL DTF–address [,SKIP–Y/N] [,REQK–Y/N] [,RTN–Y/N] [,LAST–Y/N]	Creates entries in the check list.	(3 lines)	3
$CLOS	[name] $CLOS [DTF–address]	Closes files.	3	9
$COMN	$COMN	Generates common labels.	(14 lines)	0
$DTFB	[name] $DTFB RECL–decdig,BLKL–decdig, RCAD–address,FTYP–RCV/TSM [,BUFST–address,BUFEND–address] [,BUFNO–decdig] [,CODE–E/A] [,LINE–1/2] [,UP–binary/0] [,CHN-name] [,CONV–Y/N] [,ITB–Y/N] [,TRANSP–Y/N] [,RVIADR–address,RVIMSK–hex] [,DLYCT–decdig] [,TYPE–PP/MP/CS/AC/MC/AA/MA] [,TERMAD–hex] [,AUTORS–Y/N] [,LISTAD–address,ERRLOG–address] [,POLRES–Y/N] [,LIMIT–decdig] [,DIAL–address,DIALCT–decdig] [,RCVID–address,RCVCT–decdig/ ,SWLIST–Y/N] [,SNDID–address,SNDCT–decdig] [,SPAN–Y/N] [,RECSEP–hex] [,ERRCT–decdig]	Define BSC DTFs.	(31 lines)	62

Figure 14 (Part 1 of 2). Macro Instruction Summary Chart

Macro Instruction	Format	Function	Approximate Number of Instructions Generated, Excluding Mnotes	Approximate Number of Bytes Required, Excluding Mnotes
$DTOB	$DTOB	Generates displacements and labels for BSC DTFs.	(137 lines)	0
$GETB	[name] $GETB [DTF–address] [,REJECT–address] [,OPC–N/BLK]	Instructs Model 10 BSC to receive (GET) data.	31	94
$LOGB	[name] $LOGB NUM–decdig,LEN–decdig	Allocates the Terminal Statistics Logging Area.	(5 lines)	Depends on the number of terminals and number of polling/addressing characters.
$OPEN	[name] $OPEN [DTF–name]	Opens files.	3	9
$POLB	[name] $POLB ID–hex,TERMAD–hex, LEN–decdig [,LAST–N/OPEN/WRAP]	Creates entries in polling and addressing lists.	(5 lines)	4 (plus poll/addressing characters)
$PUTB	[name] $PUTB [DTF–address] [,REJECT–address] [,OPC–N/EOB/EOF/EOW]	Instructs Model 10 BSC to transmit (PUT) data.	26	77
$RFT	[name] $RFT PARM–address [,FROM–address] [,LEN–decdig] [,DTF–address] [,REJECT–address]	Instructs Model 10 BSC to perform an online test.	25	86
$RFTL	[name] $RFTL TYPE–00/01/06/14, NUM–decdig,LEN–decdig [,CODE–E/A] [,TERMAD–hex]	Generates the online test parameter list.	(5-6 lines)	7-12
$SNAP	[name] $SNAP ID–hex,START–address, END–address	Dumps main storage.	(4 lines)	10
$SWIB	[name] $SWIB SELECT–hex,STATID–hex, LEN–decdig [,LAST–Y/N]	Creates entries in a switched ID list.	(3-6 lines)	3-18
$TRAN	[name] $TRAN [TRL–address]	Translates data.	8	8
$TRL	[name] $TRL TO–address,FROM–address, LEN–decdig,TRT–address	Creates a translate list.	(13 lines)	9
$TRTB	[name] $TRTB [CODE–E/A] [,HEX-hex]	Generates a translate table.	If CODE–A, 12 lines; if CODE–E, 20 lines.	If CODE–A, 130 bytes; if CODE–E, 258 bytes.

Figure 14 (Part 2 of 2). Macro Instruction Summary Chart

Technical Newsletter

This Newsletter No. GN21-5587
Date 25 November 1977

Base Publication No. GC21-7573-4
File No. S3-30

Previous Newsletters None

IBM System/3
Multiline/Multipoint
Binary Synchronous Communications
Reference Manual

This technical newsletter, a part of version 02, modification 00 of the IBM System/3 Model 15 SCP (Program Number 5704-SC2), also applicable to version 06, modification 00 of the IBM System/3 Model 15 SCP (Program Number 5704-SC1), provides replacement pages for the subject publication. These replacement pages remain effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

iii, iv	57, 58
11, 12	87 through 104
53, 54	

Changes to text and illustrations are indicated by a vertical line at the left of the change except for the updated sample program.

Summary of Amendments

Updated sample program and miscellaneous technical corrections.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.

Technical Newsletter

This Newsletter No.	GN21-5691
Date	21 December 1979
Base Publication No.	GC21-7573-4
File No.	S3-30
Previous Newsletters	None

IBM System/3
Multiline/Multipoint
Binary Synchronous Communications
Reference Manual

This technical newsletter applies to the current versions and modifications of the applicable System/3 programs listed in the edition notice and provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

Pages 17 and 18 of this reference manual have been transposed. The pages in question are enclosed in the correct sequence.

Summary of Amendments

Correct out-of-sequence pages

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.