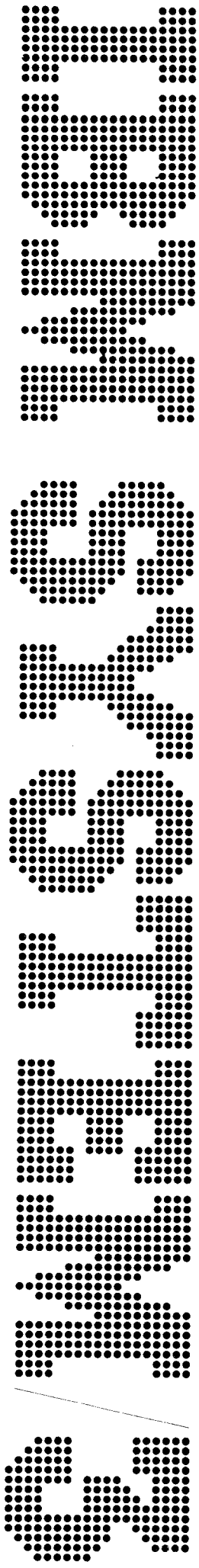


IBM System/3 Model 15
System Control Programming Macros
Reference Manual

GC21-7608-2
File No. S3-36

Preface

This manual describes the macro instructions provided by the IBM System/3 Model 15. The publication is intended for persons who are programming in the Basic Assembler Language or its equivalent and who are familiar with the concept of macro instructions and system programming for the IBM System/3 Model 15.

The following topics are discussed in this publication:

- Coding macro instructions.
- Descriptions of the various macro instructions
- OCL necessary to call the macro processor.
- Error conditions detected by the macro processor.
- A sample program showing how macro instructions are used
- Descriptions of the pre-open and post-open DTF (Define the File) Control Blocks for unit record and disk devices.

Related Publications

These publications contain information which further describes topics discussed in this manual:

- *IBM System/3 Basic Assembler Reference Manual*, SC21-7509
- *IBM System/3 Model 15 System Control Programming Reference Manual* for Program Number 5704-SC1, GC21-5077
- *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual* for Program Number 5704-SC2, GC21-5162

Third Edition (December 1976)

This is a major revision of, and obsoletes, GC21-7608-1 and Technical Newsletters GN21-5353 and GN21-5432. Information has been added to support SCP 5704-SC2 as well as SCP 5704-SC1. Changes to text and illustrations are indicated by a vertical line.

This edition applies to version 05, modification 00 of the IBM System/3 Model 15 System Control Program 5704-SC1, to version 01, modification 00 of the IBM System/3 Model 15 System Control Program 5704-SC2, and to all subsequent versions and modifications until otherwise indicated in new editions or technical newsletters.

Changes are periodically made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the *IBM System/3 Bibliography*, GC20-8080, for the editions that are applicable and current.

Use this publication only for the purposes stated in the *Preface*.

Publications are not stocked at the address below. Requests for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Contents

A macro instruction is a source statement that causes generation of a predetermined set of assembler statements each time the macro instruction is used. The Model 15 System Control Program provides macro instructions which perform both system services and input/output device support. By using these macro instructions, you can perform both system and input/output operations with less coding.

Figure 1 is an overview of the operation of the macro processor. The OCL statements used to call the macro processor are explained in *Chapter 3: OCL and Sample Program.*

WRITING MACRO INSTRUCTIONS

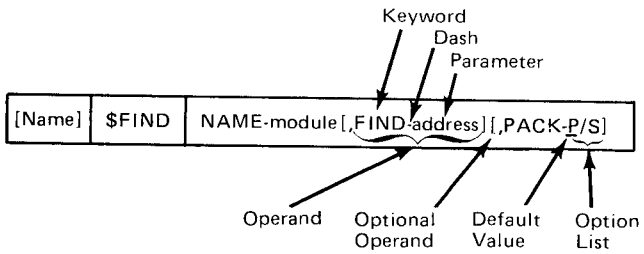
You code macro instructions as follows:

Starting Column 1 | 8 | 14 | 72

Name	Operation	Operands	Continuation
Symbol or blank	Macro name	No operands or one or more separated by commas	Any nonblank character if continuation is being used

The name field can contain any valid assembler language symbolic name beginning in column 1. The name is assigned to the first byte of generated code. Since the name is optional, it is shown enclosed in brackets.

The desired mnemonic operation code (macro instruction name) must appear as specified in the macro instruction description. The operation code must start in column 8.



Operands specify the available services and options. The operands must start in column 14, and are written as follows:

- Each operand consists of a keyword followed by a dash and a parameter.
- Commas separate the operands; no blanks should be left between operands.
- Keywords – those shown in capital letters – are coded exactly as shown. The keyword part of each operand must correspond to one of the keywords in the macro instruction description.
- The parameter part of the operand must immediately follow the dash.
- Parameters – those shown in lowercase letters – indicate information you must supply. Some operands are not required. These optional operands are indicated by enclosing the operand within brackets [KEYWORD-parameter].
- An option list for a keyword parameter is specified as follows:

  KEYWORD-A/B/C

  This list indicates that the keyword has the options A, B, or C. These are the only valid options for the keyword parameter.

  When the options Y/N are given in a macro instruction, Y indicates a yes response, N indicates a no response.

- The operands may be written in any order. If a key word is not specified, the default value is used. A default value is selected for optional keywords that are omitted. The default value is indicated in the macro instruction description by a line under the default option. For example, [KEY-A/B/C] indicates the option A is the default value.

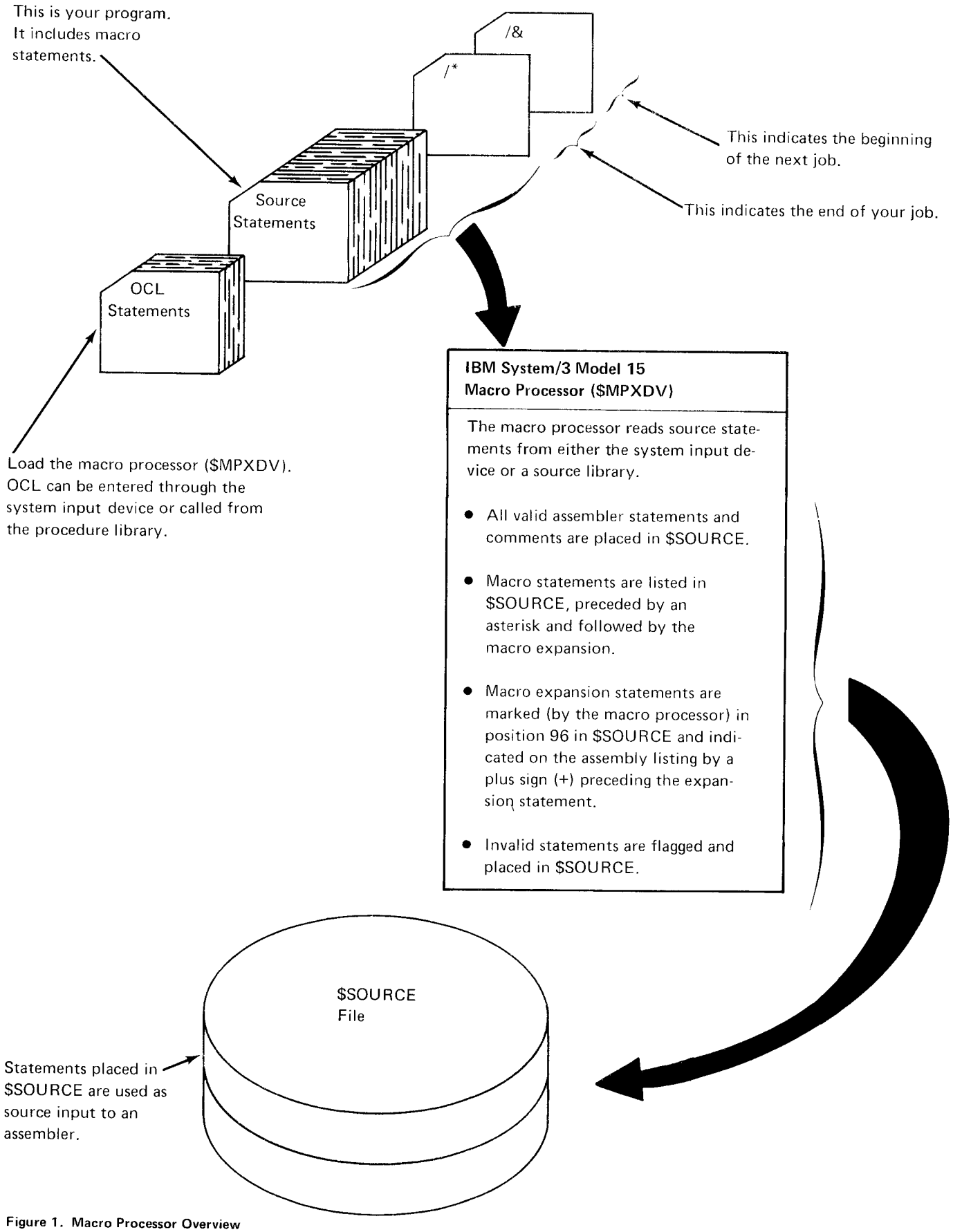


Figure 1. Macro Processor Overview

No operands can be specified beyond column 71. If continuation is required, column 72 must contain a nonblank 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 2.

Comments must be separated from the operand or comma by at least one blank space. Comments cannot be inserted between operands on a one-line macro instruction. Figure 3 shows examples of comments used with macro instructions. On the assembler listing, all comments on the generated code are justified by the macro processor to begin in column 40. Any comments too long to be contained in columns 40 through 71 are truncated from the right.

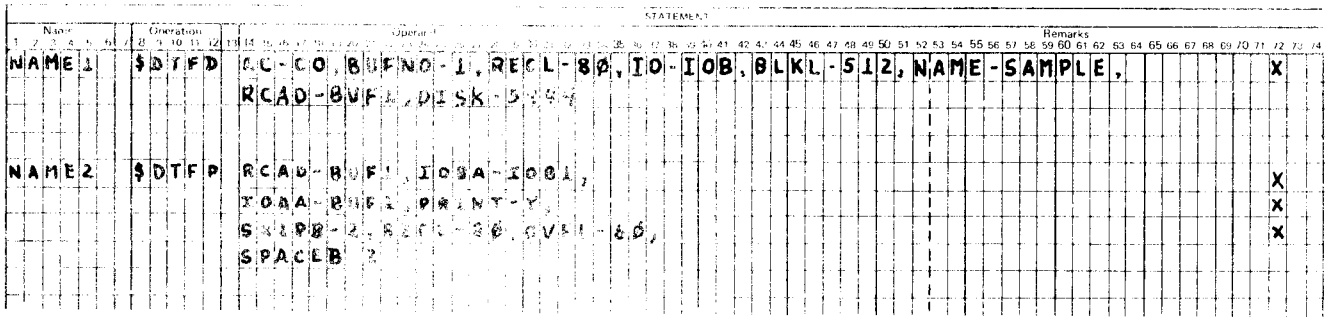


Figure 2 Continuation Coding Examples

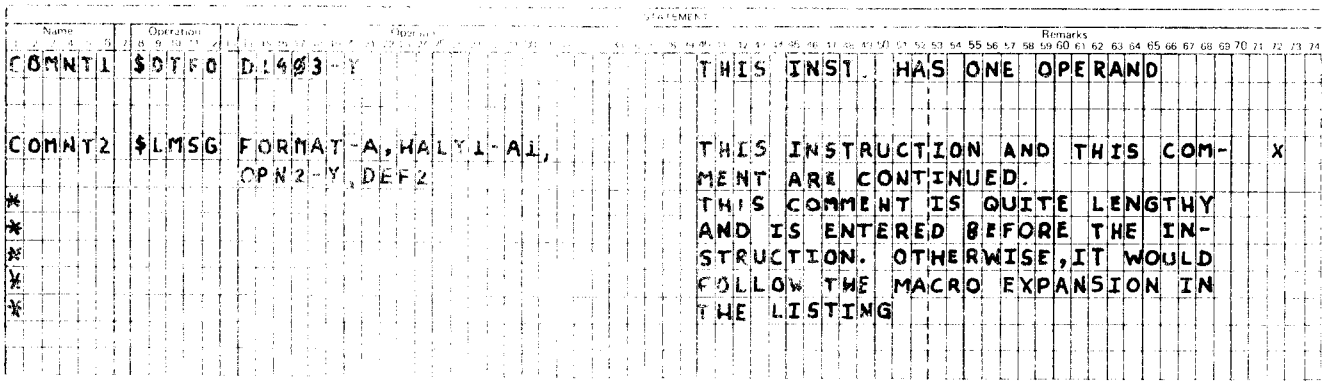


Figure 3 Comments on Macro Instructions

SYSTEM CONFIGURATION

The minimum system configuration for using the Model 15 macros is:

- 5415 Processing Unit with at least 48K bytes of main storage.
- 3277 Display Station Model 1 with Feature 4632. This comprises the IBM System/3 Model 15 CRT/Keyboard, usually referred to in this manual as the CRT/Keyboard.
- Disk Storage Device.
- A system input device.
- 1403 Printer Model 2, 5, or N1.

The following input and output devices are supported:

- 5424 Multi-Function Card Unit (MFCU) Model A1 or A2, 1442 Card Read Punch Model 6 or Model 7, or 2560 Multi-Function Card Machine (MFCM) Model A1 or A2.
- 1403 Printer Model 2, 5, or N1.
- 3277 Display Station Model 1 with Feature Number 4632 (CRT/Keyboard). This device can be used both as the system input device and the system log device.
- 3410/3411 Magnetic Tape Subsystem Model 1, 2, or 3.
- 2501 Card Reader Model A1 or Model A2.
- 3284 Printer
- 3741 Data Station Model 1 or 2 or Programmable Work Station Model 3 or 4.
- Disk Storage Requirements.

	5704			
	SC1			SC2
Minimum	**15A**	**15B**	**15C**	**15D**
5444 Disk Storage Drive, Model A2	X			
3340 Direct Access Storage Facility, Model A2		X	X	X
Optional				
5444 Disk Storage Drive, Model A3	X			
5445 Disk Storage, Models 1, 2, and 3	X			
3340 Direct Access Storage Facility, Models B1 and B2		X	X	X[1]
3344 Direct Access Storage, Model A2				X[1]

[1] Mutually exclusive

The macro processor operates under control of the IBM System/3 Model 15 System Control Program.

MACRO INSTRUCTIONS PROVIDED

The macro instructions provided by the Model 15 System Control Program (SCP) and the functions they perform are shown in Figure 4.

All macros you want to use must be in the source library on the program pack or the system pack. The program pack is the disk pack from which the macro processor is loaded. The system pack is the disk pack from which initial program load (IPL) is performed. Note that the macro processor and IPL code may be on the same pack.

You may want to delete some macro instructions from your library to reduce the amount of disk space required for the macro instructions. For instance, if your system does not include the 3410/3411 Magnetic Tape Subsystem, the tape macro instructions would be of no use to you. You can delete macro instructions from your library by using the library maintenance utility program, $MAINT.

Device Type Supported	Macro Instruction Name	Function
System Reader	$RLST	Generate reader parameter list
	$RLSD	Offsets in reader parameter list
	$READ	Linkage to system reader
System Log	$LWTO	Generate parameter list for WTO or WTOR
	$LMSG	Generate parameter list for halt message on system log
	$LOG	Linkage to system log
	$LOGD	Offsets in log parameter list
General SCP	$ROLL	Rollout/rollin linkage (5704-SC1 Only)
	$FIND	Find a directory entry
	$LOAD	Load a module
	$FTCH	Load a module and pass control
	$XCTL	Load a module and exchange control
	$TRL	Generate a translate parameter list
	$TRTB	Generate a translate table
	$TRAN	Generate an interface to the translate routine
	$SNAP	Snap dump main storage
	$DATE	Retrieve system date
	$TIOB	Generate timer IOB
	$SIT	Set interval timer
	$TOD	Return time of day and system date
	$RIT	Return amount of time left in timer interval
	$EOJ	End of job
General I/O	$ALOC	Allocate disk space or device
	$OPEN	Prepare an I/O device
	$CKL	Generate a checklist
	$CHK	Check for I/O completion for BSCA operations
	$CLOS	Prepare a device for termination
	$DTFO	DTF offsets for all devices
	$COMN	Generate equates
Card	$DTFC	Define the file for a card device
	$GETC	Construct a card GET interface
	$PUTC	Construct a card PUT interface
	$GPC	Construct a GET or a PUT interface to a card file

Figure 4 (Part 1 of 2). Macro Instructions

Device Type Supported	Macro Instruction Name	Function
Printer	$DTFP	Define the file for a printer
	$PUTP	Construct a printer PUT interface
Disk	$DTFD	Define the file for disk
	$GETD	Construct a disk GET interface
	$PUTD	Construct a disk PUT interface
	$IOBD	Generate a disk I/O block
	$IOED	Generate offsets in an I/O block for disk
	$RDD	Read from disk
	$WRTD	Write to disk
	$WAIT	Wait for disk I/O completion
Tape	$DTFT	Define the file for tape
	$GETT	Construct a tape GET interface
	$PUTT	Construct a tape PUT interface
	$RDT	Read from tape
	$WRTT	Write to tape
	$CTLT	Control command for tape
	$WTT	Wait for tape I/O completion
Device Independent	$DTFI	Define a device independent file
	$GETI	Construct a device independent GET interface
	$PUTI	Construct a device independent PUT interface
CRT/Keyboard	$DTFS	Define the file for CRT
	$GETS	Construct a keyboard GET interface
	$PUTS	Construct a CRT PUT interface
	$PGS	Construct a PUT, then a GET request to CRT/Keyboard
	$CQEP	Generate a parameter list for a program function key request
	$PFKY	Program function key request
	$PFKT	Test if program function key pressed
3741	$DTFK	Define the file for 3741
	$GETK	Construct a 3741 GET interface
	$PUTK	Construct a 3741 PUT interface

Figure 4 (Part 2 of 2). Macro Instructions

Chapter 2. Macro Instruction Statements

You code macro instructions to generate a block of assembler statements that perform a certain function. Some functions may be the same each time they are used, others may be modified by specifying different operands. This chapter explains the System/3 Model 15 macro instructions in detail.

The macro instructions are grouped in this chapter according to the functions they perform:

- System services
- Input/output support

Input/output support macro instructions are further divided according to the device supported.

PROGRAMMING CONSIDERATIONS

When you use the macro processor you should remember the following restrictions:

1. The generated code for some macro instructions uses register 1; the generated code for other macro instructions uses register 2. You should save the contents of the register used by the generated code before issuing the macro instruction; otherwise, the contents are destroyed. These macro instructions use register 1:

   $PFKY
   $PFKT
   $RDD
   $TRAN
   $WAIT
   $WRTD

   These macro instructions use register 2:

$ALOC	$GETT	$PUTS
$CHK	$GPC	$PUTT
$CLOS	$LOAD	$RDT
$CTLT	$LOG	$READ
$DATE	$OPEN	$RIT
$FIND	$PGS	$SIT
$FTCH	$PUTC	$TOD
$GETC	$PUTD	$WRTT
$GETD	$PUTI	$WTT
$GETI	$PUTK	$XCTL
$GETK	$PUTP	
$GETS		

2. The code generated by the macros is assigned labels, which begin with the dollar sign ($). To avoid duplicate label errors, you should not use the dollar sign as the first character of a label.

SYSTEM SERVICES MACROS

By using system services macro instructions, you can communicate with the Model 15 system control program.

These macro instructions can do the following:

- Read records from the system input device.
- Log and write error messages.
- Determine the location of an object module on disk.
- Obtain object modules from disk and load them into main storage.
- Pass control to modules in main storage.
- Terminate the current job.

The system services macro instructions are divided into three groups:

1. System reader macro instructions, which provide support and linkage to the system reader function.

   $READ
   $RLSD
   $RLST

2. System log macro instructions, which provide support and linkage to system log functions.

   $LMSG
   $LOG
   $LOGD
   $LWTO

3. General SCP macro instructions, which provide linkage to system functions.

   $DATE
   $EOJ
   $FIND
   $FTCH
   $LOAD
   $ROLL (5704-SC1 only)
   $RIT
   $SIT
   $SNAP
   $TIOB
   $TRAN
   $TRL
   $TRTB
   $TOD
   $XCTL

System Reader Support

You read a record from the system reader by calling the system reader routine through the $READ macro instruction. The system reader may be one of the following:

- CRT/Keyboard. Only 96-byte, single-buffered input is allowed for this device. Double buffering is ignored.
- 2501 Card Reader. Single and double buffering are supported. Only 80 bytes of the 96-byte buffer are used as input; the remaining 16 bytes are cleared to blanks.
- 2560 Multi-Function Card Machine (MFCM). Single and double buffering are supported. Only 80 bytes of the 96-byte buffer are used as input; the remaining 16 bytes are cleared to blanks. Support for both the primary (MFCM) and secondary (MFCM2) hoppers is provided.
- 1442 Card Read Punch. Single and double buffering are supported. Only 80 bytes of the 96-byte buffer are used as input; the remaining 16 bytes are cleared to blanks.
- 5424 Multi-Function Card Unit (MFCU). Both single and double buffering are supported. Support for both the primary (MFCU1) and secondary (MFCU2) hoppers is provided. All 96 bytes are used as input.
- Directly attached 3741 Data Station Model 1 or 2 or Programmable Work Station Model 3 or 4. Single and double buffering are supported. Only 96-byte records may be read.

To call the system reader, you must do the following:

1. Use the $RLST macro to construct a parameter list as input to the system reader routine.
2. Use the $RLSD macro instruction to establish equates for the system reader parameter list.
3. Issue the $READ macro instruction.

The $READ macro generates the code to load the parameter list address into register 2, set the operation code, call the system reader routine, and check the return codes. Because the return code is in the same byte as the operation code, the operation code must be reset before each call. The $RLSD macro is provided to generate the offsets into the parameter list, the values for the operation codes, and the values for the return codes.

Generate a System Reader Parameter List ($RLST)

This macro instruction generates a reader parameter list.

The format of the $RLST macro instruction is:

[Name]	SRLST	BUF1-address,WORK-address[,BUF2-address]

BUF1-address specifies the address of the leftmost byte of a 96-byte buffer that is aligned on a 128-byte boundary. This operand is required.

WORK-address specifies the address of the leftmost byte of a 47-byte work area; this operand is required.

BUF2-address specifies the address of the leftmost byte of a 96-byte buffer that is aligned on a 128-byte boundary. This is the second buffer if double buffering is used; if this operand is not specified, single buffering is assumed.

System Reader Parameter List Offsets ($RLSD)

This macro instruction generates a list of equates used to label the fields in the system reader parameter list. $RLSD also generates the operation codes and return codes used by $READ. To avoid duplicate labels, you should use this macro instruction only once in a program.

The format of the $RLSD macro instruction is:

	$RLSD	

Linkage to System Reader Function ($READ)

This macro instruction generates the linkage to call the system reader function and check for return codes.

The format of the $READ macro instruction is:

[Name]	$READ	[LIST-address] [,OPC-code] [,EOF-address] [,EOJ-address] [,ERR-address]

LIST-address specifies the address of the leftmost byte of the system reader parameter list. If this operand is not specified, the address of the parameter list is assumed to be in register 2.

OPC-code specifies the operation code for this read request. The allowable codes and their meanings are:

Code	Meaning
RD	Set the operation code to read one record from the system reader single buffer.
RDF	Set the operation code to read one record from the system reader into buffer 1 and start a read on buffer 2.
RDD	Set the operation code to wait on buffer 2, switch the buffers, and start the other buffer.
RDL	Set the operation code to wait on buffer 2, and switch the buffers; do not start the other buffer.
N	Do not set any operation code. If this value is specified, it is your responsibility to set the operation code before issuing this macro.

The default value for this operand is RD.

EOF-address specifies the address in your program that receives control when an end-of-file statement (/*, /&, or /.) is detected. If this operand is not supplied, no code is generated to check for the end-of-file condition.

EOJ-address specifies the address of the routine that should get control if an end-of-job or end-of-step statement (/& or /.) was detected on the previous read. If this operand is not specified, the test for the return code is not generated. Once a /& statement is read from the system reader, nothing can be read from the system reader until the end of step. Once a /. statement is read from the system reader, nothing can be read from the system reader until the end of job.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code.

Notes:

1. If ERR or EOF addresses are not specified, you should check the return code in your program to determine the outcome of the operation.
2. When double buffering is used, return code should be provided to return to the read macro instruction until end-of-file (EOF) is received. When the program's EOF is dependent upon a user-defined EOF statement, the last request to the read macro should be a wait only call; this will allow all outstanding read requests to be cleared from the device queues. The last wait request is necessary to prevent overlaying the system input work area before the program reaches end of job.

System Log Support

Specifying a $LOG macro instruction in your program generates a call to system log (system log is a group of system output routines which provide communication with the operator). You may want to use system log to notify the operator of error conditions, error recovery procedures, and the validity of previous operator responses to messages. If the operator selects an invalid option in response to a message, the response is not accepted by system log. Instead, another message is issued to the operator until a correct option is taken.

Note: When an immediate cancel (option 3) is selected, control is passed directly to the end-of-job (EOJ) routine by system log.

Two types of printed output are available through system log — logs and messages. Both are printed on the system log device.

- A *log* is a four- or six-character statement that identifies the type and source of an error.
- A *message* is a printed statement which can be used to indicate errors that have occurred or to issue instructions to the operator, such as requesting that a disk file be placed on a certain drive.

Logs and messages can be issued with or without being displayed on the 3277.

Three devices can be used as the system log device: the 1403 printer, the 3284 printer, or the CRT/Keyboard. You can change devices by entering a LOG statement in your job stream.

To use system log, you must do the following:

1. Build the log parameter list using the $LWTO or $LMSG macro.

2. Use the $LOGD macro instruction to establish equates for the log parameter list.

3. Issue the macro instruction:

[Name]	$LOG	[LIST-address] [,OPN0-address] [,OPN1-address] [,OPN2-address]

4. Process the operator's reply in your program.

Two types of messages can be displayed on the log: the system message and the message to the operator.

The $LMSG macro instruction generates a parameter list to display the standard system message. You can also include from 1 to 107 characters of text with this message. The operator can respond to this message by pressing the PF12 key, taking one of the four allowable options, then pressing ENTER. Option 0, 1, or 2 is returned to you for checking; option 3 means end-of-job. If the operator takes option 3, the system log routine goes directly to the end-of-job.

The $LWTO macro instruction generates a parameter list to display a message to the operator. This message is from 1 to 107 characters long and is prefixed with six characters which you specify using the $LWTO macro. You can request a reply (1 to 72 characters) from the operator, but if no reply is requested, the operator responds to the message by pressing PF12, then ENTER.

For either of the message types, the $LOGD macro instruction is specified to generate offsets into the parameter list and to define the meaning of the fields within the parameter list.

Text Length Considerations: Message text is displayed on the CRT in one, two, or three 36-character lines, depending on the text length. For example, if the text length is 60 characters, it takes one line of 36 characters and 24 characters from the second line to display the message.

Reply Length Considerations Three, 36-character lines (107 characters plus the CRT control character) are available for the message and reply. The number of characters allowed for the reply depends on the number of lines that contain test characters. Once a test character has been entered on a line, any non-text positions of that line cannot be used for a reply. Instead, the reply must begin on the next line. If the third line contains a text character, no reply can be made. For example:

If the text length is:	**The reply length can be:**
more than 1 character but less than 36	from 1 to 72 characters
more than 36 characters but less than 72	from 1 to 36 characters
more than 72 characters but less than 108	no reply can be made

Generate a Parameter List for WTO or WTOR ($LWTO)

This macro instruction generates a system log parameter list for the write-to-operator or write-to-operator-with-reply function.

The format of the $LWTO macro instruction is:

[Name]	$LWTO	TLEN-number, TADR-address [,COMP-code] [,HALT-code] [,SUBH-code] [,REPLY-Y/N] [,RLEN-number] [,RADR-address]

TLEN-number specifies the text message length (an unsigned, non-zero, decimal value). This operand is required and can be from 1 to 107 characters long.

TADR-address specifies the address of the leftmost byte of the text message; this operand is required.

COMP-code specifies the first two characters of the halt message. These characters are the component identification. If this operand is not specified, the default value is two blanks.

HALT-code specifies the second two characters of the halt message — the halt identification. If this operand is not specified, the default value is two blanks.

SUBH-code specifies the last two characters of the halt message — the subhalt identification. If this operand is not specified, the default value is two blanks.

REPLY-Y or N specifies whether or not a reply is requested. If Y (yes) is specified, system log waits for the operator to reply. If N (no) is specified or if this operand is omitted, no reply is allowed.

RLEN-number is the length of the reply (an unsigned, non-zero, decimal value, from 1 to 72). If REPLY-Y is specified, this operand is required.

RADR-address specifies the address of the leftmost byte of the reply area. If REPLY-Y is specified, this operand is required.

Generate a Parameter List for Message on System Log ($LMSG)

This macro instruction generates a system log parameter list for a log and/or message to the operator.

The format of the $LMSG macro instruction is:

[Name]	$LMSG	[FORMAT-code] [,COMP-code] [,HALT-code] [,SUBH-code] [,SEV-code] [,DEF-code] [,OPN0-Y/N] [,OPN1-Y/N] [,OPN2-Y/N] [,OPN3-Y/N] [TLEN-number] [,TADR-address]

FORMAT-code specifies the type and length of the system log parameter list. The valid code values and their meaning follow:

Code	Length of List	Format of List
A	7 bytes	F D CC HH O
B	9 bytes	F D CC HH O II
C	10 bytes	F D CC HH O L AA
D	12 bytes	F D CC HH O II L AA

where: F and D are flag bytes
CC is the component ID
HH is the message ID
O is the option indicator (this is determined by the settings of operands OPN0, OPN1, OPN2, and OPN3)
II is the sub-halt ID
L is the length of the text
AA is the address of the text

If this operand is omitted, FORMAT-B is assumed.

COMP-code specifies the first two characters of the message — the component description (CC). If this operand is not specified, two blanks are assumed.

HALT-code specifies the second two characters of the message — the message identification (HH). If this operand is not specified, two blanks are assumed.

SUBH code specifies the last two characters of the message — the subhalt identification (II). If FORMAT B or D was specified and this operand is omitted, two blanks are assumed.

SEV-code specifies the severity which conditions the selection of the default (DEF) option operand. This entry corresponds to the severity code entry in the NOHALT statement. If the severity code specified in the NOHALT statement is less than the value specified in this entry, the halt will be issued. Valid entries (from lowest to highest severity) are 1, 2, 4, and 8. If this operand is omitted, a severity of 8 is assumed.

DEF-code specifies the default option to select when executing in unattended mode. Valid entries are N, 0, 1, 2, and 3; if this operand is not specified, N (none) is assumed.

OPN0-Y/N specifies whether option 0 is allowed. If Y (yes) is entered, option 0 is allowed; if N (no) is entered or if this operand is omitted, option 0 is not allowed.

OPN1-Y/N specifies whether option 1 is allowed. If Y is entered, option 1 is allowed; if N (no) is entered or if this operand is omitted option 1 is not allowed.

OPN2-Y/N specifies whether option 2 is allowed. If Y (yes) is entered, option 2 is allowed; if N (no) is entered or if this operand is omitted, option 2 is not allowed.

OPN3-Y/N specifies whether option 3 is allowed. If Y (yes) is specified, option 3 is allowed; if N (no) is specified or if this operand is omitted, option 3 is not allowed.

Note: If option 3 is allowed, control will not be returned to your program.

TLEN-number specifies the text length. This entry (L), which is a decimal entry from 1 to 107, is required if FORMAT-C or D is specified.

TADR-address specifies the leftmost byte of the text address. This operand (AA) is required if FORMAT-C or D is specified.

Generate the Linkage to the System Log ($LOG)

This macro instruction generates the linkage required to use the system log function, and checks the response returned. The $LOGD macro instruction must be used with this macro instruction, to establish offsets in the system log parameter list.

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $LOG macro instruction.

The format of the $LOG macro instruction is:

[Name]	$LOG	[LIST-address] [,OPN0-address] [,OPN1-address] [,OPN2-address]

LIST-address specifies the address of the leftmost byte of the system log parameter list. If this operand is not specified, the address of the parameter list is assumed to be in register 2.

OPN0-address specifies the address of the routine that should receive control if option 0 is taken. If this operand is not specified, no check is made for a response of 0. You would use this operand only if the $LMSG macro was used to generate the parameter list.

OPN1-address specifies the address of the routine that should receive control if option 1 is the response. If this operand is not specified, no check is made for a response of 1. You would use this operand only if the $LMSG macro was used to generate the parameter list.

OPN2-address specifies the address of the routine that should receive control if option 2 is taken. If this operand is not specified, no check is made for a response of 2. You would use this operand only if the $LMSG macro was used to generate the parameter list.

Generate Displacements for System Log ($LOGD)

This macro instruction generates the field labels and offsets for the system log parameter lists. To avoid duplicate labels, you should use this macro instruction only once in a program.

The format of the $LOG macro instruction is:

	$LOGD	

General SCP Support

The general SCP macro instructions allow you to provide linkage to system functions by communicating with the Model 15 system control program.

Rollout/Rollin Linkage ($ROLL)

This macro applies to Program Number 5704-SC1 only.

You use $ROLL to interrupt (roll out) the current program so that another program can be executed. When the second program is finished, the first program is reinstated (rolled in) and continues executing.

Once rollout is initiated, the CRT/Keyboard becomes the system input device until otherwise specified by the user or until the interrupted program is rolled in.

When using rollout, you should follow these procedures:

1. Note the following restrictions:
   a. A program using $ROLL can execute in either partition, but will acknowledge the rollout request only when running in partition 1.
   b. A program using $ROLL must be so defined to the linkage editor using the ATTR parameter in the OPTIONS statement (see *IBM System/3 Overlay Linkage Editor Reference Manual*, GC21-7561).
   c. If the interrupting program also recognizes rollout requests, these requests will be ignored.
   d. The same I/O devices are available to the interrupting program as were available to the original program with exception of tape units.
   e. Whenever an interrupting program shares the same disk files as a rolled-out program, only reading and updating are allowed by the two programs. Loading and additions are not allowed.
   updating are allowed by the two programs. Loading and additions are not allowed.

2. Issue the $ROLL macro instruction:

   The coding generated by the $ROLL macro instruction determines whether rollout has been requested by entering the ROLLOUT operator control command. If rollout has not been requested, the current program continues executing. If the request is pending the rollout routines are called. Rollout performs the following steps:

   a. Places the current program (program being executed and the current contents of the scheduler work area on disk.
   b. Allows a new program or procedure to be run in place of the current program.
   c. Reloads the original program and restores the previous contents of the scheduler work area and passes control to the point where the original program was interrupted.

The format of the $ROLL macro instruction is:

[Name]	$ROLL	[INDEX-1/2]

INDEX-1/2 specifies which register can be used in the macro instruction. If this operand is omitted, register 2 is used.

Find a Directory Entry ($FIND)

A load module must be in the object library. Specific information must be obtained from the module's object library directory entry before a load or fetch can be performed. There are two ways you can locate a load module and obtain the information:

- Issue a $FIND before issuing a $LOAD, Form II. The information obtained during the find is used during the load operation.

- Issue a load with find ($LOAD, Form I), a fetch ($FTCH) or a fetch to address ($XCTL). These functions perform the find operation as part of their normal functions.

The $FIND macro instruction searches the object library directory for the requested module name and returns the directory entry in the parameter list.

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $FIND macro instruction.

The format of the $FIND macro instruction is:

[Name]	$FIND	NAME-module[,FIND-address] [,PACK-P/S]

NAME-module provides the name of the module to be found. Only names of object modules (O modules) can be entered here.

FIND-address specifies the label that becomes the address of a 12-byte parameter list built by the generated code. Initially the parameter list contains input to the supervisor. After execution, it contains the directory entry of the module. The format and contents of the parameter list after execution are shown in Figure 5. If this operand is not specified, a macro label is generated.

PACK-P/S specifies the program disk pack (P) or the system disk pack (S) to be searched. If this operand is not specified, P is assumed.

Load a Module ($LOAD)

This macro instruction loads a module into storage at the address you specify. Control is returned after the module has been loaded. You may then pass control to the module at the specified address. If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $LOAD macro instruction. Two forms of this macro can be used: load with find and load only.

Load with Find (Form I)

The load with find macro instruction locates the module and loads it into main storage.

The format of this macro instruction is:

[Name]	$LOAD	NAME-module name [,FIND-address] [,LOAD-2/address] [,USE-R/NR] [,PLIST-address] [,PACK-P/S]

Entry	Number of Bytes	Displacement	Description
Disk Address	2	1	Cylinder/sector address of the module.
Number of text sectors	1	2	Type O—Sector length of the module in hexadecimal Type R—Category of routine
Link edit address	2	4	Hexadecimal storage address at which the module was linkage edited. See note 2.
Displacement of RLDs	1	5	Number of bytes, in hexadecimal, into the first sector containing RLDs, of the first relocation directory (RLD) entry of the module.
Entry point address	2	7	Hexadecimal storage address at which program execution begins (without RLDs).
Storage size	1	8	Amount of storage (in sectors) required to execute the program.
Attributes	2	A	Byte 1: Bit 0 1 = Permanent entry 0 = Temporary entry 1 1 = Inquiry program 2 1 = Rollout-evoking program (5704-SC1 only) 1 = External buffers (if byte 2, bit 1 = 1) (5704-SC2 only) 3 1 = Must run in dedicated environment (5704-SC1 only) Reserved (5704-SC2 only) 4 1 = Requires source information 5 1 = Deferred mounting allowed 6 1 = PTF applied 7 1 = Overlay object program Byte 2: Bit 0 1 = The system input device must be dedicated to this program 1 1 = Checkpoint/Restart program 1 = External buffers (if byte 1, bit 2 = 1) (5704-SC2 only) See note 1 2 1 = This program will access the source file directly 3 1 = Macro processor is allowed (5704-SC1 only) 1 = Model 15D program (5704-SC2 only) 4 1 = This is a privileged program 5 1 = This program requires that a new load address be calculated at load time to place it in main storage beyond its own program common region 6 1 = 3340 data management (5704-SC1 only) Reserved (5704-SC2 only) 7 Memory resident overlays
Level	1	B	Release version of this entry.

Notes:

1. For 5704-SC2 the checkpoint/restart and external buffer attributes are mutually exclusive.
2. For 5704-SC2, if the entry has the attributes for external buffers, the low-order byte of the link edited address contains the number of sectors minus 1.

Figure 5. Find Parameter List Description

Entry	Number of Bytes	Displace-ment	Description
Disk Address	2	1	Cylinder/sector address of the module in hexadecimal. See note 1.
Number of text sectors	1	2	Sector length of the module, in hexadecimal.
Link edited address	2	4	Storage address at which the module was linkage edited. See note 2.
Displacement of RLDs	1	5	Hexadecimal displacement, in bytes, into the first sector containing RLDs, of the first relocation directory (RLD) entry of the module.
Relocated entry point address	2	7	Storage address at which program execution begins, after resolving RLDs.
Load address	2	9	Address at which the requested module is loaded.

Notes:

1. If a directory entry was not found on a load with find, the first byte contains a character 0.
2. For 5704-SC2, if the entry has the attributes for external buffers, the low-order byte of the link edited address contains the number of sectors minus 1.

Figure 6. Find Parameter List after Load Execution

NAME-module name provides the name of the module to be loaded and is required. Only O modules can be specified.

FIND-address becomes the address of the parameter list passed to the find routine. The parameter list is generated by the macro processor. After execution of the load, this parameter list contains the modified entry for the module as shown in Figure 6.

LOAD-2/address specifies the address where the module is to be loaded into main storage. The 2 indicates that the address is in register 2; the address is the symbolic address where the module is to be loaded. If this operand is not specified, 2 is assumed.

USE-R/NR indicates whether the code generated by the macro instruction is to be reusable (R) or nonreusable (NR). If the operand is not specified, NR is assumed.

You can reuse the generated code to load the same module more than one time, or to load different modules. If you wish to load different modules using the same generated code, you should also specify the PLIST operand.

PLIST-address is used only when the generated code is reusable. The address specified identifies the leftmost byte of a parameter list passed to the load routine. To load a different module using the same generated code, you must update the parameter list to indicate the desired module. Figure 7 shows the format and contents of the parameter list.

PACK-P/S specifies the program disk pack (P) or the system disk pack (S) containing the requested module. If this operand is not specified, P is assumed.

Load Only (Form II)

The load-only macro instruction loads a module previously found by the $FIND macro instruction. The format of this macro instruction is:

[Name]	$LOAD	FIND-address [,LOAD-2/address] [,PACK-P/S]

FIND-address is the address used in the previous $FIND macro instruction. It identifies the directory entry of the module in main storage. After execution of the load, this address points to the directory entry of the module as shown in Figure 6.

LOAD-2/address specifies the address where the module is to be loaded in main storage. The 2 indicates that the address is in register 2; the address is the symbolic address where the module is to be loaded. If this operand is not specified, 2 is assumed.

PACK-P/S specifies the program disk pack (P) or system disk pack (S) containing the requested module. If this operand is not specified, P is assumed.

Load a Module and Pass Control ($FTCH)

The fetch macro instruction ($FTCH) finds a module in the directory, loads the module into main storage, and passes control to it. Your program does not regain control. When a module is fetched into main storage, the relocation factor is added, as necessary, to the module's link edit address. This determines the location in main storage where the module is loaded. The module receives control at its entry point.

Entry	Number of Bytes	Displacement	Description
Module Type	1	0	Must contain O to indicate an object module.
Module Name	6	6	The name of the module to be loaded.
FE	1	7	X'FE'
Load Address	2	9	The address at which the module is to be loaded.

Figure 7. Load Parameter List Description

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $FTCH macro instruction.

The format of the $FTCH macro instruction is:

[Name]	$FTCH	Name-module name[,PACK-P/S]

NAME-module name specifies the object module to be fetched into main storage. The name must be the same as the name in the directory entry.

PACK-P/S specifies the program disk pack (P) or the system disk pack (S) containing the requested module. If this operand is not specified, P is assumed.

Load a Module and Exchange Control ($XCTL)

This macro instruction finds a module in the directory, loads the module into main storage at the address you specify, and passes control to it. Control is not returned to your program. As with the $FTCH macro instruction, relocation factors are resolved, and control is passed to the entry point of the program.

The format of the $XCTL macro instruction is:

[Name]	$XCTL	NAME-module name[,LOAD-2/address] [,PACK-P/S]

NAME-module name specifies the name of the module to be loaded and given control. The module must be an O module.

LOAD-2/address specifies the address where the module is to be loaded in main storage. The 2 indicates that the address is in register 2; the address is the symbolic address where the module is to be loaded. If this operand is not specified, 2 is assumed.

PACK-P/S specifies the program disk pack (P) or the system disk pack (S) containing the requested module. If this operand is not specified, P is assumed.

Generate a Translate Parameter List ($TRL)

This macro instruction generates a parameter list needed by the Model 15 Translate routine. This list is called via the $TRAN macro instruction. $TRL does not generate executable code. Figure 8 shows the format of the translate parameter list.

Translate Routine Operation

To use the Model 15 translate routine, you must provide a translate area. The format of the area is:

Byte	Field Description
0	Byte contents used to determine whether a character is to be translated.
1	Byte contents are substituted for characters that are not to be translated.
2-257	256-byte translate table.

The translate routine processes a field, specified by the $TRAN macro instruction, one byte at a time.

The translate table must be constructed so that the displacement (from the beginning of the table) of the translated representation of a character is equal to the hexadecimal representation of the untranslated character. (For example, if you want to translate X'C1' to X'41', you could construct a translate table in which the value at displacement X'C1' in the table is X'41'.)

The contents of the byte at a given displacement are compared with the contents of the first byte in the translate area (byte 0). If an equal compare results, the character is considered to be invalid and the following actions are performed:

- The completion code in the parameter list is set to indicate that an invalid character was detected.
- The hexadecimal value in the second byte of the translate area (byte 1) is substituted for the original character.

If an unequal compare results, the hexadecimal value in the translate table is substituted for the original character.

The format of the $TRL macro instruction is:

[Name]	$TRL	TO-address,FROM-address,LEN-number, TRT-address

TO-address specifies the symbolic address of the first byte of the data to which the translated data will be moved.

FROM-address specifies the symbolic address of the first byte of the data field to be translated. This address may be the same as the address specified in the TO operand.

LEN-number specifies the decimal length of the FROM field.

TRT-address specifies the symbolic address of the first byte of the translate area.

All four operands are required.

Generate a Translate Table ($TRTB)

This 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, and can be addressed by $TRL when you translate data.

The format of the $TRTB macro instruction is:

[Name]	$TRTB	[CODE-E/A] [,HEX-hex]

CODE-E/A specifies whether the character code of the data to be translated is EBCDIC (E) or ASCII (A). If this operand is omitted, EBCDIC (E) is assumed. If CODE-E is specified, $TRTB generates a 258-byte table; if CODE-A is specified, $TRTB generates a 130-byte table.

Note: If you specify CODE-A, you may want to specify DC 128XL1'FF' after the $TRTB macro instruction to allow for invalid ASCII characters.

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 X'3F' for EBCDIC or X'1A' for ASCII.

Generate an Interface to the Translate Routine ($TRAN)

This macro instruction generates an interface to the Model 15 translate routine. After the translate routine has finished, control is returned to your program with a completion code in the translate routine parameter list. The address of the parameter list is in register 1. You should check the completion code to see if any characters that are not to be translated were encountered.

The format of the $TRAN macro instruction is:

[Name]	$TRAN	[TRL-address]

TRL-address specifies the symbolic address of the translate parameter list. If this operand is not entered, the address is assumed to be in register 1. See Figure 8 for a description of the parameter list.

Field Length	Field Description
2	Address of the translate area (your program must define the translate area)
2	FROM field address, for translation
2	TO field address for translation
2	Number of bytes to translate
1	Completion code: X'00' —translation complete, no errors X'FF' —invalid character detected

Figure 8. Translate Parameter List

Snap Dump Main Storage ($SNAP)

This macro instruction provides an interface with the non-terminating system storage dump routine. You must specify a dump identifier and the limits of the area to be dumped. The contents of the specified main storage area are put on the 1403 printer. Output from the dump routine consists of:

- The specified dump identifier.
- The contents of register 1 (XR1), register 2 (XR2), the Instruction Address Register (IAR), the Program Mode Register (PMR), the Address Recall Register (ARR), and the Local Storage Register (LSR).
- The contents of the specified main storage area.

Control is returned to the next sequential instruction in your program.

The format of the $SNAP macro instruction is:

[Name]	$SNAP	ID-hex,START-address,END-address

ID-hex specifies a 2-byte hexadecimal number to be used as the dump identifier.

START-address specifies the symbolic address of the low-storage limit of the area to be dumped.

END-address specifies the symbolic address of the high-storage limit of the area to be dumped.

All three operands are required.

Obtain System Date ($DATE)

This macro instruction generates the code necessary to retrieve the system date and place it at a specified location in your program.

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $DATE macro instruction.

The format of the $DATE macro, instruction is:

[Name]	$DATE	[LABEL-address]

LABEL-address indicates the address of the leftmost byte of a six-byte area in which you want the system date placed. If this operand is not specified, the address at which you want the date to be placed is assumed to be in register 2.

Generate Timer IOB ($TIOB)

This macro instruction generates an IOB for the interval timer. $TIOB must be used if you use $SIT, $RIT, or $TOD in your program. The format of the timer IOB is shown in *Appendix C: Input/Output Blocks*.

The format of the $TIOB macro instruction is:

[Name]	$TIOB	[DATE-Y/N]

DATE-Y/N specifies whether a date field is to be generated. If this operand is not entered, N (no) is assumed. If the $TOD macro instruction is used in your program, Y (yes) must be entered. If DATE-Y is specified, use of the $SIT macro instruction destroys the date in the timer IOB.

Set Interval Timer ($SIT)

This macro instruction sets the interval timer to cause a timer interrupt after the specified amount of time has elapsed. Before issuing this macro instruction you must place the desired interval in bytes 2-7 of the timer IOB. When the interval is set, byte 8 in the timer IOB is set to X'00'. When the interval is expired, byte 8 is set to X'40'.

The format of the $SIT macro instruction is:

[Name]	$SIT	[IOB-name]	[,TYPE-DEC/BIN/TU/TOD] [,ITYPE-REAL/WAIT/TASK]

IOB-name specifies the name of the timer IOB generated by the $TIOB macro instruction. If this operand is omitted, the address of the IOB is assumed to be in register 2.

TYPE-DEC/BIN/TU/TOD specifies the format of the time interval in the IOB. You must establish the time interval in bytes 2-7 of the IOB before issuing the $SIT macro instruction. The valid time interval formats are:

- DEC: The time interval is the amount of time in decimal that is to elapse before the timer interrupt. The time interval is a six-byte decimal number specifying hours, minutes, and seconds (HHMMSS).
- BIN: The time interval is a 32-bit binary number specifying the number of seconds that are to elapse before the timer interrupt. The binary value must be right-justified in bytes 4-7 of the IOB field.
- TU: The time interval is a 32-bit binary number specifying the number of timer units that are to elapse before the timer interrupt. One timer unit is 3.33 milliseconds. The binary value must be right-justified in bytes 4-7 of the IOB field.
- TOD: The time interval is the actual time of day when the timer interrupt is to occur. The time is a six-byte decimal number specifying the hour, minute and second (HHMMSS).

If this operand is omitted, DEC is assumed.

ITYPE-REAL/WAIT/TASK specifies the type of interval to be timed. The types of time intervals are:

- REAL: The timer decrements the time interval continuously for all types of processing.
- WAIT: The program issuing the $SIT macro instruction is placed in a wait state for the specified time interval. When the time has expired, control returns to the instruction following the $SIT macro instruction.
- TASK: The timer decrements the time interval only while the task issuing the $SIT macro instruction is running.

If this operand is omitted, REAL is assumed.

Return Time and Date ($TOD)

This macro instruction returns the time of day and the system date to the program. The time of day is returned in bytes 2-7 of the timer IOB, the system date in the next six bytes. DATE-Y must be specified in the $TIOB macro instruction if the $TOD macro instruction is used.

The format of the $TOD macro instruction is:

[Name]	$TOD	[IOB-name] [,REF-Y/N] [,TYPE-DEC/BIN/TU]

IOB-name specifies the name of the timer IOB generated by the $TIOB macro instruction. If this operand is omitted, the address of the IOB is assumed to be in register 2.

REF-Y/N specifies whether the macro instruction is being issued from the transient area or a program partition. Y (yes) indicates the macro instruction is issued by a transient that must be refreshed; N (no) indicates it is issued from a program partition. If this operand is omitted, N (no) is assumed.

TYPE-DEC/BIN/TU specifies how the time is to be returned in the timer IOB. The valid formats are:

- DEC: The time returned is a six-byte decimal number indicating the time in hours, minutes and seconds (HHMMSS).
- BIN: The time returned is a 32-bit binary number indicating the time in seconds. The binary number is right-justified in bytes 4-7 of the IOB field.
- TU: The time returned is a 32-bit binary number indicating the time in timer units. One timer unit is 3.33 milliseconds. The binary number is right-justified in bytes 4-7 of the IOB field.

If this operand is omitted, DEC is assumed.

Return Interval Time ($RIT)

This macro instruction returns the remaining amount of time in a time interval or cancels an unexpired time interval. The remaining time is returned in bytes 2-7 of the timer IOB established by the $TIOB macro instruction. The time interval must have been set by the $SIT macro instruction and is returned in the format specified in that macro instruction.

The format of the $RIT macro instruction is:

[Name]	$RIT	[IOB-name] [,CANCEL-Y/N]

IOB-name specifies the name of the timer IOB generated by the $TIOB macro instruction. If this operand is omitted, the address of the IOB is assumed to be in register 2.

CANCEL-Y/N specifies whether the remaining time in the interval is to be cancelled. If this operand is omitted, N (no) is assumed.

End-of-Job ($EOJ)

The $EOJ macro instruction generates the linkage required to execute the end of job routine. The option to perform an immediate cancel or a controlled cancel is provided.

The format of the $EOJ macro instruction is:

[Name]	$EOJ	[CANCEL-NORMAL/IMMED/CONTRL]

CANCEL-NORMAL/IMMED/CONTRL specifies the action that the system should take as it terminates the program. If this operand is omitted or if NORMAL is specified and the system is in HALT mode, an end of job step message is issued and job processing terminates in the partition; if NOHALT mode is in effect, no message is issued and job processing continues.

If IMMED is specified, the disposition of files depends upon the function being performed:

- New files being created will not be retained
- Old files being deleted will be retained
- Old files being added to will not reflect the additions
- Old files being updated will reflect the updates made prior to the point at which this $EOJ macro instruction was executed
- Add files (except for consecutive) will reflect additions if file sharing (5704-SC2 only).

If CONTRL is specified, the files being used by your program reflect all activity which took place up to the point the $EOJ macro instruction was executed.

If your program specifies CANCEL-IMMED or CANCEL-CONTRL and it is a step of a job or a procedure in a chained procedure, all remaining steps in the job or all remaining procedures in the chained procedure are cancelled. Also, EJ or ES is displayed regardless of the status of the system halt mode, and job processing in that partition terminates.

If either CANCEL-IMMED or CANCEL CONTRL is specified, the input stream is flushed in the following manner:

- If you are executing in job mode on a system on which input spooling is active, job processing in the partition resumes with the next job on the job input queue.
- If you are executing in job mode on a system on which input spooling is not active, the job stream in the system reader is flushed until a JOB statement or a slash period (/.) statement is encountered.
- If you are executing in step mode, the input stream will be flushed until a JOB, LOAD, CALL, slash period (/.) or slash ampersand (/&) statement is encountered.

INPUT/OUTPUT SUPPORT MACROS

The input/output support macro instructions provide access to devices without requiring that you write extensive routines to perform each function. The input/output support macro instructions are divided into eight groups.

1. General macro instructions are used with all device types. The following macros are in this group:

   $ALOC
   $CKL, $CHK — Teleprocessing only
   $CLOS
   $COMN
   $DTFO
   $OPEN

2. Card macro instructions support card devices. The following macros are in this group:

   $DTFC
   $GPC
   $GETC
   $PUTC

3. Printer macro instructions support printer devices. The following macros are in this group.

   $DTFP
   $PUTP

4. Disk macro instructions provide support and linkage to disk data management. The following macros are in this group:

   $DTFD
   $GETD
   $IOBD
   $IOED
   $PUTD
   $RDD
   $WAIT
   $WRTD

5. Tape macro instructions provide support and linkage to tape data management. The following macros are in this group:

   $CTLT
   $DTFT
   $GETT
   $PUTT
   $RDT
   $WRTT
   $WTT

6. Device independent macro instructions provide support and linkage to device independent data management. The following macros are in this group:

   $DTFI
   $GETI
   $PUTI

7. CRT macro instructions support CRT devices. The following macros are in this group:

   $CQEP
   $DTFS
   $GETS
   $PFKT
   $PFKY
   $PGS
   $PUTS

8. 3741 macro instructions provide support and linkage to 3741 data management. The following macros are in this group:

   $DTFK
   $GETK
   $PUTK

General I/O Support

The general I/O support macro instructions are used with both unit record, tape, and disk devices. The normal sequence for using these macro instructions is:

1. $ALOC to allocate the device to your program partition.

2. $OPEN to prepare the file or device for use.

3. I/O operations and any processing required.

4. $CLOS to prepare the file and/or device for job termination.

Allocate Space ($ALOC)

The routines called by the $ALOC macro instruction allocate unit record input/output devices and space on disk and tape devices. These routines check to ensure that:

- The system supports the requested device.
- The device requested is available to the requesting program.
- The LOCATION parameter of the OCL file statement is valid.
- The correct disk pack is mounted and space is available to the calling program.
- The number of disk, tape, and device independent DTFs in the calling program is limited as follows:
  - 5704-SC1—The limit is 40.
  - 5704-SC2—The limit is 192 with at least a 10K partition. The limit is 128 with only an 8K partition.

  For 5704-SC2, not all 192 disk and tape DTFs may be allocated in a single call to allocate. The maximum number of tape and/or disk DTFs in a single chain is as follows:
  - If all are disk, the maximum is 165.
  - If all are tape, the maximum is 148.
  - For combinations of disk and tape, the maximum varies between 148 and 165. To determine whether or not the DTFs can be allocated in a single call to allocate, multiply the number of disk DTFs by 17 and multiply the number of tape DTFs by 19. If the sum of the above two numbers is not greater than 2816, the DTFs may be allocated in a single call.

  If the DTFs cannot all be allocated in a single call, make two DTF chains and two calls to allocate.
- The correct tape input file is mounted, or the tape label is written on the output file, and that the tape is positioned at the beginning of the file.

An allocate request requires that pre-open DTFs be supplied as input to the routine. For a description of DTFs, see *Define the File for Card ($DTFC), Define the File for Disk ($DTFD), Define the File for Device Independent ($DTFI), Define the File for Printer ($DTFP), Define the File for CRT ($DTFS), Define the File for Tape ($DTFT),* and *Define the File for 3741 ($DTFK).* When the allocate request is for a disk, a tape device, or a device independent file, OCL file statements are also required. More than one DTF can be allocated at one time by chaining the DTFs. To chain DTFs, you must enter the address of the next DTF in the DTF you are building. The last DTF in a chain has X'FFFF' entered in place of the address. If your program operates as an interrupt handler, such as a binary synchronous communications program, all DTFs in the program should be chained together and allocated in one operation. When an error condition occurs, the allocate routine calls halt/syslog to display the proper halt code.

Note that if you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $ALOC macro instruction.

The following output is produced when control is returned to your program.

- The contents of register 1 are restored.
- The format-1 labels and configuration record are updated.
- For a non-disk or non-tape DTF, bit 1 in the rightmost byte of the attribute bytes of the post-open DTF is set on to indicate device allocation.
- For a device independent DTF, the device code (displacement 0 in the DTF – $DFDEV) is altered to indicate the appropriate device.
- The address of the first DTF allocated is returned in register 2.

Note: If you are using telecommunications, $ALOC must not be issued while a telecommunications operation is in process.

The format of the $ALOC macro instruction is:

[Name]	$ALOC	[DTF-address]

DTF-address specifies the address of the high-order byte of the DTF being allocated. If this operand is not entered, the address of the DTF is assumed to be in register 2.

Prepare An I/O Device ($OPEN)

This macro instruction prepares an input/output file for data transfer. The file to be prepared (opened) must previously have been allocated by using the allocate macro instruction. Depending on the device, one or more of the following functions are performed for each file opened.

- The post-open DTF is formatted (see Figure 9).
- Pre-open DTF information is preserved in the format-1 label as required.
- Input/output buffers, index buffers, and IOBs are formatted.
- Buffers are initialized as required.
- Disk file share area is prepared as required (5704-SC2 only).
- Cards are positioned at the wait station for card output files.
- The index area on disk for indexed files and the data area on disk for direct files are formatted as required.
- Diagnostics are performed to ensure that:
  - The access method and the file organization are compatible.
  - The volume and file are mounted on the correct disk or tape drive.
  - Share attributes are checked for disk files (5704-SC2 only).

Note: More than one DTF can be opened at one time by chaining the DTFs. To chain DTFs, you must enter the address of the next DTF in the DTF you are building. The last DTF in a chain has X'FFFF' entered in place of the address. See *$DTFC, $DTFD, $DTFI, $DTFP, $DTFS, $DTFT,* and *$DTFK.*

Pre-Open Conditions	Post-Open Conditions
1. Unformatted DTFs are present for output files. 2. The I/O buffer is in the unformatted mode.	1. Formatted DTFs are created. 2. I/O buffers, IOBs, and various work areas are formatted. 3. A bit is set on in the DTF attribute bytes to indicate an opened file.

Figure 9. Comparison of Pre-Open and Post-Open DTFs and Data Areas

Input: The pre-open DTF and format-1 label are input to the open routine. Before the open macro instruction is issued, you must be sure to have the device allocated by previously issuing the allocate macro instruction. Also, if you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $OPEN macro instruction. You must also consider the following in preparing the DTF:

- The disk access method must be compatible with the disk file organization of the file being opened.
- The access method must be compatible with the access method of the same file opened in the other program level or for an inquiry program (see *Rollout*).
- The record length, block length, and key length must be specified correctly.
- The file must have share specified if it is to be shared for disk files (5704-SC2 only).

Output: The open routine returns control to your program when the requested file has been opened. The following output is produced:

- The contents of register 1 are restored.
- The format-1 labels are updated.
- Bit 7 in the rightmost attribute byte in the post-open DTF is set on to indicate the file has been opened.
- The device code (displacement 0 in the DTF – $DFDEV) is altered to indicate the unit on which the disk or tape file resides.
- The buffers are initialized.

- The address of the last DTF opened is returned in register 2.
- The file share area is updated for disk files with share specified (5704-SC2 only).

The format of the $OPEN macro instruction is:

[Name]	$OPEN	[DTF-address]

DTF-address specifies the address of the leftmost byte of the DTF for the file to be opened. If this operand is not entered, it is assumed that the address is in register 2.

Note: Any files opened during a job should be closed before the job ends ($CLOS macro). For 5704-SC2 only, any open files will be closed during EOJ processing (normal or abnormal). Therefore, DTFs must be available at this time. Any disk files opened using file sharing must be closed to perform required actions on the file share area.

Generate a Checklist ($CKL)

This macro instruction creates an entry for a checklist. It does not generate executable code. A checklist identifies DTFs to be checked for I/O completion, or determines whether the PF9 key has been pressed. Two kinds of DTFs can be identified in the checklist:

- Binary Synchronous Communications (BSC) DTFs
- Dummy DTFs (15-byte DTFs used to check for the PF9 key)

All the checklist entries that are to be tested by the same $CHK macro instruction must be issued consecutively. The same DTF may be in the list more than once. The checklist entries that are generated are contiguous in main storage. You can then issue the $CHK macro instruction to test the entire list, by specifying the first entry in the list; or begin testing anywhere in the list, by specifying the label of one of the later entries.

Note: The address you specify in the $CHK macro instruction identifies the beginning of the check operation. Any entries occurring earlier in the list are ignored in that operation.

For a description of the checklist entries, see Figure 10.

Disp	Field Description
0	Flag byte: X'80'—Skip this entry X'40'—Request key (PF Key 9) should be checked X'20'—This is the last entry in the checklist X'10'—Return control to the user if no I/O completion is found (significant only in the first entry of a checklist) X'08'—Code destroyed X'04'—Program function key not available
1-2	Address of the DTF for this entry

Figure 10. Checklist Format

For a description of BSC, see *IBM System/3 Multiline/ Multipoint Binary Synchronous Communications Reference Manual,* GC21-7573.

The format of the $CKL macro instruction is:

[Name]	$CKL	DTF-address[,SKIP-Y/N] [,REQK-Y/N] [,RTN-Y/N] [,LAST-Y/N]

DTF-address specifies the symbolic address of the first byte of the DTF for which this entry is being created. A dummy DTF, 15 bytes in length, is required to check if the PF9 key was pressed. Displacement X'00' in this DTF should contain a X'10' and a completion code will be returned at displacement X'0E'.

SKIP-Y/N specifies whether this entry should be skipped when the checklist is scanned. If this operand is omitted, N (no) is assumed. If Y is specified, you must update the checklist entry before you can check the DTF specified in this macro instruction. You can access the skip indicator in the entry by using the name specified on the macro instruction.

REQK-Y/N specifies whether the check routine (see index entry *$CHK macro instruction*) should check if the PF9 key has been pressed. Whenever you want the check routine to check for a PF9 request, 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 pressed the PF9 key, a completion code of X'50' is posted at displacement X'0E' of the dummy DTF.

RTN-Y/N specifies whether you want control returned to your program even if no I/O operation is complete. This operand is valid only for the first entry in the check list. If this operand is not entered, N (no) is assumed.

LAST-Y/N specifies whether this is the last entry in the checklist. LAST-Y (yes) must be specified for the last entry. If this operand is omitted, N (no) is assumed.

Check for I/O Completion ($CHK)

This macro instruction generates the linkage required to use the check routine. You must issue the $CHK macro instruction for each BSC get, put, read, write, or online test request. For a description of BSC macro instructions, see *IBM System/3 Multiline/Multipoint Binary Synchronous Communications Reference Manual,* GC21-7573.

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $CHK macro instruction.

Check Routine Operation

Note: When using this macro, the user program must equate $PARM to 2.

The check routine tests for completion of an I/O operation by examining the DTFs identified in the checklist – see *Generate a Checklist ($CKL).* If an I/O operation is complete, the completion code is set in the DTF, and the address of the DTF is returned in register 2 to the calling program. No subsequent DTFs in the list are tested.

When no I/O completion is found by the end of the checklist, control is returned to your program with the address of the last DTF in the list register 2 if:

Each entry in the list is inactive, closed, or has the skip indicator on (a completion code of X'57').

RTN-Y was specified in the $CKL macro instruction that created the first entry in the checklist (a completion code of X'56').

The format of the $CHK macro instruction is:

[Name]	$CHK	[CKL-address]

CKL-address specifies the symbolic address of the first entry in the checklist. You can also begin at a subsequent point in the checklist by specifying the symbolic address of a later entry. If this operand is omitted, the address is assumed to be in register 2.

Note: The address you specify identifies the beginning of the check operation. Any entries occurring earlier in the list are ignored in that operation.

Prepare a Device for Termination ($CLOS)

The close macro instruction prepares a device for job termination. The routine returns post-open DTFs to their pre-open state and updates file labels to reflect the current file status. For devices other than disk or tape, only the entries related to the requested functions are restored. If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $CLOS macro instruction.

Input to the close routine consists of the post-open DTF and the format-1 labels. The allocate and open macro instructions must have previously been issued.

Output created by $CLOS is returned to your program when control is returned. The output consists of:

- The contents of register 1 restored.
- The post-open DTFs reinitialized to the pre-open state.
- Any pending operations for unit record devices performed.
- The format-1 label for disk updated to indicate current file status.
- The buffer contents scheduled for disk or tape output and disk update operations written.
- The data and index written to disk, and an indicator set if key sorting is required at end-of-job for output files and file additions.
- Tape trailer labels read or written.
- The file share area is cleared or updated for the disk file specified as share.

Note: More than one DTF can be closed at one time by chaining the DTFs. To chain DTFs, each DTF to be closed must contain the address of the next DTF in the chain. The last DTF in a chain has X'FFFF' entered in place of the address.

The format of the $CLOS macro instruction is:

[Name]	$CLOS	[DTF-address]

DTF-address specifies the address of the leftmost byte of the DTF to be closed. If this operand is not entered, the address is assumed to be in register 2.

Generate DTF Offsets ($DTFO)

This macro defines the DTF labels, offsets, field contents, and field lengths for all devices and access methods supported for the Model 15. To avoid duplicate labels, this macro instruction should be used only once in each program; you should also set the operands to indicate any devices you plan to use in the program.

The format of the $DTFO macro instruction is:

[Name]	$DTFO	[DISK-Y/N] [,TAPE-Y/N] [,IND-Y/N] [,MFCU-Y/N] [,MFCM-Y/N] [,D3741-Y/N] [,D2501-Y/N] [,D1442-Y/N] [,D1403-Y/N] [,D3284-Y/N] [,CRT-Y/N] [,ALL-Y/N] [,FIELD-Y/N]

DISK-Y/N specifies whether labels are to be generated for disk devices. If this operand is not entered, N (no) is assumed.

TAPE-Y/N specifies whether labels are to be generated for a tape device. If this operand is not entered, N (no) is assumed.

IND-Y/N specifies whether labels are to be generated for an independent device. If this operand is not entered, N (no) is assumed.

MFCU-Y/N specifies whether labels are to be generated for the MFCU. If this operand is not entered, N (no) is assumed.

MFCM-Y/N specifies whether labels are to be generated for the MFCM. If this operand is not specified, N (no) is assumed.

D3741-Y/N specifies whether labels are to be generated for the 3741. If this operand is not specified, N(no) is assumed.

D2501-Y/N specifies whether labels are to be generated for the 2501. If this operand is not entered, N (no) is assumed.

D1442-Y/N specifies whether labels are to be generated for the 1442. If this operand is not entered, N (no) is assumed.

D1403-Y/N specifies whether labels are to be generated for the 1403. If this operand is not entered, N (no) is assumed.

D3284-Y/N specifies whether labels are to be generated for the 3284. If this operand is not entered, N (no) is assumed.

CRT-Y/N specifies whether labels are to be generated for the CRT. If this operand is not entered, N (no) is assumed.

ALL-Y/N specifies whether labels are to be generated for all devices supported. If this operand is not entered, N (no) is assumed.

FIELD-Y/N specifies whether to generate the labels which define the contents for a DTF field. If this operand is not specified, Y (yes) is assumed.

COMMON Equates ($COMN)

This macro instruction generates equates for various labels and values, such as register equates, which may be used in the program. This macro instruction is not required.

The format of the $COMN macro instruction is:

	$COMN	

Card Device Support

This section describes the macro instructions that support card devices. The following functions are provided:

- Build a pre-open DTF for card device and assign its offsets.
- Build the interface required to read/punch/print records using a card device.

The DTF provides information to the card device data management routines that perform input/output operations.

Define the File for Card ($DTFC)

The DTF provides information needed to allocate, open, and access a card device. This macro instruction generates the code that builds a card DTF.

The format of the $DTFC macro instruction is:

[Name]	$DTFC	IOBA-address, NIOB-number [,DEV-code] [,UP-mask] [,CHN-address] [,RCAD-address] [,OPC-code] [,DEFER-Y/N] [,CARDI-Y/N] [,PRINT4-Y/N] [,FEED-Y/N] [,STACKR-number] [,READA-address] [,PUNCHA-address] [,PRINTA-address] [,READL-number] [,PUNCHL-number] [,PRINTL-number] [,PRHEAD-mask] [,ALIGN-Y/N]

IOBA-address is a required operand specifying the address of the leftmost byte of the first IOB. The area identified by this operand must be large enough to contain one IOB for single buffering or two IOBs for double buffering (specified via the operand NIOB-2). The sizes for each card IOB are:

2501 – 25 bytes
MFCU – 29 bytes
1442 – 29 bytes
MFCM – 31 bytes

NIOB-number is a required operand specifying the number of IOBs associated with this DTF. This entry must have a value of 1 or 2.

DEV-code specifies the card device desired. The possible values for code are MFCU1, MFCU2, MFCM1, MFCM2, 1442, and 2501. If this operand is omitted, MFCU1 is assumed.

UP-mask specifies the mask to test the eight external indicators. The code for the UP-mask must be specified as 8 binary bits. For example, to test bits 0, 3, 5, and 7, you would enter UP-10010101. The UP-mask is compared to the external indicators set on by the SWITCH statement for conditionally opening files. If the bits that are on in the UP-mask are also on in the external indicators set on by the SWITCH statement, the file will be opened. If the UP-mask is all zeroes or not used, the file will be unconditionally opened.

Note: Information on setting external indicators (SWITCH statement) can be found in the *IBM System/3 Model 15 System Control Programming Reference Manual* (for Program Number 5704-SC1), GC21-5077, and in the *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual* (for Program Number 5704-SC2), GC21-5162.

CHN-address indicates the address of the next DTF in the chain of DTFs. If there is no DTF chain or if this is the last DTF in the chain, the operand should be omitted (X'FFFF' is assumed).

RCAD-address specifies the leftmost byte of the logical record for an output operation. If this operand is not entered, a value of X'0000' is assumed; this value must be updated before an output operation is performed. If this operand is specified for an input operation, the value specified will not be used, and this operand will be modified by the card data management module to contain the address of the card that was read.

OPC-code defines the operation code for the DTF. A list of the possible codes and their meanings are:

Code	Meaning	
RD	Read	
PU	Punch	
PR	Print	
PP	Punch and print	Not allowed for MFCM1 or MFCM2
RPU	Read and punch	
RPR	Read and print	
RPP	Read, punch, and print	

If this operand is not specified, the operation code for a feed is set.

DEFER-Y/N is used only with output operations to the MFCM and MFCU. This operand enables you to print one record on a card and punch a different record in the same card. The $DTFC, $PUTC, and $GPC macro instructions can be used to specify this process. For $GPC or $PUTC, you must first issue that macro instruction for either a print or a punch with DEFER-Y. You then modify the logical record as needed to a different format and issue that macro instruction for the remaining operation with DEFER-N. Both operations are then performed. If this operand is not specified, N (no) is assumed.

CARDI-Y/N specifies whether or not to perform a read card image operation on the 1442 or 2501. If N (no) is specified or if this operand is omitted, a card image read is not performed.

PRINT4-Y/N specifies whether to print three or four tiers for a print request from the MFCU. Y (yes) indicates that four tiers should be printed; if this operand is omitted or if N (no) is specified, three tiers are printed.

FEED-Y/N specifies whether or not to perform a feed operation after a 1442 punch operation. If Y (yes) is specified or if this operand is omitted, a feed is performed after the punch operation.

STACKR-number specifies the stacker to be used for this card operation. This operand is not used with the 2501. If this operand is not specified, the byte containing the stacker number will be set to X'00'.

Note: If this operand is zero the following occurs: stacker one is used if the card originated in hopper 1; the highest number stacker is used if the card originated in hopper 2.

READA-address specifies the address (at a 128-byte boundary) of the buffer used for read operations. If double buffering is used, both buffers must be on 128-byte boundaries, and the two buffers must be contiguous in main storage. You must specify this operand if you plan to use this DTF for a read operation.

PUNCHA-address specifies the address (at a 128-byte boundary) of the buffer used for punch operations on the 1442, MFCU, or MFCM. If double buffering is used, both buffers must be on 128-byte boundaries, and the two buffers must be contiguous in storage. You must specify this operand if you plan to use this DTF for a punch operation.

PRINTA-address specifies the address (at a 256-byte boundary) of the buffer used for print operations on the MFCM or the MFCU. For the MFCM, the buffer size must be 64 bytes times the highest print head used. If double buffering is used, both print buffers must be contiguous in main storage. For the MFCU, the buffer size must be 128 bytes for single buffering and 256 bytes for double buffering. If two DTFs are being used for printing from both hoppers of the MFCU, the operation must be single buffered and the DTFs must use the same 256-byte print area. You must specify this operand if you plan to use this DTF for a print operation.

READL-number specifies the number of card columns to read. This operand should be used with the MFCM or 2501 only. If this operand is not specified, the maximum value (80) is assumed.

PUNCHL-number specifies the number of card columns to punch. This operand should be used with the MFCM or 1442 only. If this operand is not specified, the maximum value (80) is assumed.

PRINTL-number specifies the number of columns to print per head on the MFCM. If this operand is not specified, the maximum value (64) is assumed.

PRHEAD-mask defines the print heads selected for the MFCM. The mask must be specified as an eight-bit field. If this operand is omitted, the current setting of the print head selection byte in the DTF is not modified. The following table shows the meaning of each bit.

Bit	Meaning
0 and 1 = 0	Unused, must be zero
2 = 1	Select print head six
3 = 1	Select print head five
4 = 1	Select print head four
5 = 1	Select print head three
6 = 1	Select print head two
7 = 1	Select print head one

ALIGN-Y/N specifies whether to print in a special format on the MFCM. If Y (yes) is specified, the first 64 characters are printed with print head 1 and the next 16 characters are printed right-justified with print head 2. If N (no) is specified or if this operand is not entered, printing occurs in the normal manner.

Construct a Card Get Interface ($GETC)

The $GETC macro instruction generates the interface required to communicate with card data management when a record is being read from a card device. To use this instruction, you must construct a card DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF. If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $GETC instruction. You must also provide the labels for the necessary data management routines, via EXTRN statements in your program. The names of the data management routines for each device are:

Module Name	Device
$$MFRD	MFCU
$$MMRD	MFCM
$$ARFF	1442
$$ARRD	2501

The code generated by this macro instruction gives control to the data management routine; the routine completes execution and returns control to the generated code. If the ERR or EOF operand is specified, the generated code tests the completion code returned by data management and branches to your routine.

The format of the $GETC macro instruction is:

[Name]	$GETC	[DEV-code] [,DTF-address] [,EOF-address] [,ERR-address] [,OPC-Y/N] [,READL-number] [,CARDI-Y/N] [,STACKR-number]

DEV-code specifies the appropriate device. One of the following codes must be used: MFCU, MFCM, 1442, or 2501. If this operand is not specified, default is made to MFCU.

DTF-address indicates the address of the leftmost byte of the DTF for this file. If this operand is not specified, the address is assumed to be in register 2.

EOF-address specifies the address in your program that receives control when the end-of-file is detected. If this operand is not supplied, no code is generated to check for the end-of-file condition.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code.

Note: If ERR or EOF addresses are not specified, you should check the return code in your program to determine the outcome of the operation.

OPC-Y/N indicates whether or not the read operation code should be altered in the DTF. If Y (yes) is specified, the read/punch/print bit in the operation code is set to indicate a read. If N (no) is specified or if this operand is omitted, read/punch/print bit in the operation code is not modified.

READL-number specifies the number of columns to read from the MFCM or 2501. If this operand is not specified, the DTF remains unchanged.

CARDI Y/N specifies whether to perform a read card image operation on the 1442 or 2501. If Y (yes) is specified, the card image read bit in the operation code is set on. If N (no) is specified, the card image bit in the operation code is set off. If this operand is omitted, the status of the card image read bit in the operation code is not modified.

STACKR number specifies the stacker to be used for holding the cards after the read operation. This operand is not used with the 2501. If this operand is not given, the value in the DTF is not changed.

Construct a Card Put Interface ($PUTC)

This macro instruction generates the interface needed to communicate with card data management when punching and/or printing a card file. You must provide a DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF. You must also provide, through EXTRN statements in your program, the labels of the card data management modules necessary to perform the output operation. The names of the data management routines for each device are:

Module Name	Device
$$MFPP	MFCU
$$MMPP	MFCM
$$ARFF	1442

If you need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $PUTC instruction.

The code generated by this macro instruction gives control to the data management routine. The routine completes execution and returns control to the generated code. If the ERR operand is specified, the generated code checks the completion code for errors and branches to your error routine if errors occurred.

The format of the $PUTC macro instruction is:

[Name]	$PUTC	[DEV-code] [,DTF-address] [,OPC-code] [,DEFER-Y/N] [,PRINT4-Y/N] [,FEED-Y/N] [,STACKR-number] [,PUNCHL-number] [,PRINTL-number] [,PRHEAD-mask] [,ERR-address]

DEV-code specifies the appropriate device. One of the following codes must be used: MFCU, MFCM, or 1442. If this operand is not specified, default is made to MFCU.

DTF-address specifies the address of the leftmost byte of the DTF for this file. If this operand is not entered, the address is assumed to be in register 2.

OPC-code specifies the operation code to be used. If this operand is not specified, the status of the print/punch bits in the operation code is not changed. Valid codes are as follows:

Code	Meaning
PU	Punch
PR	Print
PP	Print and punch (This code is not allowed for MFCM.)

DEFER-Y/N specifies whether the print and/or punch operation should be deferred. If Y (yes) is specified, the defer operation bit in the operation code is set on; if N (no) is specified, the defer operation bit in the operation code is set off. If this operand is not specified, the status of the defer bit is not changed.

PRINT4-Y/N specifies whether to print on four tiers of the MFCU card. If Y (yes) is specified, the fourth tier print bit in the operation code is set on; if N (no) is specified, the fourth tier print bit is set off. If this operand is omitted, the status of the fourth tier print bit in the operation code is not changed.

FEED-Y/N specifies whether to perform a feed operation following the 1442 punch operation. Y (yes) indicates that a feed should occur after the punch; if N (no) is entered, a feed is not performed after the punch operation. If this operand is omitted, the status of the feed/no feed bit in the DTF is not changed.

STACKR-number specifies the stacker to be used for this operation. If this operand is omitted, the value of the stacker-select byte in the DTF is not changed.

PUNCHL-number specifies the number of columns to punch. If this operand is omitted, the value in the punch-length byte is not changed. This operand should be used with the MFCM or 1442 only.

PRINTL-number indicates the number of columns to be printed by each head on the MFCM. If this operand is omitted, the value in the print-length byte is not changed.

PRHEAD-mask defines the MFCM print heads selected. The mask must be specified as an eight bit field. If this operand is omitted, the current setting of the print head selection byte in the DTF is not modified. The following chart shows the meaning of each bit:

Bit	Meaning
0 and 1 = 0	Unused, must be zero
2 = 1	Select print head six
3 = 1	Select print head five
4 = 1	Select print head four
5 = 1	Select print head three
6 = 1	Select print head two
7 = 1	Select print head one

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code, and you should check the return code in your program to determine the outcome of the operation.

Construct an Interface for Reading, Punching, and Printing Cards ($GPC)

The $GPC macro instruction generates the interface required to communicate with card data management when a record is being read, punched, and/or printed on a card device. To use this macro instruction, you must construct a card DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF. If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $GPC instruction. You must also provide labels for the necessary data management routines via EXTRN statements in your program. The name of the data management routine for each device is:

Module Name	Device
$$MFFF	MFCU
$$MMFF	MFCM
$$ARFF	1442

The code generated by this macro instruction gives control to the data management routine; the routine completes execution and returns control to the generated code. If the ERR or EOF operand is specified, the generated code tests the completion code returned by data management and branches to your routine.

The format of the $GPC macro instruction is:

[Name]	$GPC	[DEV-code] [,DTF-address] [,OPC-code] [,DEFER-Y/N] [,CARDI-Y/N] [,PRINT4-Y/N] [,FEED-Y/N] [,STACKR-number] [,READL-number] [,PUNCHL-number] [,PRINTL-number] [,PRHEAD-mask] [,EOF-address] [,ERR-address]

DEV-code specifies the card device desired. The possible values for this code are: MFCU, MFCM, and 1442. If this operand is not specified, MFCU is the default value.

DTF-address specifies the address of the leftmost byte of the DTF for this file. If this operand is not entered, the address is assumed to be in register 2.

OPC-code specifies the operation code to be used. Valid codes and their meanings are:

Code	Meaning	
RD	Read only	
PU	Punch only	
PR	Print only	
PP	Punch and print	Not allowed for MFCM
RPU	Read and punch	
RPR	Read and print	
RPP	Read, punch, and print	

If this operand is not specified, the status of the read/punch/print bits in the operation code is not changed.

DEFER-Y/N specifies whether to defer the punch and/or print request. If N (no) is specified the request is not deferred; if this operand is not specified the status of the defer bit in the DTF is not changed.

CARDI-Y/N specifies whether to perform a read card image operation on the 1442 or 2501. If N (no) is specified or if this operand is omitted, a card-image read is not performed.

PRINT4-Y/N specifies whether to print three or four tiers for a print request from the MFCU. Y (yes) indicates that four tiers should be printed; if this operand is omitted or if N (no) is specified, three tiers are printed.

FEED-Y/N specifies whether or not to perform a feed operation after a 1442 punch operation. If Y (yes) is specified or if this operand is omitted, a feed is performed after the punch operation.

STACKR-number specifies the stacker to be used for this card operation. This operand is not used with the 2501. If this operand is not specified, the byte containing the stacker number is set to X'00'.

READL-number specifies the number of card columns to read. This operand should be used with the MFCM or 2501 only. If this operand is not specified, the default value is 80.

PUNCHL-number specifies the number of card columns to punch. This operand should be used with the MFCM or 1442 only. If this operand is not specified, the default value is 80.

PRINTL-number specifies the number of columns to print per head on the MFCM. If this operand is not specified, the default value is 64.

PRHEAD-mask defines the print heads selected on the MFCM. The mask must be specified as an eight bit field. If this operand is omitted, the current setting of the print head selection byte in the DTF is not modified. The following table shows the meaning of each bit:

Bit	Meaning
0 and 1 = 0	Unused, must be zero
2 = 1	Select print head six
3 = 1	Select print head five
4 = 1	Select print head four
5 = 1	Select print head three
6 = 1	Select print head two
7 = 1	Select print head one

EOF-address specifies the address in your program that receives control when the end-of-file is detected. If this operand is not supplied, no code is generated to check for the end-of-file condition.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code.

Note: If ERR or EOF addresses are not specified, you should check the return code in your program to determine the outcome of the operation.

Printer Support

This section describes the macro instructions that support the printers. The following functions are provided:

- Build a pre-open DTF for a printer and assign its offsets. The DTF provides information to printer data management routines that perform input/output operations.
- Build the interface needed to print data.

Define the File for Printer ($DTFP)

The DTF provides information needed to allocate, open, and access a printer. This macro instruction generates the code that builds the printer DTF.

The format of the $DTFP macro instruction is:

[Name]	$DTFP	RCAD-address,IOBA-address,IOAA-address, OVFL-number,PAGE-number [,DEV-code] [,UP-mask] [HUC-Y/N] [,CHN-address] [,PRINT-Y/N] [,SKIPB-number] [SPACEB-number] [,SKIPA-number] [,SPACEA-number] [,RECL-number]

RCAD-address is a required operand which gives the address of the leftmost byte of the logical record.

IOBA-address is a required operand specifying the address of the leftmost byte of the IOB. The IOB will be 50 bytes long.

IOAA-address is a required operand which specifies the address of the leftmost byte of the I/O area. For the 1403, this address must define the I/O area as beginning on a 124-byte boundary. The length of the I/O area must be:

- 132 bytes for the 1403 printer
- The record length +7 for the 3284 printer.

OVFL-number specifies the line number on the printer after which the overflow completion code will be returned. If this operand is not specified, default is made to 6 lines less than the number specified for the PAGE operand.

PAGE-number specifies the number of lines to print per page. If this operand is not specified, default is made to the system value for the number of lines per page.

DEV-code specifies the printer desired. The possible values for this code are 1403 and 3284. If this operand is not specified, 1403 is assumed.

UP-mask specifies the mask to test the eight external indicators. The code for the UP-mask must be specified as 8 binary bits. For example, to test bits 0, 3, 5, and 7, you would enter UP-10010101. The UP-mask is compared to the external indicators set on by the SWITCH statement for conditionally opening files. If the bits that are on in the UP-mask are also on in the external indicators set on by the SWITCH statement, the file will be opened. If the UP-mask is all zeroes or not used, the file will be unconditionally opened.

Note: Information on setting external indicators (SWITCH statement) can be found in the *IBM System/3 Model 15 System Control Programming Reference Manual* for Program Number 5704-SC1, GC21-5077 and *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual* for Program Number 5704-SC2, GC21-5162.

HUC-Y/N specifies whether to halt if an unprintable character is detected. If N (no) is specified or if this operand is omitted, no halt occurs.

CHN-address indicates the address of the next DTF in the chain of DTFs. If there is no DTF chain or if this is the last DTF in a chain, this operand should be omitted (a value of X'FFFF' is assumed).

PRINT-Y/N specifies whether to perform a print and a skip or space or only a skip or space. Default is N (no), meaning that a print is not performed.

SKIPB-number specifies the line to skip to before the print operation. If this operand is not entered, the default value is zero.

SPACEB-number specifies the number of lines (maximum of 3 lines) to space before the print operation. If this operand is not entered, the default value is zero.

SKIPA-number specifies the line to be skipped to after the print operation. If this operand is not specified, the default value is zero.

SPACEA-number specifies the number of lines to space (0, 1, 2, or 3) after the print operation. If this operand is not specified, it defaults to a value of zero if DEV-1403 is specified or to a value of one if DEV-3284 is specified. A space after of zero is not allowed for the 3284 printer, and, if zero is specified, the operand defaults to a space after of one.

RECL-number specifies the length of the line to be printed. If this operand is omitted, default is 132 positions.

Construct a Printer Put Interface ($PUTP)

This macro instruction generates the interface needed to communicate with printer data management. You must provide a DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF. You must also provide, through an EXTRN statement in your program, the label $$LPRT, for the 1403, or $$LPMP, for the 3284. (These labels are for the printer data management module necessary to perform the printer output operation.)

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $PUTP macro instruction.

The code generated by this macro instruction gives control to the data management routine. The routine completes execution and returns control to the generated code. If the ERR operand is specified, the generated code checks the completion code for errors and branches to your error routine if errors occurred.

The format of the $PUTP macro instruction is:

[Name]	$PUTP	[DEV-code] [,DTF-address] [,PRINT-Y/N] [,SKIPB-number] [,SPACEB-number] [SKIPA-number] [,SPACEA-number] [,ERR-address] [,OVFL-address]

DEV-code specifies the printer device desired. The possible values for this code are 1403 and 3284. If this operand is not specified, 1403 is the default value.

DTF-address specifies the address of the leftmost byte of the DTF for this file. If this operand is not entered, the address is assumed to be in register 2.

PRINT-Y/N specifies whether to perfrom a print and a skip or space or only a skip or space. If this operand is not specified, the DTF remains unchanged.

SKIPB-number specifies the line to skip to before the print operation. If this operand is not entered, the DTF remains unchanged.

SPACEB-number specifies the number of lines to space before the print operation. If this operand is not entered, the DTF remains unchanged.

SKIPA-number specifies the line to be skipped to after the print operation. If this operand is not entered, the DTF remains unchanged.

SPACEA-number specifies the number of lines to space after the print operation. If this operand is not entered, the DTF remains unchanged.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code, and you should check the return code in your program to determine the outcome of the operation.

OVFL-address specifies the address in your program that should receive control if page overflow occurs.

Disk Device Support

This section describes the macro instructions that support disk devices. The following functions are provided:

- Build a pre-open DTF for disk GET/PUT operations and assign its offsets.
- Build an input/output block (IOB) for disk read/write operations and assign its offsets.
- Build the interfaces required to get input records from a disk device via a get or a read.
- Build the interfaces required to put output records to a disk device via a put or a write.
- Build the interface to wait for disk completion.

The disk DTFs provide information to the disk data management, and the disk IOBs provide information to the input/output supervisor routines that perform the input or output operations. These operations are provided through the disk support macro instructions.

The IBM 3340 Direct Access Storage Facility attaches to System/3 Models 15B, 15C, and 15D. Also, the IBM 3344 Direct Access Storage attaches to System/3 Model 15D.

Certain areas on the 3340 and 3344 disks are treated as 5444 disks. These areas, known as *simulation areas*, are used for program libraries and can also be used for data files. These areas cannot contain multivolume or indexed files. The remainder of the disk space, known as *main data areas,* can only be used for data files.

Reference in this manual to 5444, 5445, and 3340 are to be interpreted according to which disk storage device(s) is attached to the system. The following table should be used to determine the meaning of the reference:

References to D3 and D4 Q-numbers in this manual may be replaced with D31 and D41 for 5704-SC2.

Reference	Model 15A Meaning	Model 15B and 15C Meaning	Model 15D Meaning
5444	5444 Disk Storage Drive	Simulation area on 3340	Simulation area on 3340 or 3344
5445	5445 Disk Storage	Main data area on 3340	Main data area on 3340 or 3344
3340	Not applicable	Main data area on 3340	Main data area on 3340 or 3344

For Program Number 5704-SC2, there are certain restrictions on where the DTFs and associated fields can be located relative to 40K (X'E000') in the user partition:

- DTFs must be located such that the entire DTF resides at an address less than 40K for batch disk files specified as SHARE-YES and for CCP disk files.
- For the following disk access methods where SHARE-YES is specified, the high add key and the high prime key areas (ADKEY parameter) must be located completely below 40K:
  - Indexed output add
  - Indexed random input and add
  - Indexed random input, update and add
- If a disk file is specified with external buffers, then the following must be located completely below 40K:
  - The DTFs
  - The index IOBs (IO parameter)
  - The master track index (MSTX parameter)
  - The space for the multivolume extent table (IO parameter, M_1 and M_2 definitions)

Define the File for Disk ($DTFD)

The DTF provides information needed to allocate, open, and access a file on the disk device. This macro instruction generates the code that builds the disk DTF. See *Appendix B: Define the File Control Blocks* for a description of the pre-open and post-open disk DTFs.

The format of the $DTFD macro instruction is:

[Name]	$DTFD	AC-code,RECL-number,NAME-filename, BLKL-number,IO-address [,DISK-5444/5445/3340] [,UP-mask] [,BUFNO-1/2] [,MVF-N/Y] [,LIM-N/Y] [,ORD-N/Y] [,BIN-N/Y] [,CHN-address] [,RCAD-address] [,ENT-number] [,MVFN-number] [,KEYL-number] [,KEYD-number] [,KEYA-address] [,MVFT-address] [,MSTX-address] [,IBLKL-number] [,ADKEY-address] [,EOVK-address] [,SHR-Y/N][1] [,EXTBUF-Y/N][1]

[1] These parameters are valid only for Program Number 5704-SC2.

AC-code specifies the access method used for the file. This operand is required. The codes and their meanings are as follows:

Code	Access Method
CA	Consecutive add
CG	Consecutive get
CO	Consecutive output
CU	Consecutive update
DG	Direct get
DO	Direct output
DU	Direct update
IA	Indexed add
IO	Indexed output
IS	Indexed sequential get
ISA	Indexed sequential add
ISL	Indexed sequential input with limits
ISU	Indexed sequential update
ISUA	Indexed sequential update and add
ISUL	Indexed sequential input update with limits
IR	Indexed random get
IRA	Indexed random add
IRU	Indexed random update
IRUA	Indexed random update and add

RECL-number specifies the decimal length of the logical record. This operand must be specified.

NAME-file name specifies the name of the file. The name must be eight characters or less in length. This operand must be specified.

BLKL-number specifies the number of bytes in the buffer. The minimum number can be determined as follows, except for the access methods listed on the following pages, for which the minimum number of bytes is 256:

- If the record length is less than or equal to 256 and evenly divisible into 256, the buffer length is 256.
- If the record length is greater than 256 and a multiple of 256, the buffer length is equal to the record length.
- If the record length is not evenly divisible into 256 and not a multiple of 256, the buffer length is the multiple of 256 that is next higher than the record length plus 255.
- If the record length is an odd multiple of 128, the buffer should be the record length plus 128.

Note: These buffer lengths are minimum lengths. Larger lengths may be specified, but must be in multiples of 256.

The following access methods can always operate in a minimum of a 256-byte buffer:

Consecutive output
Consecutive add
Consecutive output multivolume
Consecutive add multivolume
Consecutive input
Consecutive input multivolume
Indexed output
Indexed output multivolume

IO-address provides the address of the leftmost byte of an area in main storage allocated to contain all buffers and IOBs for the access method. This operand must be specified. The amount of main storage required is shown in the following chart:

Access Method	Formula	EXTBUF-Y (Space in I/O area)	Formula for External Buffers on Option Statement
Consecutive, Direct	(BLKL + 30) times BUFNO	36 times BUFNO	BLKL times BUFNO
Indexed Output	(BLKL + 256 + 60) times (BUFNO) + KEYL for MVF	72 times BUFNO + KEYL for MVF	BLKL + 256 times BUFNO
Indexed Random Input, or Indexed Random Input and Update	(BLKL + 30) + (258 + 30) + M_1, where M_1 = 0 for single volume files M_1 = 84 + KEYL for multivolume files (5704-SC1) M_1 = 112 + KEYL for multivolume files (5704-SC2)	72 + M_1, where M_1 = 0 for single volume files M_1 = 84 + KEYL for multivolume files (5704-SC1) M_1 = 112 + KEYL for multivolume files (5704-SC2)	BLKL + 512
Indexed Random Input and Add, or Indexed Random Input, Update, and Add	2 times (BLKL + 30) + (258 + 30) + 256 times (BLKL + 30 + M_2, where M_2 = 0 for single volume files M_2 = 180 + KEYL for multivolume files	144 + M_2, where M_2 = 0 for single volume files M_2 = 180 + KEYL for multivolume files	2 times BLKL + 512 + (256 times IBLKL)
Indexed Add (without input)	(BLKL + 30) + (258 + 30) + 256 times IBLKL + 30	108	BLKL + 512 + (256 times IBLKL)
Indexed Sequential Input, or Indexed Sequential Input and Update	(BLKL + 30) + 256 times IBLKL + 30	72	BLKL + (256 times IBLKL)
Indexed Sequential Input and Add, or Indexed Sequential Input Update, and Add	2 times (BLKL + 30) + 2 times (256 times IBLKL + 30)	144	2 times BLKL + 2 times (256 times IBLKL)
Indexed Sequential/limits Input, or Indexed Sequential/Limits Input and Update	(BLKL + 30) + (258 + 30) + 256 (BLKL + 1)	72	BLKL + 256 + 256 (BLKL)

DISK-5444/5445/3340 specifies whether the disk device is the 5444 Disk Storage Drive, the 5445 Disk Storage, or the 3340 Direct Access Storage Facility. If this operand is not specified, 5444 is assumed.

UP-mask specifies the mask to test the eight external indicators. The code for the UP-mask must be specified as 8 binary bits. For example, to test bits 0, 3, 5, and 7, you would enter UP-10010101. The UP-mask is compared to the external indicators set on by the SWITCH statement for conditionally opening files. If the bits that are on in the UP-mask are also on in the external indicators set on by the SWITCH statement, the file will be opened. If the UP-mask is all zeroes or not used, the file will be unconditionally opened.

Note: Information on setting external indicators (SWITCH statement) can be found in the *IBM System/3 Model 15 System Control Programmming Reference Manual* (for Program Number 5704-SC1), GC21-5077, and in the *IBM System/3 Model 15 Control Programming Concepts and Reference Manual* (for Program Number 5704-SC2), GC21-5162.

BUFNO-1/2 allows you to specify either one or two buffers for the file. You can use two buffers only with consecutive access methods and indexed output. All consecutive access methods allow dual buffering except the consecutive update and consecutive update multivolume. If this operand is omitted or if SHR-Y is specified, one buffer is assumed.

MVF-N/Y specifies whether the access method is multivolume. If this operand is omitted, N (no) is assumed.

LIM-N/Y is specified only for indexed sequential get and indexed sequential update. It specifies whether the sequential access is within limits. If this operand is not entered, N (no) is assumed.

ORD-N/Y specifies whether an ordered load is to be used with the indexed output access method. This operand can be specified only with the indexed output access method. ORD-Y must be specified for indexed multivolume output access methods. If this operand is not entered, N (no) is assumed.

BIN-N/Y is specified only with the direct output, direct get, and direct update access methods. Y (yes) indicates direct binary relative record numbers; N (no) indicates direct decimal relative record numbers. If this operand is omitted, N is assumed.

CHN-address specifies the address of the next DTF in the chain of DTFs. If there is no DTF chain or if this DTF is the last DTF in the chain, this operand should be omitted and X'FFFF' assumed.

RCAD-address specifies the address of the leftmost byte of the logical record. If this operand is not entered, X'0000' is assumed. Depending on the disk access method being used for an input operation, either move mode or locate mode is used. If move mode is used, the record is provided at the address specified in the RCAD parameter. If locate mode is used, the address of the input record is contained at the displacement of $DFLRA in the DTF.

The specified address plus the total length of the area must be less than logical X'E000' for files that are being shared or for files that are being used for external buffers (5704-SC2 only).

ENT-number specifies the number of entries in the master track index. This operand is specified only for indexed random or indexed add access methods.

MVFN-number indicates the number of volumes for a multivolume direct access method. This operand must be specified for these access methods.

KEYL-number specifies the length of the key field and must be used for all indexed access methods, but no others. The key field length can be no more than 29 bytes.

KEYD-number is entered for all indexed access methods. It indicates the displacement into the record of the rightmost byte of the key field. The displacement of the first byte in the record is zero, the second byte is one, and so on.

KEYA-address specifies one of the following and is a required operand for these access methods:

- Main storage address of the leftmost byte of the key field for indexed random access methods.
- Main storage address of the leftmost byte of the relative record number field for direct access methods.
- Main storage address of the leftmost byte of the save area for current and last keys for indexed sequential add access methods.
- Main storage address of the leftmost byte of the save area for high and low keys for indexed sequential with limits access methods (LIM-Y). The specified address plus the total length of the area must be less than logical X'E000' for files that are being shared or for files that are being used for external buffers (5704-SC2 only).

You must allocate the main storage space for the fields. The amount of space required is:

- The number of bytes in the key field for indexed random access methods.
- 23 bytes for direct access methods with decimal keys. The decimal key is located in the rightmost 15 bytes of the field.
- 8 bytes for direct access methods with binary keys. The binary key is located in the rightmost 3 bytes of the field.
- Two times the key length for indexed sequential add or indexed sequential with limits access methods. The low key is located in the left half of the field; the high key in the right half.

MVFT-address must be specified for all multivolume direct files, and only for the access methods used with these files. This operand specifies the address of the leftmost byte of the table of extents used for the access methods used with these files. You must allocate main storage space for the table. The number of bytes allocated must be equal to seven times the number of volumes in the file. With 5704-SC2, if the multivolume file is to be shared, the number of bytes allocated must be ten times the number of volumes in the file. The specified address plus the total length of the area must be less than logical X'E000' for files that are being shared or for files that are being used for external buffers (5704-SC2 only).

MSTX-address specifies the address of the leftmost byte of the master track index in main storage. This operand must be specified for indexed random and indexed add access methods. You must allocate space in main storage for the master track index. The length of the master track index is determined by the following formulas:

- For single volume random access methods,
  Length = ENT (key length + 2)
- For multivolume random access methods,
  Length = ENT (keylength + 2)
  ENT must be equal to or greater than 4.

The specified address plus the total length of the area must be less than logical X'E000' for files that are being shared or for files that are being used for external buffers (5704-SC2 only).

IBLKL-number specifies the number of sectors in the index buffer. This operand is used only with indexed access methods. If this operand is not specified, one sector is assumed. For indexed sequential access methods with both input and add, the specified index buffer size applies to both the input index buffer and the add index buffer. For indexed random access methods, the specified buffer size applies only to the add index buffer. Increasing the size of this index buffer will increase the efficiency of processing for a random file.

ADKEY-address specifies the address of the leftmost byte of an area used to save the highest key in the prime index and the highest key in the add index. The area at this address must be equal in length to 2 times KEYL for each volume used by the file. This operand is required for indexed add, indexed random input and add, and indexed random input, update, and add access methods. It is ignored if specified with other access methods. The specified address plus the total length of the area must be less than logical X'E000' for files that are being shared or for files that are being used for external buffers (5704-SC2 only).

EOVK-address specifies the address of the leftmost byte of an area used to save a key if force end-of-volume is specified (by the HIKEY parameter on the FILE OCL statement) for a multivolume indexed output file. The area at this address must be equal in length to KEYL. This operand is used only when the multivolume indexed output access method is used and force end-of-volume will be used. The specified address plus the total length of the area must be less than logical X'E000' for files that are being shared or for files that are being used for external buffers (5704-SC2 only).

SHR-Y/N, applicable to Program Number 5704-SC2 only, allows the user to specify whether or not file sharing is permitted on the file. File sharing is permitted, when possible, if this parameter is not used.

EXTBUF-Y/N, applicable to Program Number 5704-SC2 only, allows the user to have external buffers. If this parameter is not used, external buffering does not occur.

Construct a Disk Get Interface ($GETD)

The $GETD macro instruction generates the interface needed to communicate with disk data management when a record is being read from a disk file. To use this macro instruction, construct a disk DTF for the file and use the $DTFO macro instruction to establish the offsets for the DTF. You must also provide the labels for the necessary data management routines through EXTRN statements in your programs. The names of the data management modules and the functions of the modules are shown in Figure 11. If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $GETD macro instruction.

The code generated by this macro instruction gives control to the data management routine; the routine completes execution and returns control to the generated code. The generated code tests the completion codes returned by data management.

[Name]	$GETD	{[AC-code] / [EBAC-code]} [,DTF-address] [,ERR-address] [,EOF-address] [,NFR-address] [,LSTV-address] [,NOKY-address]

AC-code or *EBAC-code* specifies the appropriate access method for the file. The *EBAC-code* is for external buffering and is valid only for Program Number 5704-SC2. One of these parameters must be specified.

The codes that must be used for the *AC-code* and the *EBAC-code* parameters are shown in Figure 11.

DTF-address indicates the address of the leftmost byte of the DTF for this file. If this operand is not specified, the address is assumed to be in register 2.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code.

EOF-address specifies the address in your program that receives control when the end-of-file is detected. If this operand is not supplied, no code is generated to check for the end-of-file condition. You must not use this operand with random or direct access methods.

Note: If ERR or EOF addresses are not specified, you should check the return code in your program to determine the outcome of the operation.

NRF-address must be used only for random and direct access methods. It specifies the address in your program that is to receive control when a no-record-found condition occurs.

LSTV-address is used when processing a random, offline, multivolume file. This operand supplies the address in your program which receives control when the requested key is too high for the final volume in a multivolume file.

NOKY-address supplies the address in your program that is to receive control under either of the following conditions:

- The requested key is too low for the current volume when processing an indexed random offline multivolume file.
- The requested key is too high for any volume when processing an indexed random online multivolume file.

This operand is not used with other access methods.

Construct a Disk Put Interface ($PUTD)

The $PUTD macro instruction generates the interface needed to communicate with disk data management when putting a record to disk or updating a previously retrieved record. You must provide a DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF. You must also provide, through EXTRN statements in your program, the labels of the disk data management modules necessary to perform the output operation (see Figure 11). If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $PUTD macro instruction.

$GETD	$PUTD	AC-code 5444	AC-code 5445	Move/ Locate Mode	System Module 5444	System Module 5445	Access Method
	X	CA	CA5	M	$$CSOP	$$CFOP	Consecutive Add
	X	CAM	CAM5	M	$$CSOM	$$CFOM	Consecutive Add Multivolume
X		CG	CG5	M	$$CSIP	$$CFIP	Consecutive Get
X		CGM	CGM5	M	$$CSIM	$$CFIM	Consecutive Get MVF
	X	CO	CO5	M	$$CSOP	$$CFOP	Consecutive Output
	X	COM	COM5	M	$$CSOM	$$CFOM	Consecutive Output MVF
X	X	CU	CU5	L	$$CSUP	$$CFUP	Consecutive Update
X	X	CUM	CUM5	L	$$CSUM	$$CFUM	Consecutive Update MVF
X		DG	DG5	L	$$DAID	$$DFID	Direct Get
X		DGA	DGA5	L	$$DAIB	$$DFIB	Direct Get (Binary Keys)
X		DGAM	DGAM5	L	$$DAIT	$$DFIT	Direct Get (Binary Keys) MVF
X		DGM	DGM5	L	$$DAIM	$$DFIM	Direct Get MVF
X	X	DO	DO5	L	$$DAUD	$$DFUD	Direct Output
X	X	DOA	DOA5	L	$$DAUB	$$DFUB	Direct Output (Binary Keys)
X	X	DOAM	DOAM5	L	$$DAUT	$$DFUT	Direct Output (Binary Keys) MVF
X	X	DOM	DOM5	L	$$DAUM	$$DFUM	Direct Output MVF
X	X	DU	DU5	L	$$DAUD	$$DFUD	Direct Update
X	X	DUA	DUA5	L	$$DAUB	$$DFUB	Direct Update (Binary Keys)
X	X	DUAM	DUAM5	L	$$DAUT	$$DFUT	Direct Update (Binary Keys) MVF
X	X	DUM	DUM5	L	$$DAUM	$$DFUM	Direct Update MVF
	X	IA	IA5	M	$$IOAD	$$IFAD	Indexed Add
	X	IAM	IAM5	M	$$IOAM	$$IFAM	Indexed Add MVF
	X	IO	IO5	M	$$IOUT	$$IFUT	Indexed Output
	X	IOM	IOM5	M	$$IOUM	$$IFUM	Indexed Output MVF
X		IR	IR5	L	$$IRIP	$$IGIP	Indexed Random Input
X	X	IRA	IRA5	L	$$IRAD	$$IGAD	Indexed Random Add
X	X	IRAM	IRAM5	L	$$IRAM	$$IGAM	Indexed Random Add MVF
X	X	IRBM	IRBM5	L	$$IRBM	$$IGBM	Indexed Random Update & Add MVF
X		IRM	IRM5	L	$$IRIM	$$IGIM	Indexed Random Input MVF
X	X	IRU	IUR5	L	$$IRUP	$$IGUP	Indexed Random Update
X	X	IRUA	IRUA5	L	$$IRUA	$$IGUA	Indexed Random Update & Add
X	X	IRUM	IRUM5	L	$$IRUM	$$IGUM	Indexed Random Update MVF
X		IS	IS5	M	$$ISIP	$$IHIP	Indexed Sequential Input
X	X	ISA	ISA5	M	$$ISAD	$$IHAD	Indexed Sequential Add
X	X	ISAM	ISAM5	M	$$ISAM	$$IHAM	Indexed Sequential Add MVF
X	X	ISBM	ISMB5	M	$$ISBM	$$IHBM	Indexed Sequential Update & Add MVF
X		ISL	ISL5	M	$$ISIL	$$IHIL	Indexed Sequential Input Within Limits
X		ISM	ISM5	M	$$ISIM	$$ISHIM	Indexed Sequential Input MVF
X	X	ISU	ISU5	L	$$ISUP	$$IHUP	Indexed Sequential Update
X	X	ISUL	ISUL5	L	$$ISUL	$$IHUL	Indexed Sequential Update Within Limits
X	X	ISUM	ISUM5	L	$$ISUM	$$IHUM	Indexed Sequential Update MVF
X	X	ISUA	ISUA5	M	$$ISUA	$$IHUA	Indexed Sequential Update & Add
X		ISLM	ISLM5	M	$$ISIB	$$IHIB	Indexed Sequential Input Within Limits MVF
X	X	ISUB	ISUB5	M	$$ISUB	$$IHUB	Indexed Sequential Update Within Limits MVF
X		ISC	ISC5	M	$$ISIC	$$IHIC	Indexed Sequential Input Within Changeable Limits
X		ISCM	ISCM5	M	$$ISID	$$IHID	Indexed Sequential Input Within Changeable Limits MVF
X	X	ISUC	ISUC5	M	$$ISUC	$$IHUC	Indexed Sequential Update Within Changeable Limits
X	X	ISUD	ISUD5	M	$$ISUD	$$IHUD	Indexed Sequential Update Within Changeable Limits MVF

Figure 11 (Part 1 of 2). Disk Data Management Modules Without External Buffering

$GETD	$PUTD	AC-code 5444	AC-code 5445	Move/ Locate Mode	System Module 5444	System Module 5445	Access Method
	X	CA	CA5	M	$$WSOP	$$WFOP	Consecutive Add
	X	CAM	CAM5	M	$$WSOM	$$WFOM	Consecutive Add Multivolume
X		CG	CG5	M	$$WSIP	$$WFIP	Consecutive Get
X		CGM	CGM5	M	$$WSIM	$$WFIM	Consecutive Get MVF
	X	CO	CO5	M	$$WSOP	$$WFOP	Consecutive Output
	X	COM	COM5	M	$$WSOM	$$WFOM	Consecutive Output MVF
X	X	CU	CU5	M	$$WSUP	$$WFUP	Consecutive Update
X	X	CUM	CUM5	M	$$WSUM	$$WFUM	Consecutive Update MVF
X		DG	DG5	M	$$YAID	$$YFID	Direct Get
X		DGA	DGA5	M	$$YAIB	$$YFIB	Direct Get (Binary Keys)
X		DGAM	DGAM5	M	$$YAIT	$$YFIT	Direct Get (Binary Keys) MVF
X		DGM	DGM5	M	$$YAIM	$$YFIM	Direct Get MVF
X	X	DO	DO5	M	$$YAUD	$$YFUD	Direct Output
X	X	DOA	DOA5	M	$$YAUB	$$YFUB	Direct Output (Binary Keys)
X	X	DOAM	DOAM5	M	$$YAUT	$$YFUT	Direct Output (Binary Keys) MVF
X	X	DOM	DOM5	M	$$YAUM	$$YFUM	Direct Output MVF
X	X	DU	DU5	M	$$YAUD	$$YFUD	Direct Update
X	X	DUA	DUA5	M	$$YAUB	$$YFUB	Direct Update (Binary Keys)
X	X	DUAM	DUAM5	M	$$YAUT	$$YFUT	Direct Update (Binary Keys) MVF
X	X	DUM	DUM5	M	$$YAUM	$$YFUM	Direct Update MVF
	X	IA	IA5	M	$$XOAD	$$XFAD	Indexed Add
	X	IAM	IAM5	M	$$XOAM	$$XFAM	Indexed Add MVF
	X	IO	IO5	M	$$XOUT	$$XFUT	Indexed Output
	X	IOM	IOM5	M	$$XOUM	$$XFUM	Indexed Output MVF
X		IR	IR5	M	$$XRIP	$$XGIP	Indexed Random Input
X	X	IRA	IRA5	M	$$XRAD	$$XGAD	Indexed Random Add
X	X	IRAM	IRAM5	M	$$XRAM	$$XGAM	Indexed Random Add MVF
X	X	IRBM	IRBM5	M	$$XRBM	$$XGBM	Indexed Random Update & Add MVF
X		IRM	IRM5	M	$$XRIM	$$XGIM	Indexed Random Input MVF
X	X	IRU	IUR5	M	$$XRUP	$$XGUP	Indexed Random Update
X	X	IRUA	IRUA5	M	$$XRUA	$$XGUA	Indexed Random Update & Add
X	X	IRUM	IRUM5	M	$$XRUM	$$XGUM	Indexed Random Update MVF
X		IS	IS5	M	$$XSIP	$$XHIP	Indexed Sequential Input
X	X	ISA	ISA5	M	$$XSAD	$$XHAD	Indexed Sequential Add
X	X	ISAM	ISAM5	M	$$XSAM	$$XHAM	Indexed Sequential Add MVF
X	X	ISBM	ISMB5	M	$$XSBM	$$XHBM	Indexed Sequential Update & Add MVF
X		ISL	ISL5	M	$$XSIL	$$XHIL	Indexed Sequential Input Within Limits
X		ISM	ISM5	M	$$XSIM	$$XSHIM	Indexed Sequential Input MVF
X	X	ISU	ISU5	M	$$XSUP	$$XHUP	Indexed Sequential Update
X	X	ISUL	ISUL5	M	$$XSUL	$$XHUL	Indexed Sequential Update Within Limits
X	X	ISUM	ISUM5	M	$$XSUM	$$XHUM	Indexed Sequential Update MVF
X	X	ISUA	ISUA5	M	$$XSUA	$$XHUA	Indexed Sequential Update & Add
X		ISLM	ISLM5	M	$$XSIB	$$XHIB	Indexed Sequential Input Within Limits MVF
X	X	ISUB	ISUB5	M	$$XSUB	$$XHUB	Indexed Sequential Update Within Limits MVF
X		ISC	ISC5	M	$$XSIC	$$XHIC	Indexed Sequential Input Within Changeable Limits
X		ISCM	ISCM5	M	$$XSID	$$XHID	Indexed Sequential Input Within Changeable Limits MVF
X	X	ISUC	ISUC5	M	$$XSUC	$$XHUC	Indexed Sequential Update Within Changeable Limits
X	X	ISUD	ISUD5	M	$$XSUD	$$XHUD	Indexed Sequential Update Within Changeable Limits MVF

Figure 11 (Part 2 of 2). Disk Data Management Modules With External Buffering (5704-SC2 only)

The code generated by this macro instruction gives control to the data management routine; the routine completes execution and returns control to the generated code. Completion codes are tested and control is returned to your program.

The format of the $PUTD macro instruction is:

[Name]	$PUTD	{[AC-code] / [EBAC-code]} [,DTF-address] [,ERR-address] [,EOX-address] [,DUP-address] [,SERR-address] [,KERR-address] [,UPD-Y/N] [,LSTV-address] [,NOKY-address] [,HKER-address]

AC-code or *EBAC-code* specifies the appropriate access method for the file. The *EBAC-code* is for external buffering and is valid only for Program Number 5704-SC2. One of these parameters must be specified.

The codes that must be used for the *AC-code* and the *EBAC-code* parameters are shown in Figure 11.

DTF-address specifies the address of the DTF associated with this file. If this operand is not specified, the address is assumed to be in register 2.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code.

EOX-address supplies the address in your program that is to receive control when an end of extent is reached during the operation. This operand is entered only when creating a consecutive or indexed file or when records are to be added to the file.

DUP-address provides the address in your program that is to receive control when an attempt to add a duplicate record has occurred. This operand is used only with an add access method.

SERR-address is the address in your program where control is passed in the event of a sequence error while loading an indexed file.

KERR-address specifies the address of your routine to be called when an attempt has been made to update a record in an indexed file and the attempt would destroy the record key.

UPD-Y/N indicates whether an update is to be performed. If this operand is not entered, N (no) is assumed.

LSTV-address specifies the address in your program that receives control when a requested key is too high for the last specified volume. This operand is used only when processing an indexed, random, offline, multivolume file.

NOKY-address supplies the address in your program that is to receive control under either of the following conditions:

- The requested key is too low for the current volume when processing an indexed, random, offline, multivolume file.
- The requested key is too high for any volume when processing an indexed random online multivolume file.

This operand is not used with other access methods.

HKER-address specifies the address in your program that is to receive control when an indexed sequential add multivolume is attempted and the requested key is higher than any key presently in the file, but lower than the highest permissible key.

Input/Output Block for Disk ($IOBD)

This macro instruction generates a disk input/output block (IOB) for use by the disk input/output supervisor. A 30-byte IOB is generated for 5444, 5445, and 3340 disk devices. For a detailed description of the disk IOB, see *Appendix C: Disk Input/Output Block.*

The format of the $IOBD macro instruction is:

[Name]	$IOBD	[DISK-5444/5445/3340] [,CYL-number] [,SCTR-number] [,HEAD-number] [,NUM-number] [,BUFF-address] [,Q-number] [,ERREC-IOS/USER] [,LOG-Y/N] [,VER-Y/N] [,CHN-address]

DISK-5444/5445/3340 specifies whether the disk device being used is the 5444 Disk Storage Drive Model 1, 5445 Disk Storage, or the 3340 Direct Access Storage Facility. If a device is not specified, 5444 Disk Storage Drive Model 1 is assumed.

CYCL-number indicates the beginning cylinder to be accessed. You can specify the cylinder by a decimal number (0-199) or a hexadecimal number (X'00'-X'C7'). If this operand is not entered, X'FF' is assumed. You must then insert the correct number into the IOB before performing the input/output operation. This can be done through the macro used to initiate the I/O operation.

SCTR-number specifies the first sector to be accessed. The number specified must be a decimal from 1 through 48 for the 5444 disk drive, from 1 through 20 for the 5445 disk storage, or from 1 to 48 for the 3340 disk drive. If this operand is not entered, X'FF' is assumed. You must then insert the correct number before performing the input/output operation. You can specify the sector through the $RDD and $WRTD macro instructions.

HEAD-number is specified only for the 5445 or 3340 disk storage drives. It specifies the head to be used with the cylinder and sector when an I/O operation is performed. The number specified may be decimal (0-19) or hexadecimal (X'00'-X'13'). If this operand is omitted, X'FF' is assumed and the value must be updated when the I/O operation is performed.

NUM-number specifies the number of sectors used. You may specify the number in either decimal or hexadecimal form. If this operand is not entered, one sector is assumed.

BUFF-address is the address of the leftmost byte of your data area. If this operand is omitted, X'FFFF' is assumed, and you must update the IOB before performing the input/output operation.

Q-number specifies the drive on which the record is located. You may specify the disk drive alone F1, R1, F2, R2, D1, D2, D3/D31, D32, D33, D34, D4/D41, D42, D43, and D44, or you may specify the hexadecimal Q-code in the form Q-X'nn', where nn is a valid hexadecimal Q-code. The valid Q-codes are shown in Figure 12. If you specify only the disk drive, you must set the read/write bits (the last four bits of the Q-code) before you can perform the I/O operation. This can be done through the $RDD or $WRTD macro instructions.

Note: D31, D32, D33, D34, D41, D42, D43, and D44 are for 5704-SC2 only.

I/O Operation	Q-Byte Setting (Hex)			
	Drive 1	Drive 2	Drive 3	Drive 4
5444 Removable Disk				
Control	A0	B0	—	—
Read	A1	B1	—	—
Write	A2	B2	—	—
Scan	A3	B3	—	—
5444 Fixed Disk				
Control	A8	B8	—	—
Read	A9	B9	—	—
Write	AA	BA	—	—
Scan	AB	BB	—	—
5445 Disk				
Control	C0	C8	D0	D8
Read	C1	C9	D1	D9
Write	C2	CA	D2	DA
Scan	C3	CB	D3	DB

Figure 12. Q-Byte Hexadecimal Settings[1]

ERREC-IOS/USER indicates whether the input/output supervisor is to handle error recovery. If you specify IOS, the supervisor handles error recovery and retries the operation when errors occur. If you specify USER, the supervisor does not retry the operation and returns control to you. If this operand is not specified, IOS is assumed.

LOG-Y/N indicates whether the I/O supervisor is to log errors that occur during the operation. If you specify Y (yes), error conditions are logged on the system pack. This information is used by IBM customer engineers. N (no) indicates no logging is to be done for this IOB. If this operand is not entered, Y is assumed.

VER-Y/N is used for output operations. Y (yes) indicates the written data should be verified; N (no) indicates it should not. If this operand is omitted, Y is assumed.

CHN-address specifies the address of the leftmost byte of the next IOB for the operation if more than one IOB is required.

[1] Figure 12 applies *only* to 5704-SC1. For 5704-SC2, you can specify *only* the disk drive (F1, R1, etc.) with the Q parameter, and read/write bits are set with $RDD or $WRTD.

Input/Output Block Offsets ($IOED)

This macro instruction generates equates to establish labels for the disk IOBs. These labels are offsets from the beginning of the IOB and are used as displacements from the beginning of the IOB when you wish to refer to one of the fields. The labels generated by this macro instruction are given with the fields of the IOB in *Appendix C: Disk Input/Output Block*. To avoid duplicate labels, you should use this macro instruction only once in a program.

The format of the $IOED macro instruction is:

	$IOED	

Read from Disk ($RDD)

This macro instruction generates an interface to the disk input/output supervisor that is to read from the disk device. When using this macro instruction, you must:

- Provide an IOB and use the $IOED macro instruction to establish the offsets in the IOB.
- Wait for the completion of the input operation (using $WAIT).
- Check for end of extent when the record is received — completion code of X'70' ($CPEOX) at $DFCMP in the DTF.

If both reading and writing for a program are to be performed using a single IOB, the bits of the Q-byte are altered to indicate an invalid operation. In this case, you must set off the bits of the Q-byte for all but the first read (or write) operation in the program.

If you will need to use the data in register 1 at a later time, you should save the contents of that register before issuing the $RDD macro instruction.

The format of the $RDD macro instruction is:

[Name]	$RDD	IOB-address,CS-address,NSECT-number [,DISK-5444/5445/3340]

IOB-address provides the address of the leftmost byte of the IOB which you created through your $IOBD macro instruction. The label provided must be the same as the name specified on your $IOBD macro instruction.

CS-address is the address of the rightmost byte of the main storage area containing the disk cylinder/sector address of the area you want to read. The cylinder/sector address for use with the 5444 is a two-byte, hexadecimal number. The first byte specifies the cylinder; the second specifies the sector. For use with the 5445, or 3340, a three-byte hexadecimal disk address is provided through this entry. The first byte specifies the cylinder; the second, the head number; the third, the sector.

NSECT-number indicates the hexadecimal number of sectors, minus one, to be read in this operation.

DISK-5444/5445/3340 specifies whether the operation is on a 5444, a 5445, or a 3340 disk drive. If this operand is omitted, 5444 is assumed.

Write to Disk ($WRTD)

This macro instruction generates an interface to the disk input/output supervisor needed to write records to disk. When you use this macro instruction, you must:

- Provide an IOB, and use the $IOED macro instruction to establish the offsets in the IOB.
- Wait for the completion of the output operation (using $WAIT).

If both reading and writing for a program are to be performed using a single IOB, the bits of the Q-byte are altered to indicate an invalid operation. In this case, you must set off the bits of the Q-byte for all but the first read (or write) operation in the program.

If you will need to use the data in register 1 at a later time, you should save the contents of that register before issuing the macro instruction.

The format of the $WRTD macro instruction is:

[Name]	$WRTD	IOB-address,CS-address,NSECT-number [,DISK-5444/5445/3340]

IOB-address provides the address of the disk IOB for this operation. The address is the name specified on the related $IOBD macro instruction.

CS-address is the address of the rightmost byte of the main storage area containing the disk cylinder/sector address of the area to which you want to write. The cylinder/sector address for use with the 5444 is a two-byte hexadecimal number. The first byte specifies the cylinder; the second specifies the sector. For use with the 5445 or the 3340, a three-byte hexadecimal disk address is provided through this entry. The first byte specifies the cylinder; the second, the head number; the third, the sector.

NSECT-number specifies the hexadecimal number of disk sectors, minus one, to be written.

DISK-5444/5445/3340 specifies whether the operation is on a 5444, a 5445, or a 3340 disk drive. If this operand is omitted, 5444 is assumed.

Wait for Disk IOS Completion ($WAIT)

This macro instruction is used with the $RDD and $WRTD macro instructions. It generates the code that allows you to wait for completion of the disk IOS operation. You provide the label of the associated IOB and an address to receive control in the event of an error.

If you will need to use the data in register 1 at a later time, you should save the contents of that register before issuing the $WAIT macro instruction.

The format of the $WAIT macro instruction is:

[Name]	$WAIT	[IOB-label] [,ERR-address]

IOB-label is the name assigned to the IOB in the $IOBD macro instruction. This same IOB must have previously been specified in either a $RDD or $WRTD macro instruction. If this operand is not entered, the address is assumed to be in register 1.

ERR-address specifies the address of the routine in your program that handles errors detected in the operation. If this operand is not entered, no error checking is performed, and you should check the return code in your program to determine the outcome of the operation.

3741 Device Support

This section describes the macro instructions that support the directly attached 3741 Data Station Model 1 or 2 or Programmable Work Station Model 3 or 4. The following functions are provided:

- Build a pre-open DTF and assign its offsets.
- Build the interface required to read input records from the 3741 device via a GET.
- Build the interface required to write output records to the 3741 device via a PUT.

The DTFs provide information to the data management routines that perform the input/output operations. The interfaces to these operations are provided through the 3741 macro instructions.

Define the File for 3741 ($DTFK)

The DTF provides information needed to allocate, open, and access a file on the 3741. This macro generates the DTF for this purpose.

The format of the $DTFK macro instruction is:

[Name]	$DTFK	NAME-filename,RECL-number,IO-address [,AC-I/O] [,RCAD-address] [,BUFNO-1/2] [,CHN-address] [,UP-mask]

NAME-filename specifies the name of the file. The name must be eight characters or less in length. This operand must be specified.

RECL-number specifies the decimal length (from 1 to 128) of the logical record. This operand must be specified.

IO-address provides the address of the leftmost byte of an area in main storage that contains the buffers and the IOB. This operand must be specified. The amount of storage must be the record length plus 26 times the BUFNO.

AC-I/O specifies whether the DTF is input or output. If this operand is not specified, input is assumed.

RCAD-address specifies the address of the leftmost byte of the logical record. If this operand is not entered, X'FFFF' is assumed.

BUFNO-1/2 allows you to specify either one or two buffers. If this operand is omitted, one buffer is assumed.

CHN-address indicates the address of the next DTF in the chain of DTFs. If there is no DTF chain, the operand is omitted and X'FFFF' is assumed.

UP-mask specifies the mask to test the eight external indicators. The code for the UP-mask must be specified as 8 binary bits. For example, to test bits 0, 3, 5, and 7, you would enter UP-10010101. The UP-mask is compared to the external indicators set on by the SWITCH statement for conditionally opening files. If the bits that are on in the UP-mask are also on in the external indicators set on by the SWITCH statement, the file will be opened. If the UP-mask is all zeroes or not used, the file will be unconditionally opened.

Note: Information on setting external indicators (SWITCH statement) can be found in the *IBM System/3 Model 15 System Control Programming Reference Manual* (for Program Number 5704-SC1), GC21-5077, and *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual* (for Program Number 5704-SC2), GC21-5162.

Construct 3741 Get Interface ($GETK)

The $GETK macro instruction generates the interface needed to communicate with the 3741 data management when a record is being read from the 3741. To use this macro instruction, construct a 3741 DTF for the file and use the $DTFO macro instruction to establish the offsets for the DTF. You must also provide an EXTRN statement with the label $$CPIP to use this macro. If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $GETK macro instruction.

The code generated by this macro instruction gives control to the data management routine. The data management routine completes execution and returns control to the generated code.

The format of the $GETK macro instruction is:

[Name]	$GETK	EOF-address [,DTF-address] [,ERR-address]

EOF-address specifies the address in your program that receives control when the end of file is detected. This operand must be specified.

DTF-address indicates the address of the leftmost byte of the DTF for this file. If this operand is not specified, the address is assumed to be in register 2.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code.

Construct 3741 Put Interface ($PUTK)

The $PUTK macro instruction generates the interface needed to put a record on the 3741. You must provide a DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF. You must also provide an EXTRN statement in your program with the label $$CPOP to use this macro. If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $PUTK macro instruction.

The code generated by this macro instruction gives control to the data management routine. The data management routine completes execution and returns control to the generated code.

The format of the $PUTK macro instruction is:

[Name]	$PUTK	[DTF-address] [,ERR-address]

DTF-address indicates the address of the leftmost byte of the DTF for this file. If this operand is not specified, the address is assumed to be in register 2.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code.

Tape Device Support

This section describes the macro instructions that support the IBM 3410/3411 Magnetic Tape Subsystem. The following functions are provided:

- Build a pre-open DTF for tape and assign its offsets.
- Build the interfaces required to read input records from a tape device via a get or a read.
- Build the interfaces required to write output records to a tape device via a put or a write.
- Build the interface required to issue tape control commands.
- Wait for completion of read, write, or tape control operations.

The tape DTFs provide information to the tape data management routines that perform the input/output operations. These operations are provided through the tape support macro instructions.

Define the File for Tape ($DTFT)

The DTF provides information needed to allocate and open a tape device. This macro instruction generates the code that builds the tape DTF. See *Appendix B: Define the File Control Blocks* for a description of the pre-open and post-open DTFs.

The format of the $DTFT macro instruction is:

[Name]	$DTFT	NAME-filename,IO-address,BLKL-number, RECL-number [,UP-mask] [,AC-IN/OUT] [,CHN-address] [,BASIC-Y/N] [,RCAD-address] [,MODE-LOCATE/MOVE] [,MBUFF-Y/N] [,RECFM-code] [,LIOA-number] [,SPAN-Y/N] [,CODE-A/E] [,OSET-B/number] [,END-code] [,MVF-Y/N]

NAME-filename is a required operand specifying the name of the tape file. The filename can be up to eight characters in length and must be the same as the name of the // FILE statement.

IO-address specifies the address of the leftmost byte of the main storage area used to contain all buffers and IOBs. This operand is required. The length of the area specified by this address is specified in the LIOA operand.

Note: If basic data management routines are used to process the file, this operand should point to a 40-byte area to contain the tape IOB.

BLKL-number is a required operand that specifies the decimal block length for the file. The minimum block length allowed is 18 bytes. If a shorter length is specified, 18 is assumed. For files with fixed-length records, the block length must be a multiple of the record length; for files with variable-length records, the block length must equal the length of the longest record plus eight.

Note: If basic tape data management is used, the block length in the DTF ($DFBKL) must be updated after the file is opened and before any read or write operation is performed. The field must also be updated before any subsequent read or write if the length used is different than the previous read or write.

RECL-number is a decimal value specifying the length of a logical record in the file. If variable-length records are used for the file, the record length specified must be equal to the longest record plus four. The minimum record length when variable-length records are used is four, which results in zero-length records. The minimum record length for files using fixed-length records is 18. This operand is required.

UP-mask specifies the mask to test the eight external indicators. The code for the UP-mask must be specified as 8 binary bits. For example, to test bits 0, 3, 5, and 7, you would enter UP-10010101. The UP-mask is compared to the external indicators set on by the SWITCH statement for conditionally opening files. If the bits that are on in the UP-mask are also on in the external indicators set on by the SWITCH statement, the file will be opened. If the UP-mask is all zeroes or not used, the file will be unconditionally opened.

Note: Information on setting external indicators (SWITCH statement) can be found in the *IBM System/3 Model 15 System Control Programming Reference Manual* for Program Number 5704-SC1, GC21-5077, and *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual* for Program Number 5704-SC2, GC21-5162.

AC-IN/OUT specifies the type of file. IN specifies an input file; OUT, an output file. If this operand is not entered, IN (input) is assumed.

Note: The operation code in the Tape DTF will be initialized according to the entries specified (or defaults) of the AC and BASIC operands, as follows:

	AC-IN	AC-OUT
BASIC-Y	Read forward	Write
BASIC-N	Get	Put

CHN-address specifies the address of the next DTF in the chain of DTFs. If there is no DTF chain or if this DTF is the last DTF in the chain, this operand should be omitted and X'FFFF' assumed.

BASIC-Y/N specifies whether this DTF uses the basic access method. If this operand is not entered, N (no) is assumed.

Notes:

1. BASIC-Y must be specified if any of the following macro instructions are used to process the file: $RDT, $WRTT, $CTLT or $WTT.
2. If you process ASCII files using the basic access method, you must translate the characters in your program.
3. Multivolume files are supported with the basic access method, but the EXTRN statement that is provided must be $$BTMM for multivolume support. $$BTAM and $$BTMM should not be used in the same program. $$BTMM supports both single and multivolume files.
4. Deferred open is not allowed with the basic access method.

RCAD-address specifies the symbolic record area address. This operand is required for all operations using the basic access method. If standard data management is used, this operand is required for an output operation and for an input operation using move mode.

MODE-LOCATE/MOVE indicates whether the locate mode or move mode is used. If this operand is not specified, MOVE is assumed. When locate mode is specified, the record address (RCAD-address) is set to the address of the record in the buffer. When move mode is used, records are moved from the buffer to the location specified by the record address.

Locate mode is valid only for input files. This operand should not be used if BASIC Y is specified.

RECFM-code specifies the record format used for the file. The codes and their meanings are:

Code	Record Format
F	Fixed, EBCDIC or ASCII
FB	Fixed blocked, EBCDIC or ASCII
V	Variable, EBCDIC
VB	Variable blocked, EBCDIC
D	Variable, ASCII
DB	Variable blocked, ASCII

If this operand is not specified, F is assumed.

MBUFF-Y/N indicates whether more than one buffer is used. If this operand is not specified, N (no) is assumed. The number of buffers is determined by the length of the I/O area, specified by the LIOA operand. This operand should not be used if BASIC-Y is specified.

LIOA-number is the total decimal length of the I/O area if more than two buffers are required (MBUFF-Y). If the number specified is zero or if this entry is omitted, two buffers are allocated in the I/O area. If this entry is not zero, as many buffers as possible are allocated in the I/O area.

The following formula can be used to determine the length of the I/O area:

Length of the I/O area = (40 + block length) times (number of buffers)

This operand should not be used if BASIC-Y is specified.

SPAN-Y/N specifies whether spanned records are used. If spanned records are used, BASIC-Y must also be specified. If this operand is omitted, N (no) is assumed. Specifying SPAN-Y causes the spanned record bit in the tape label to be set on. When you use SPAN-Y, you must span the records from block to block.

CODE-A/E specifies whether the file is an EBCDIC file or ASCII file. If the file is an EBCDIC file, specify CODE-E. If the file is an ASCII file or can be either ASCII or EBCDIC, specify CODE-A. If this operand is not entered, E is assumed.

OSET-B/number specifies the buffer offset of an ASCII block. B indicates that the first four bytes of the block contain the decimal block length and no buffer offset is present. B is valid only when RECFM-D or RECFM-DB is also specified. Only OSET-B or OSET-00 are valid for output files. OSET-number specifies, in decimal, the length of the buffer offset for the ASCII block. This buffer offset is skipped over when the record is supplied to your program. The maximum valid specification is OSET-99. If this operand is not specified, zero is assumed.

END-code specifies the tape control actions to be taken when the file is closed. The valid codes and their meanings are:

Code	Action
REWIND	Rewind the tape
UNLOAD	Rewind and unload the tape
LEAVE	No action taken

If this operand is not entered, REWIND is assumed.

MVF-Y/N specifies whether this file has multivolume tape output. If this operand is not specified, Y (yes) is assumed.

Construct a Tape Get Interface ($GETT)

The $GETT macro instruction generates the interface required to communicate with tape data management when a record is being read from a tape file. To use this instruction, you must construct a tape DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF.

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $GETT macro instruction.

You must also provide the labels for the necessary data management routines through EXTRN statements in your program. The names and functions of the data management routines are shown in Figure 13.

Module Name	Type of File Being Processed
$$CSIT	EBCDIC fixed input
$$CSOT	EBCDIC fixed output
$$CSIA	EBCDIC or ASCII fixed input
$$CSOA	EBCDIC or ASCII fixed output
$$CSTI	EBCDIC fixed or variable input
$$CSTO	EBCDIC fixed or variable output
$$CSAI	EBCDIC or ASCII fixed or variable input
$$CSAO	EBCDIC or ASCII fixed or variable output

Figure 13. Tape Data Management Modules

The code generated by this macro instruction gives control to the data management routine; the routine completes execution and returns control to the generated code. If the ERR or EOF operand is specified, the generated code tests the completion code returned by data management and branches to your routine. If reading variable length records, tape data management returns the length of the record at label $DFCRL in the DTF.

The format of the $GETT macro instruction is:

[Name]	$GETT	MODULE-name[,DTF-address] [,RCAD-address] [,OPC-Y/N] [,ERR-address] [,EOF-address]

MODULE-name is a required operand that specifies the module name of the tape data management subroutine. Following are the module names used and the types of files they will process:

$$CSIT	EBCDIC fixed input
$$CSIA	EBCDIC or ASCII fixed input
$$CSTI	EBCDIC (fixed or variable) input
$$CSAI	EBCDIC or ASCII (fixed or variable) input

An EXTRN must be provided for the module name that is used.

DTF-address indicates the address of the leftmost byte of the DTF for this file. If this operand is not specified, the address is assumed to be in register 2.

RCAD-address specifies the address of the leftmost byte of the logical record area. If this operand is not entered, the address of the record area is assumed to be in the DTF at $DFLRA.

OPC-Y/N specifies whether to generate the code to set the operation code. If this operand is not entered, N (no) is assumed, and the operation code is not modified.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code.

EOF-address specifies the address in your program that receives control when the end-of-file is detected. If this operand is not supplied, no code is generated to check for the end-of-file condition.

Note: If ERR or EOF addresses are not specified, you should check the return code in your program to determine the outcome of the operation.

Construct a Tape Put Interface ($PUTT)

This macro instruction generates the interface needed to communicate with tape data management when writing a record to tape. You must provide a DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF. You must also provide, through EXTRN statements in your program, the labels of the tape data management modules necessary to perform the output operation (see Figure 13).

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $PUTT macro instruction.

The code generated by this macro instruction gives control to the data management routine. The routine completes execution and returns control to the generated code. If the ERR operand is specified, the generated code checks the completion code for errors and branches to your error routine if errors occurred.

The format of the $PUTT macro instruction is:

[Name]	$PUTT	MODULE-name [,DTF-address] [,RCAD-address] [,OPC-Y/N] [,LENAD-address] [,ERR-address]

MODULE-name, is a required operand, and specifies the name of the tape data management subroutine to be used. Following are the module names used and the type of files they will process:

$$CSOT	EBCDIC fixed output
$$CSOA	EBCDIC or ASCII fixed output
$$CSTO	EBCDIC fixed or variable output
$$CSAO	EBCDIC or ASCII fixed or variable output

DTF-address specifies the address of the leftmost byte of the DTF for the file. If this operand is not specified, the address is assumed to be in register 2.

LENAD-address specifies the address of the rightmost byte of a two-byte area which contains the length of the current record. This operand is used only for variable files. If this operand is not specified, the length of the record is assumed to be in the DTF at $DFCRL.

RCAD-address specifies the address of the leftmost byte of the record to be put. If this operand is not entered, the record address is assumed to be in the DTF at label $DFLRA.

OPC-Y/N specifies whether to generate the code to set the operation code. If this operand is not entered, N (no) is assumed, and the operation code is not modified in the DTF.

ERR-address specifies the address in your program where control should be passed if a permanent I/O error occurs. If this operand is not entered, no permanent I/O error checking code is generated and you should check the return code in your program to determine the outcome of the operation.

Read from Tape ($RDT)

This macro instruction generates an interface to basic tape data management to read from a tape device. When using this macro instruction, you must:

- Provide a tape DTF and use $DTFO to establish the offsets in the DTF.
- Wait for completion of the input operation and check for end-of-file by using the $WTT macro instruction.
- Provide EXTRN statements in your program for the basic tape data management module ($$BTAM or $$BTMM) and for the entry point to the read routine in that module (DMBTRW).

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $RDT macro instruction. The generated code for this macro instruction uses register 2.

The code generated by this macro instruction branches to basic tape data management to begin the read operation.

The format of the $RDT macro instruction is:

[Name]	$RDT	[DTF-address] [,RCAD-address] [,DIRECT-FORW/BACK]

DTF-address specifies the address of the leftmost byte of the DTF for the file. If this operand is not entered, the address is assumed to be in register 2.

RCAD-address specifies the address of the leftmost byte of the record area. If this operand is not specified, the address is assumed to be in the DTF at $DFLRA.

DIRECT-FORW or BACK specifies the direction of the read and causes the operation code in the DTF to be set (see AC operand of *$DTFT*). If this operand is not entered, the operation code is unchanged.

Write to Tape ($WRTT)

This macro instruction generates the interface to basic tape data management needed to write records to tape. When you use this macro instruction, you must:

- Provide a DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF.
- Wait for the completion of the I/O operation by using the $WTT macro instruction.
- Provide EXTRN statements in your program for the basic tape data management module ($$BTAM or $$BTMM) and for the entry point to the write routine in that module (DMBTRW).

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $WRTT macro instruction, because the generated code for that macro instruction uses register 2.

The code generated by this macro instruction branches to basic tape data management to start the operation.

The format of the $WRTT macro instruction is:

[Name]	$WRTT	[DTF-address] [,RCAD-address] [,OPC-Y/N]

DTF-address is the address of the leftmost byte of the DTF for the file. If this operand is not specified, the address of the DTF is assumed to be in register 2.

RCAD-address specifies the address of the leftmost byte of the record area. If this operand is not specified, the address is assumed to be in the DTF at $DFLRA.

OPC-Y/N specifies whether the write operation code in the DTF is to be set. If N (no) is specified or if this operand is not entered, the operation code is not modified in the DTF.

Control Command for Tape ($CTLT)

This macro instruction generates the interface to basic tape data management to issue control commands to the tape device. It is not used to get records from or put records out on a tape file. To use this macro instruction, you must:

- Provide a DTF for the file on the tape device and use the $DTFO macro instruction to establish the offsets in the DTF.
- Wait for completion of the operation by issuing the $WTT macro instruction.
- Provide EXTRN statements in your program for the basic tape data management module ($$BTAM or $$BTMM) and for the entry point to the control routine in that module (DMBTPS).

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $CTLT macro instruction.

The code generated by this macro instruction branches to the basic tape data management to initiate the operation.

The format of the $CTLT macro instruction is:

[Name]	$CTLT	[DTF-address] [,OPC-code]

DTF-address specifies the address of the leftmost byte of the DTF for the file on the tape device. If this operand is not specified, the address of the DTF is assumed to be in register 2.

OPC-code specifies the control operation to be performed. If this operand is not specified, no code is generated to modify the operation code in the DTF. The valid codes and their meanings are:

Code	Operation
FSF	Forward space file
FSB	Forward space block
BSF	Backspace file
BSB	Backspace block
REW	Rewind tape
RUN	Rewind and unload tape
WTM	Write tape mark

Wait For Tape I/O Completion ($WTT)

This macro instruction is used with the $RDT, $WRTT, and $CTLT macro instructions. It generates the linkage to basic tape data management to wait for the completion of operations that have been initiated. You must provide the address of the tape DTF for the file and use the $DTFO macro instruction to establish the offsets for that DTF. You must also provide EXTRN statements in your program for the basic tape data management module ($$BTAM or $$BTMM) and for the entry point to the wait routine in that module (DMBTWT). You may also provide addresses where control is to be returned in the event of a permanent I/O error, end-of-file condition, or end-of-tape condition.

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $WTT macro instruction.

The generated code from this macro instruction checks the completion code in the DTF to determine the outcome of the operation. When an abnormal completion is detected, control is passed to the appropriate address in your program, (if you have specified ERR, EOJ, or EOT) or to the next instruction in your program.

The format of the $WTT macro instruction is:

[Name]	$WTT	[DTF-address] [,ERR-address] [,EOF-address] [,EOT-address] [,WLRS-address] [,WLRL-address]

DTF-address specifies the address of the leftmost byte in the DTF for the file. If this operand is omitted, the address of the DTF is assumed to be in register 2.

ERR-address is the address of the routine in your program that receives control when a controlled cancel is indicated in the completion code. If this operand is not entered, the controlled cancel is ignored and control returns to the next instruction in your program.

EOF-address specifies the address of your routine that receives control when end-of-file occurs. If this operand is omitted, the end-of-file condition is ignored and control returns to the next instruction in your program.

EOT-address is the address of the routine in your program that receives control when end-of-tape is detected. If this operand is not specified, the condition is ignored and control returns to the next instruction in your program.

WLRS-address specifies the address of the routine that is to get control when a record that is too short is read. This operand should be used when waiting for a completion of a read operation.

WLRL-address specifies the address of the routine that should get control when a record that is too long is read. This operand should be used when waiting for the completion of a read operation.

Note: If ERR, EOF, EOT, WLRS, or WLRL addresses are not specified, you should check the return code in your program to determine the outcome of the operation.

Device Independent Support

This section describes the macro instructions that support device independent files. Device independent data management supports sequential files on disk, tape, and card devices, and on printers. The device type is determined at execution time according to data on the file card. The following functions are provided:

- Build a pre-open DTF for device independent data management.
- Build the interface required to get a fixed length record from a file.
- Build the interface required to put a fixed length record to a file.

The device independent DTF provides information to the device independent data management routines that perform the input/output operation.

Define the File for Device Independent ($DTFI)

The $DTFI macro instruction provides information needed to allocate, open and access a device-independent file. This macro instruction generates the code that builds a device-independent DTF. See *Appendix B: Define the File Control Blocks* for a description of the device-independent DTFs. To use this macro instruction, you must use the $DTFO macro instruction to establish the offsets for the DTF.

The format of the $DTFI macro instruction is:

[Name]	$DTFI	NAME-filename, RECL-number, IO-address [,AC-IN/OUT] [,BLKL-number] [,RCAD-address] [,BUFNO-1/2] [,CHN-address] [,UP-mask]

NAME-filename specifies the name of the file. The name can be eight characters or less in length. This operand must be specified.

RECL-number is a decimal value specifying the length of a logical record in the file. This operand is required.

Note: The record length will be changed by the device-independent open to accommodate the physical device size. If the record length is greater than the physical device length, the record length is changed to the device length. For output files, this means that the number of positions by which the size of the logical record area exceeds the device size will be truncated from the rightmost positions of the logical record. For input files, this means that the contents of the positions by which the size of the logical record area exceeds the device size will not be changed. If the record length is less than the physical device length, the record length is not changed and, for output files, the physical record will be padded with blanks.

IO-address specifies the address of the I/O area. This area must be on a 256-byte boundary for each buffer. This I/O area must be at least equal in length to block length plus 40 bytes, or be 286 bytes, whichever is greater. This operand is required.

AC-IN/OUT specifies the type of file. IN specifies an input file; OUT, an output file.

If this operand is omitted, IN (input) is assumed.

BLKL-number specifies the decimal block length for the file. If this operand is not specified, the value of the record length (RECL) is assumed.

Note: For tape files, the actual block length is used. For disk files, the block length is rounded down to a multiple of 256. The block length is not used for unit record devices.

RCAD-address specifies the address of the leftmost byte of the logical record. If this operand is not entered, X'FFFF' is assumed and the address must be supplied when an operation is requested.

BUFNO-1/2 specifies the number of buffers to be used. If this operand is not specified, 1 is assumed.

CHN-address specifies the address of the next DTF in the chain of DTFs. If there is no DTF chain or if this DTF is the last DTF in the chain, this operand should be omitted and the end of chain (X'FFFF') assumed.

UP-mask specifies the mask to test the eight external indicators. The code for the UP-mask must be specified as 8 binary bits. For example, to test bits 0, 3, 5, and 7, you would enter UP-10010101. The UP-mask is compared to the external indicators set on by the SWITCH statement for conditionally opening files. If the bits that are on in the UP-mask are also on in the external indicators set on by the SWITCH statement, the file will be opened. If the UP-mask is all zeroes or not used, the file will be unconditionally opened.

Note: Information on setting external indicators (SWITCH statement) can be found in the *IBM System/3 Model 15 System Control Programming Reference Manual* (for Program Number 5704-SC1), GC21-5077, and in the *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual*

Construct a Device-Independent Get Interface ($GETI)

The $GETI macro instruction generates the interface needed to communicate with device-independent data management when a record is being read. To use this macro instruction, you must construct a device-independent DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF. In addition, you must provide the labels for the necessary data management routines through an EXTRN to $$CSIO in your programs.

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $GETI instruction.

The format of the $GETI macro instruction is:

[Name]	$GETI	[DTF-address] [,RCAD-address] [,ERR-address] [,EOF-address]

DTF-address indicates the address of the leftmost byte of the DTF for this file. If this operand is not specified, the address is assumed to be in register 2.

RCAD-address specifies the address of the leftmost byte of the record area. If this operand is not entered, the address is assumed to be in the DTF at $DFLRA.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code.

EOF-address specifies the address in your program that receives control when the end-of-file is detected. If this operand is not supplied, no code is generated to check for the end-of-file condition.

Note: If ERR or EOF addresses are not specified, you should check the return code in your program to determine the outcome of the operation.

Construct a Device-Independent Put Interface ($PUTI)

The $PUTI macro instruction generates the interface needed to communicate with device-independent data management when writing a record. To use this macro instruction, you must construct a device-independent DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF. In addition, you must provide the labels for the necessary data management routines through an EXTRN to $$CSIO in your programs.

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $PUTI instruction.

The format of the $PUTI macro instruction is:

[Name]	$PUTI	[DTF-address] [,RCAD-address] [,ERR-address] [,EOX-address]

DTF-address indicates the address of the leftmost byte of the DTF for this file. If this operand is not specified, the address is assumed to be in register 2.

RCAD-address specifies the address of the leftmost byte of the record area. If this operand is not entered, the address is assumed to be in the DTF at $DFLRA.

ERR-address supplies the address in your program where control is passed if the controlled cancel option is taken in response to a permanent I/O error. If this operand is omitted, no code is generated to check for the controlled cancel completion code and you should check the return code in your program to determine the outcome of the operation.

EOX-address specifies the address of the routine that receives control when the end of extent on disk is reached. The file will not contain the record for which the put was requested.

If this operand is not specified, the code that checks for the end of extent completion code is not generated.

CRT/Keyboard

This section describes the macro instructions that support the CRT/Keyboard. This support can be grouped in two categories: display support and program function key support. It provides the following capabilities:

- Builds a pre-open DTF for the CRT/Keyboard data management.
- Builds the interface to get a record from the CRT/Keyboard.
- Builds the interface to put a record to the CRT.
- Builds the interface to first put a record to the CRT, and then to get a record from the Keyboard.
- Builds the interface to use the program function keys.

The CRT/Keyboard provides information to the CRT/Keyboard data management routines that perform the input/output operations.

Display Support

The macros which follow support the display functions of the CRT/Keyboard.

Define the File for CRT ($DTFS)

The $DTFS macro instruction provides information needed to allocate, open, and access a CRT file. This macro instruction generates the code that builds a CRT DTF. When using this macro, $DTFO *CRT-Y* or *ALL-Y* must also be used. See *Appendix B: Define the File Control Blocks* for a description of the CRT DTFs.

The format of the $DTFS macro instruction is:

[Name]	$DTFS	[PUTDAT-address] [,PUTLOC-number] [,UP-mask] [,CHN-address] [,PUTLEN-number] [,OPC-code] [,GETDAT-address] [,GETLOC-number] [,GETLEN-number] [,BLANK-Y/N]

PUTDAT-address specifies the leftmost byte of the logical record for a put request. For a $PGS request, this area is used for the output. If this operand is not specified, X'0000' is assumed, and the address must be updated with (or prior to) the first $PGS or $PUTS request issued.

PUTLOC-number specifies a number which represents the starting location on the CRT for a put request. Valid entries for this operand are from 0 through 278. If this operand is not specified, 0 (the first CRT position) is assumed. If the number exceeds 278, no data is written.

UP-mask specifies the mask to test the eight external indicators. The code for the UP-mask must be specified as 8 binary bits. For example, to test bits 0, 3, 5, and 7, you would enter UP-10010101. The UP-mask is compared to the external indicators set on by the SWITCH statement for conditionally opening files. If the bits that are on in the UP-mask are also on in the external indicators set on by the SWITCH statement, the file will be opened. If the UP-mask is all zeroes or not used, the file will be unconditionally opened.

Note: Information on setting external indicators (SWITCH statement) can be found in the *IBM System/3 Model 15 System Control Programming Reference Manual* (for Program Number 5704-SC1), GC21-5077, and in the *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual* (for Program Number 5704-SC2) GC21-5162.

CHN-name specifies the address of the next DTF in the forward DTF chain. If there is no DTF chain or if this DTF is the last one in the chain, this operand should be omitted and end of chain (X'FFFF') assumed.

PUTLEN-number specifies the number of bytes to process for a put request. If this operand is not specified, the missing information must be supplied with (or prior to) the first $PGS or $PUTS request.

Valid entries for this operand are from 1 through 279. If this number plus the entry for the PUTLOC operand exceeds 279, the data written is truncated at location 278.

OPC-code specifies the operation code to be set. If this operand is not specified, the information must be supplied with (or prior to issuing) the first $GETS, $PUTS, or $PGS request. The codes and their meanings are:

Code	Meaning
IN	Set operation code for input only ($GETS).
INR	Set operation code for input on request ($GETS).
OUT	Set operation code for output only ($PUTS).
OUTIN	Set operation code for output/input (for $PGS).

GETDAT-address specifies the leftmost byte of the area into which the input data will be placed for a get request; for a $PGS request, this area is used for the input. If this operand is not specified, X'0000' is assumed, and the information must be supplied with (or prior to issuing) the first $GETS or $PGS request.

GETLOC-number specifies a number which indicates the starting location on the CRT for a get request. Valid entries for this operand are 0 through 278. If this operand is not specified, 0 (the first CRT position) is assumed. If a number greater than 278 is specified, no data will be read.

GETLEN-number is a decimal number which represents the number of bytes to get. If this operand is not specified, X'0000' is assumed, and the missing information must be supplied with (or prior to issuing) the first $PGS or $PUTS request. Valid entries for this operand are 1 through 279. If this number plus the entry specified for the GETLOC operand exceeds 279, the data read is truncated after location 278.

BLANK-Y/N determines whether to leave the previous data in the CRT buffer or to blank the buffer. If Y (yes) is specified, the operation code is set to blank the buffer. If this operand is not specified, N (no) is assumed. When used in conjunction with the $PUTS and $PGS macros, the operand causes all 279 bytes to be blanked; when used with $GETS, only the input area is blanked.

Get a Record from the CRT/Keyboard ($GETS)

The $GETS macro instruction generates the interface needed to communicate with CRT data management when a record is being read from the CRT. To use this macro instruction, construct a CRT DTF for the file and use the $DTFO macro to establish the offsets for the DTF. You must include an EXTRN for $$CODM. If you will need to use the data in register 2 at a later time, you should save the contents of that register before using the $GETS macro instruction.

The format for the $GETS macro instruction is:

[Name]	$GETS	[DTF-address] [,GETDAT-address] [,GETLEN-number] [,GETLOC-number] [,BLANK-Y/N] [,OPC-IN/INR/N] [,EOF-address]

DTF-address specifies the leftmost byte of the DTF for this file. If this operand is not specified, the address of the DTF is assumed to be in register 2.

GETDAT-address specifies the leftmost byte of the area into which the data will be placed.

GETLEN-number specifies the number of bytes to get. Valid entries for this operand are 1 through 279. If the sum of this number plus the number specified for the GETLOC operand exceeds 279, the data read is truncated after location 279.

GETLOC-number specifies a number representing the starting location on the CRT for this get. Valid entries for this operand are 0 (the first CRT position) through 278. If this entry exceeds 278, no data is read. If this operand is omitted, the corresponding entry in the DTF is not modified.

BLANK-Y/N determines whether to leave the previous data in the CRT buffer or to blank the buffer. If Y (yes) is specified, the operation code is set to blank the buffer. If this operand is not specified, N (no) is assumed.

OPC-IN/INR/N determines whether to set the operation code to input only (IN), to input on request (INR), or to leave the operation code unchanged. If INR is entered, the operator is prompted WAITING FOR REQUEST. Then, when the PA1 key is pressed, ENTER DATA is prompted. If N (no) is specified or if this operand is omitted, the operation code is not changed.

EOF-address specifies the address in your program that should receive control when end of file is recognized (end of file is indicated by /* as the first two characters of input) If this operand is not specified, no code is generated to check for the end-of-file condition, and you should check the return code in your program to determine the outcome of the operation.

Generate a PUT/GET Operation Through CRT Data Management ($PGS)

This macro instruction generates a PUT/GET data request to CRT data management. To use this instruction, you must construct a CRT DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF.

You must also provide the labels for the necessary data management routines through an EXTRN for $$CODM.

If you will need to use the data in register 2 at a later time you should save the contents of that register before issuing the $PGS macro instruction.

The format for the $PGS macro instruction is:

[Name]	$PGS	[DTF-address] [,BLANK-Y/N] [,OPC-Y/N] [,PUTDAT-address] [,PUTLEN-number] [,PUTLOC-number] [,GETDAT-address] [,GETLEN-number] [,GETLOC-number] [,EOF-address]

DTF-address specifies the address of the DTF for this file. If the operand is not specified, the address is assumed to be in register 2.

BLANK-Y/N determines whether to blank the data in the 279 byte input area before a CRT operation. If Y (yes) is specified, the operation code is set to blank the input area; if this operand is omitted, N (no) is assumed and the area is not blanked.

OPC-Y/N specifies whether to set the operation code to output/input. If N is specified or if this operand is not specified, N (no) is assumed, and the operation code is not modified in the DTF.

PUTDAT-address identifies the leftmost byte of the user area from which the data will be taken.

PUTLEN-number specifies the number of bytes to put to the CRT. Valid entries for this operand are 1 through 279. If the sum of this number plus the entry specified for the PUTLOC operand exceeds 279, the data written is truncated after location 278.

PUTLOC-number specifies the starting location on the screen for this put request. Valid entries for this operand are 0 (the first CRT position) through 278. If this number exceeds 278, no data is written.

GETDAT-number specifies the leftmost byte of the area into which the data will be placed.

GETLEN-number specifies the number of bytes to get from the CRT. Valid entries for this operand are 1 through 279. If the sum of this number plus the entry specified for the GETLOC operand exceeds 279, the data read is truncated after location 278.

GETLOC-number specifies the starting location on the CRT for this get request. Valid entries for this operand are 0 (the first CRT position) through 278. If this number exceeds 278, no data is read.

EOF-address specifies the address in your program that should receive control when end of file is recognized (indicated by the characters /* in the first two input positions). If this operand is not specified, no code is generated to check for the end-of-file condition, and you should check the return code in your program for the outcome of the operation.

Note: If the following operands – PUTDAT, PUTLOC, PUTLEN, GETDAT, GETLOC, GETLEN, BLANK, or OPC – are not specified, you must supply the missing information in the DTF before issuing the first $PGS request.

Put a Record to the CRT via Data Management ($PUTS)

This macro instruction generates a put data request to CRT data management. To use this macro instruction, you must construct a CRT DTF for the file and use the $DTFO macro instruction to establish the offsets in the DTF.

If you will need to use the data in register 2 at a later time, you should save the contents of that register before issuing the $PUTS macro instruction. You must also provide the labels for the necessary data management routines through an EXTRN to $$CODM.

The format for the $PUTS macro instruction is:

[Name]	$PUTS	[DTF-address] [,BLANK-Y/N] [,OPC-Y/N] [,PUTDAT-address] [,PUTLOC-number] [,PUTLEN-number]

DTF-address specifies the address of the DTF for this file. If this operand is not specified, the address is assumed to be in register 2.

BLANK-Y/N specifies whether to blank the data in the 279-byte input area before a CRT operation. If Y (yes) is specified, the operation code is set to blank the input area. If this operand is omitted or if N (no) is specified, the input area is not blanked.

OPC-Y/N specifies whether the operation code is set to output only. If this operand is omitted, or if N (no) is specified, the operation code is not modified.

PUTDAT-address specifies the leftmost byte of the area from which the data will be taken.

PUTLOC-number specifies the starting location on the CRT for this put. Valid entries for this operand are 0 (the first CRT position) through 278. If this number exceeds 278, no data is written.

PUTLEN-number specifies how many bytes to put to the CRT. Valid entries for this operand are 1 through 279. If the sum of this number plus the entry specified for the PUTLOC operand exceeds 279, the data written is truncated at location 278.

Note: If the following operands – PUTDAT, PUTLOC, PUTLEN, BLANK, or OPC – are missing, the missing information must be supplied in the DTF before the first put request is issued.

Program Function Key Support

Program Function (PF) keys 1-9 on the CRT/Keyboard are available for use in your program. When an assigned PF key is pressed, the program requesting its use is notified. The program may then test to see which key was pressed in order to condition subsequent operations.

Generate a Parameter List for a Program Function Key Request ($CQEP)

This macro instruction generates a ten-byte parameter list which requests a program function key. The format of the generated list is as follows:

Byte	Field Description
0-6	Reserved
7	CQE Q-code
8	CQE request code
9	PF key requested

The format of the $CQEP macro instruction is:

[Name]	$CQEP	[KEY-number]

KEY-number specifies the number of the program function key requested (keys 1-9 are available for assignment to your program). If this operand is not specified, PF9 is assigned.

Allocate Program Function Key to a Program ($PFKY)

The $PFKY macro should be used initially to allocate a function key to a program for subsequent testing via the $PFKT macro. If the function key has already been allocated, the ERR branch will be affected.

The format of the $PFKY macro instruction is:

[Name]	$PFKY	[CQE-address] [,ERR-address]

CQE-address specifies the address of the parameter list that specifies which program function key to assign, and loads this address into register 1. If this operand is not specified, the address is assumed to be in register 1 (see *$CQEP*). If you will need the contents of register 1 at a later time, you should save the contents of that register before issuing the $PFKY macro instruction.

ERR-address specifies the address in your program that should receive control if the completion code in the parameter list indicates that the requested key is not available. If this operand is not specified, no code is generated to check for a successful key assignment, and you should check the completion code in your parameter list to determine the outcome of the operation (a value of X'40' at displacement 2 in the parameter list indicates that a key was successfully assigned).

Test for Program Function Key Pressed ($PFKT)

Your program can either wait for a program function key to be pressed, or it can test at any time to see if a specified key was pressed. Only one program function key can be pressed before your program is notified.

This macro instruction generates code to test whether a program function key was pressed. The chart under *Using More Than One Program Function Key* shows the value to check for to determine if a specific key was pressed.

The format of the $PFKT macro instruction is:

[Name]	$PFKT	[CQE-address] [,WAIT-Y/N] [,LABEL-address]

CQE-address specifies the address of the parameter list that was used to request the assignment of the program function key being tested, and loads this address into register 1 (the address is assumed to be in register 1 if this operand is not specified). If you will need the contents of register 1 at a later time, you should save the contents of that register before issuing the $PFKT macro instruction.

WAIT-Y/N specifies whether to wait for the assigned program function key to be pressed. If Y (yes) is specified, your program waits until an assigned key is pressed. If no keys have been assigned, this is a permanent wait. If N (no) is specified, or if this operand is omitted, the LABEL operand must be specified.

LABEL-address specifies the address in your program that should receive control when an assigned program function key is pressed. This operand is used only if WAIT-N is specified or if the WAIT operand is omitted.

Using More Than One Program Function Key

If you want to use more than one program function key in your program, you must request each key separately, using the $PFKY macro instruction. You must use a different parameter list for each different key you wish to assign. To test if a program function key was pressed, you should issue the $PFKT macro instruction. Then, when control is returned to your program, you should determine **which** key was pressed. The following chart shows which fields in the parameter list should be modified and investigated:

PF Key	Value to set a displacement 08 before issuing $PFKY macro instruction	Value to test for at displacement 02 after issuing $PFKY macro instruction
1	X'00'	X'31'
2	X'05'	X'32'
3	X'0A'	X'33'
4	X'0F'	X'34'
5	X'14'	X'35'
6	X'19'	X'36'
7	X'1E'	X'37'
8	X'23'	X'38'
9	X'28'	X'39'

OCL FOR MACRO PROCESSOR

OCL statements used to call the macro processor can be entered through the system input device or be called as a procedure from the source library. The OCL statements necessary to call the macro processor are shown in Figure 14. The // COMPILE statement shown is only necessary when input is in a source library.

SAMPLE PROGRAM

This sample program uses the macro processor and the IBM System/3 Model 15 Basic Assembler Program: 5704-AS1 or 5704-AS2. The coding shown in Figure 15 produces an object program. To use the program, you must link edit the object program and execute it. The macro processor can be used with any valid assembler on the IBM System/3 Model 15 and is not limited to use with the program product 5704-AS1 or 5704-AS2.

Purpose of the Sample Program

The sample program in Figure 15 is used to print input records entered from the system input device. It reads data records from the system input device and prints them on a line printer. Each printed line reproduces one input record. If the spooling function is active for this program, the input may have been read prior to running this program, and the output may be retrieved at any time convenient for you.

Termination of the Sample Program

The sample program terminates in one of two ways:

1. After successful completion of the program, EJ is displayed on the CRT.

2. When an error occurs during processing, one of the following halts is displayed on the CRT.

   A1—if an error is returned from the system input routine.

   A2—if an error is returned from the printer routine.

   You respond to these halts by choosing option 2. EJ is then displayed on the CRT.

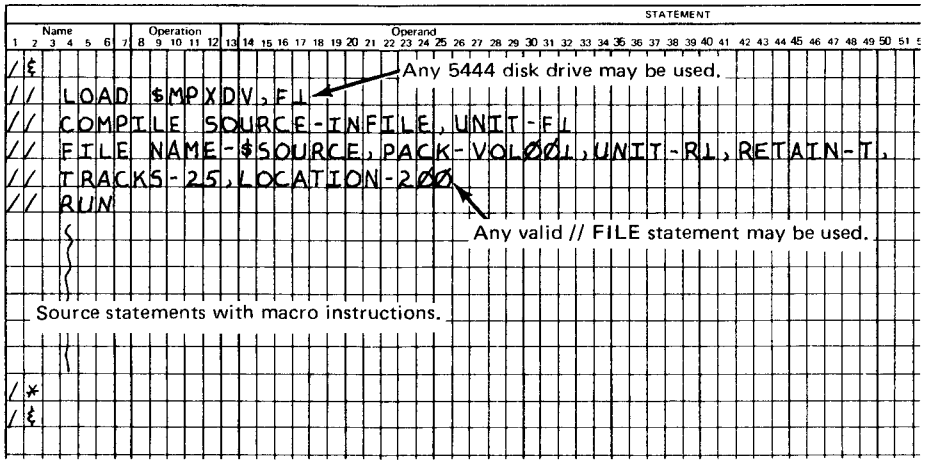


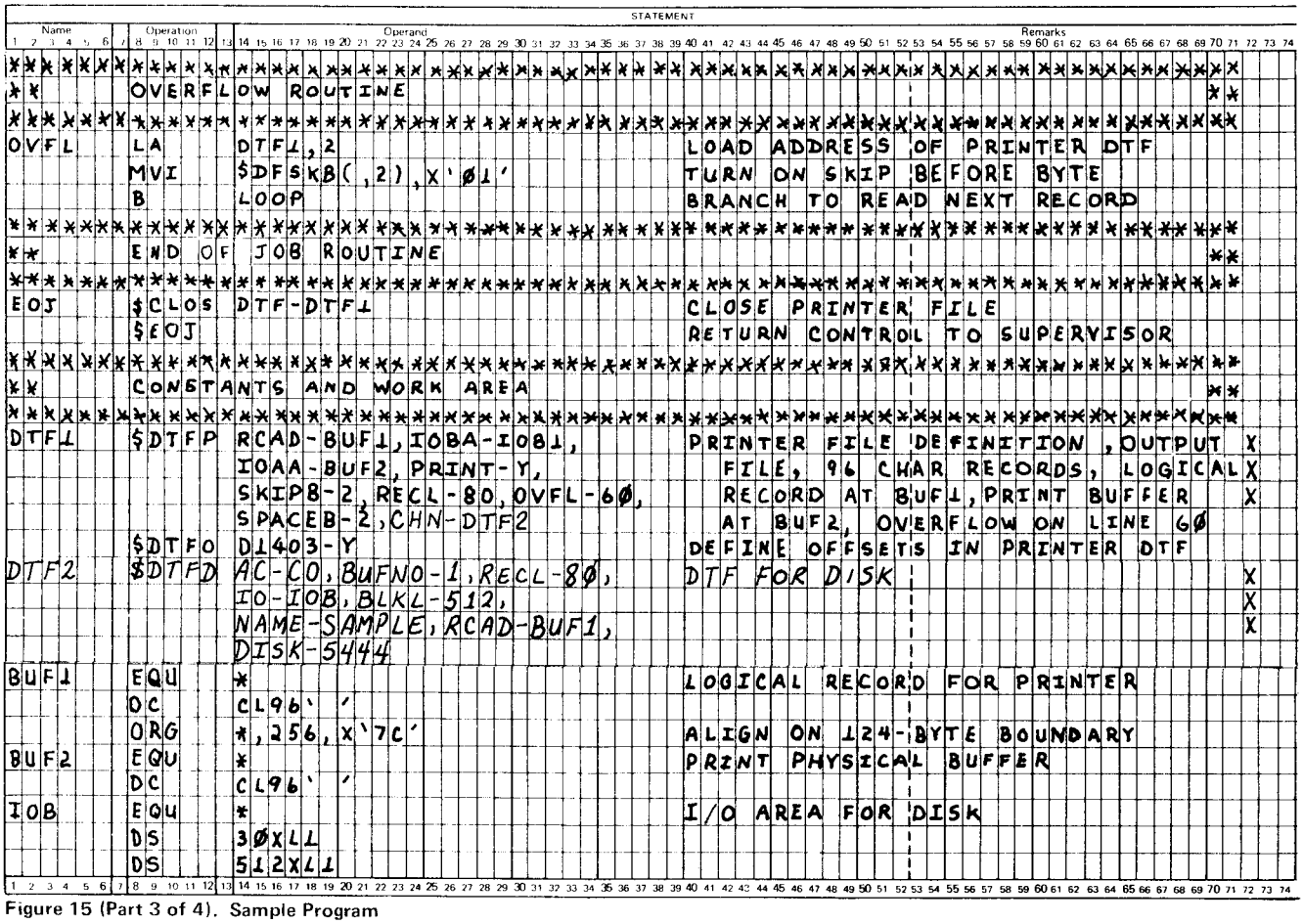
Figure 14. OCL Statements for Using the Macro Processor

```
// LOAD $MPXDV,F1                      LOAD MACRO PROCESSOR FROM SYSTEM PACK
// FILE NAME-$SOURCE,PACK-VOL001,UNIT-R1,
// RETAIN-T,TRACKS-30                  FILE STATEMENT FOR $SOURCE
// RUN
SAMPLE TITLE 'SAMPLE PROGRAM'          THIS STATEMENT APPEARS ON EACH PAGE
SAMPLE START 0
***********************************************************************
**     ALLOCATE AND OPEN PRINTER FILE                                **
***********************************************************************
       $ALOC  DTF-DTF1                 ALLOCATE PRINTER TO THIS PROGRAM
       $OPEN  DTF-DTF1                 OPEN THE PRINTER FILE
***********************************************************************
**     SUPPLY EXTERN FOR PRINTER DATA MANAGEMENT MODULE              **
***********************************************************************
       EXTRN  $$LPRT                   PRINTER DATA MANAGEMENT MODULE
       EXTRN  $$CSOP
***********************************************************************
**     PROCESSING ROUTINE                                            **
***********************************************************************
```

Figure 15 (Part 1 of 4). Sample Program

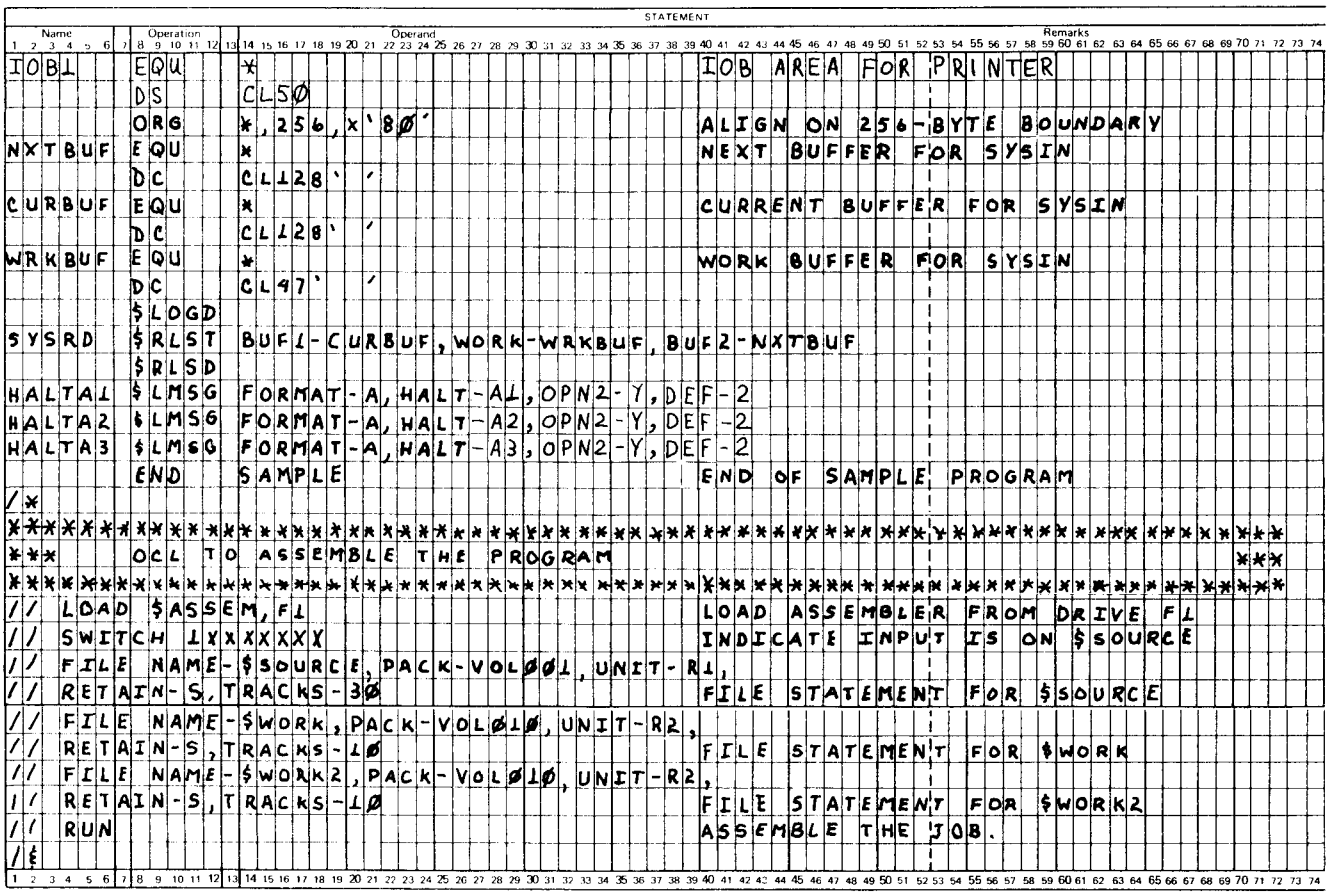
```
LOOP   LA     SYSRD,2                  LOAD ADDR OF SYSTEM READER PARM LIST
       $READ  EOF-EOJ,EOJ-EOJ,         READ FROM SYSTEM READER         X
              ERR-CANCEL
       L      $SRBF1(,2),1             LOAD CURRENT BUFFER ADDRESS
       MVC    BUF1+79,79(80,1)         MOVE CURRENT BUF TO OUTPUT BUF
       $PUTD  AC-CO,DTF-DTF2,ERR-ENDFIL PUT RECORD IN DISK FILE
       $PUTP  DTF-DTF1,ERR-CNCEL2,     PRINT INPUT RECORD TO LINE PRINTER
              OVFL-OVFL
       MVI    $DFSKB(,2),X'00'         TURN OFF SKIP BEFORE BYTE
       B      LOOP
***********************************************************************
       ERROR  HANDLING ROUTINES                                       *
***********************************************************************
CANCEL $LOG   LIST-HALTA1,OPN2-EOJ     A1 HALT ON ERROR FROM SYSIN
CNCEL2 $LOG   LIST-HALTA2,OPN2-EOJ     A2 HALT ON ERROR FROM PRINTER
ENDFIL $LOG   LIST-HALTA3,OPN2-EOJ     A3 HALT ON ERROR FROM DISK
```

Figure 15 (Part 2 of 4). Sample Program



```
STATEMENT
Name | Operation | Operand | Remarks
*********************************************************************
**  OVERFLOW ROUTINE                                               **
*********************************************************************
OVFL  LA    DTFL,2                  LOAD ADDRESS OF PRINTER DTF
      MVI   $DFSKB(,2),X'01'        TURN ON SKIP BEFORE BYTE
      B     LOOP                    BRANCH TO READ NEXT RECORD
*********************************************************************
**  END OF JOB ROUTINE                                             **
*********************************************************************
EOJ   $CLOS DTF-DTFL                CLOSE PRINTER FILE
      $EOJ                          RETURN CONTROL TO SUPERVISOR
*********************************************************************
**  CONSTANTS AND WORK AREA                                        **
*********************************************************************
DTFL  $DTFP RCAD-BUF1,IOBA-IOB1,    PRINTER FILE DEFINITION, OUTPUT X
            IOAA-BUF2,PRINT-Y,        FILE, 96 CHAR RECORDS, LOGICAL X
            SKIP8-2,RECL-80,OVFL-60,  RECORD AT BUF1,PRINT BUFFER  X
            SPACEB-2,CHN-DTF2         AT BUF2, OVERFLOW ON LINE 60
      $DTFO D1403-Y                 DEFINE OFFSETS IN PRINTER DTF
DTF2  $DTFD AC-CO,BUFNO-1,RECL-80,  DTF FOR DISK                   X
            IO-IOB,BLKL-512,                                       X
            NAME-SAMPLE,RCAD-BUF1,                                 X
            DISK-5444
BUF1  EQU   *                       LOGICAL RECORD FOR PRINTER
      DC    CL96' '
      ORG   *,256,X'7C'             ALIGN ON 124-BYTE BOUNDARY
BUF2  EQU   *                       PRINT PHYSICAL BUFFER
      DC    CL96' '
IOB   EQU   *                       I/O AREA FOR DISK
      DS    30XLL
      DS    512XL1
```

Figure 15 (Part 3 of 4). Sample Program



```
STATEMENT
Name | Operation | Operand | Remarks
IOB1  EQU   *                       IOB AREA FOR PRINTER
      DS    CL50
      ORG   *,256,X'80'             ALIGN ON 256-BYTE BOUNDARY
NXTBUF EQU  *                       NEXT BUFFER FOR SYSIN
      DC    CL128' '
CURBUF EQU  *                       CURRENT BUFFER FOR SYSIN
      DC    CL128' '
WRKBUF EQU  *                       WORK BUFFER FOR SYSIN
      DC    CL97' '
      $LOGD
SYSRD $RLST BUF1-CURBUF,WORK-WRKBUF,BUF2-NXTBUF
      $RLSD
HALTA1 $LMSG FORMAT-A,HALT-A1,OPN2-Y,DEF-2
HALTA2 $LMSG FORMAT-A,HALT-A2,OPN2-Y,DEF-2
HALTA3 $LMSG FORMAT-A,HALT-A3,OPN2-Y,DEF-2
      END   SAMPLE                  END OF SAMPLE PROGRAM
/*
*********************************************************************
***   OCL TO ASSEMBLE THE PROGRAM                                  ***
*********************************************************************
// LOAD $ASSEM,F1                   LOAD ASSEMBLER FROM DRIVE F1
// SWITCH 1XXXXXX                   INDICATE INPUT IS ON $SOURCE
// FILE NAME-$SOURCE,PACK-VOL001,UNIT-R1,
// RETAIN-S,TRACKS-30               FILE STATEMENT FOR $SOURCE
// FILE NAME-$WORK,PACK-VOL010,UNIT-R2,
// RETAIN-S,TRACKS-10               FILE STATEMENT FOR $WORK
// FILE NAME-$WORK2,PACK-VOL010,UNIT-R2,
// RETAIN-S,TRACKS-10               FILE STATEMENT FOR $WORK2
// RUN                              ASSEMBLE THE JOB.
/&
```

Figure 15 (Part 4 of 4). Sample Program

Macro Instructions Used in the Sample Program

Fifteen macro instructions are used in this sample program.
The macro instructions and their functions are:

Macro Instruction	Function
$ALOC	Allocates the printer file to this program.
$OPEN	Opens the file after allocation.
$CLOS	Closes the output file.
$DTFD	Constructs the DTF for disk.
$EOJ	Calls the end-of-job routine.
$READ	Reads input records from the system reader.
$PUTP	Prints output records on the printer.
$DTFP	Constructs the DTF for the printer.
$DTFO	Establishes the offsets for the printer and disk DTFs.
$RLSD	Generates offsets for the system reader parameter list.
$RLST	Constructs the parameter list for the system reader.
$LOGD	Generates offsets for the system log parameter list.
$LOG	Logs message on the system log device.
$LMSG	Constructs the parameter list for the system log.
$PUTD	Puts records to a disk file.

Error Code	Error Description
[illegible]	A keyword response has resulted in an invalid decimal digit or a boundary exceeded condition. (See note.)
OC	An error in continuation exists in this macro instruction. Nonblank characters were found in columns 1-13 of the continued line. All remaining lines of this macro instruction will be flagged with the error code 'OC'.
[illegible]	A keyword response resulted in a character string that exceeds the maximum length. (See note.)
[illegible]	An error in continuation exists in the previous macro instruction. Column 72 is blank. All remaining lines of this macro instruction will be flagged with the error code 'OC' or 'NF'.
[illegible]	A delimiter is missing or invalid in the operand of the previous macro instruction.
[illegible]	A keyword in the macro instruction being processed is not valid.
[illegible]	An invalid parameter has been found in one of the operands of the previous macro instruction.
[illegible]	The macro instruction being processed contains a mnemonic operation code not contained in the source library of the program pack.
[illegible]	The mnemonic operation code of the previous macro instruction is invalid. (See codes 'CE' and 'IF' for a possible cause for this error code.)
[illegible]	A keyword response has resulted in an invalid substring term. (See note.)
[illegible]	A keyword response has resulted in a substring syntax error. (See note.)
[illegible]	The variable symbol table is full. Recode your program using fewer macro instructions.

Note: These errors may be the result of any macro instruction [illegible] macro instructions that precede the error [illegible]

Figure 16. Macro Instruction Error Codes

Appendix B: Define the File Control Blocks

The DTF provides information to the data management routines about files you use. You must provide one DTF for each file you use in a program. Certain fields serve the same purpose in all pre-open DTFs. (Pre-open DTFs are reformatted to post-open when they are opened by using the allocate and open macro instructions.)

The figures in this appendix describe both the pre-open and post-open DTFs for unit record and disk devices.

Figure	DTF Described
17	MFCU
18	MFCM
19	1442
20	Line Printer
21	2501
22	Disk
23	Tape
24	Device Independent
25	CRT/Keyboard
26	3741

The labels given to the fields in these figures are the labels generated by the offset macro instruction $DTFO. Displacements are hexadecimal numbers which refer to the rightmost byte of the field; length is specified in bytes. Addresses in the DTFs point to the leftmost byte of the referenced area.

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code MFCU1 (primary hopper of MFCU) = X'F0' MFCU2 (secondary hopper of MFCU) = X'F8'
$DFUPS	1	1	External indicators
$DFAT1	2	1	Attribute byte 1 **Bit On — Meaning** 0 — Read 1 — Print 2 — Punch 3 — Card image mode
$DFAT2	3	1	Attribute byte 2 **Bit On — Meaning** 2 — Device is system reader 3 — Multiple buffers 4 — EOF on multiple buffers 5 — /. or /& read on last input operation 7 — DTF has been opened
$DFCHA	5	2	Address of next DTF in backward chain
$DFCHB	7	2	Address of next DTF in forward chain
$DFARR	9	2	Address recall register save area (return address)
$DFXRS	B	2	XR1 save area (contents of calling program register 1)
$DFLRA	D	2	Logical record address
$DFCMP	E	1	Completion code **Code — Meaning** X'40' — Successful completion X'41' — Permanent error X'42' — End of file indicator

Figure 17 (Part 1 of 2). MFCU DTF

Field Name	Displacement	Length	Comments
$DFOPC	F	1	Open options Bit On — Meaning 0 — [illegible] 1 — [illegible] 2 — [illegible] 3 — [illegible] 4 — [illegible] 5 — [illegible] 6 — [illegible]
$DFSTS	10	1	Stacker select
$DFCIA	12	2	IOB address[1]
$DFRDA	14	2	[illegible] address[1]
$DFPNA	16	2	[illegible][1]
$DFPIA	18	2	[illegible] address[1]
$DFNIO	19	1	Number of IOBs[1]
$DFO	1A	1	[illegible]
$DFR	1B	1	[illegible]
$DFRSV	1C	1	[illegible]

[1] Indicates field is used for pre open DTF.

Figure 17 (Part 2 of 2). MFCU DTF

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code MFCM1 (primary hopper of MFCM) = X'F0' MFCM2 (secondary hopper of MFCM) = X'F8'
$DFUPS	1	1	External indicators
$DFAT1	2	1	Attribute byte 1 **Bit On — Meaning** 0 Read 1 Print 2 Punch 3 Card image mode 4 DTF for MFCM 5 Interpret mode 6 Print head 5 or 6 to be used
$DFAT2	3	1	Attribute byte 2 **Bit On — Meaning** 2 Device is system reader 3 Multiple buffers (data) 4 EOF on multiple buffers 5 /. or /& read on last input operation 7 DTF has been opened
$DFCHA	5	2	Address of next DTF in backward chain
$DFCHB	7	2	Address of next DTF in forward chain
$DFARR	9	2	Address recall register save area (return address)
$DFXRS	B	2	XR1 save area (contents of calling program register 1)
$DFLRA	D	2	Logical record address
$DFCMP	E	1	Completion code **Code — Meaning** X'40' Successful completion X'41' Permanent error X'42' End of file

Figure 18 (Part 1 of 2). MFCM DTF

Field Name	Displacement	Length	Contents
$DFOPC	F	1	Operation code Bit On — Meaning 0 — Read 1 — Print 2 — Punch 3 — Defer operation 4 — Card image 5 — No feed 6 — Four tier printing
$DFSTS	10	1	Stacker select
$DFCIA	12	2	IOB
$DFRDA	14	2	Read I/O area address[1]
$DFPNA	16	2	Punch I/O area address[1]
$DFPTA	18	2	Print I/O area address[1]
$DFNIO	19	1	Number of IOBs[1]
$DFQ	1A	1	Q-byte (device address)
$DFR	1B	1	R-byte
$DFRDL	1C	1	Read length[1]
$DFPNL	1D	1	Punch length
$DFPTL	1E	1	Print length
$DFPHS	1F	1	Print head select
$DFIND	20	1	Indicator byte Bit On — Meaning 0 — Previous operation was print 1 — Feed required before print 2 — Read already completed 3 — First-time bit for interpret

[1] Indicates field is used for pre-open DTF.

Figure 18 (Part 2 of 2). MFCM DTF

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code X'50'
$DFUPS	1	1	External indicators
$DFAT1	2	1	Attribute byte 1 **Bit On** — **Meaning** 0 — Read 2 — Punch 3 — Card image mode
$DFAT2	3	1	Attribute byte 2 **Bit On** — **Meaning** 2 — Device is system reader 3 — Multiple buffers (data) 4 — EOF on multiple buffers 5 — /. or /& read on last input operation 7 — DTF has been opened
$DFCHA	5	2	Address of next DTF in backward chain
$DFCHB	7	2	Address of next DTF in forward chain
$DFARR	9	2	Address recall register save area (return address)
$DFXRS	B	2	XR1 save area (contents of calling program register 1)
$DFLRA	D	2	Logical record address
$DFCMP	E	1	Completion code **Code** — **Meaning** X'40' — Successful completion X'41' — Permanent error X'42' — End of file
$DFOPC	F	1	Operation code **Bit On** — **Meaning** 0 — Read 1 — Print 2 — Punch 3 — Defer operation 4 — Card image mode 5 — No feed

Figure 19 (Part 1 of 2). 1442 DTF

Field Name	Displacement	Length	Contents
$DFSTS	10	1	Stacker select (MFCU, MFCM, 1442)
$DFCIA	12	2	IOB address[1]
$DFRDA	14	2	Read I/O area address[1]
$DFPNA	16	2	Punch I/O area address[1]
$DFNIO	19	3	Number of IOBs[1]
$DFQ	1A	1	Q-byte (device address)
$DFR	1B	1	R-byte
$DFPNL	1D	2	Punch length

[1] Indicates field is used for pre-open DTF.

Figure 19 (Part 2 of 2). 1442 DTF

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code **Code** **Meaning** X'E0' 1403 X'11' 3284
$DFUPS	1	1	External indicators
$DFAT1	2	1	Attribute byte 1 **Bit On** **Meaning** 1 Print
$DFAT2	3	1	Attribute byte 2 **Bit On** **Meaning** 6 Halt on unprintable character 7 DTF has been opened
$DFCHA	5	2	Address of next DTF in backward chain
$DFCHB	7	2	Address of next DTF in forward chain
$DFARR	9	2	Address recall register save area (return address)
$DFXRS	B	2	XR1 save area (contents of calling program register 1)
$DFLRA	D	2	Logical record address
$DFCMP	E	1	Completion code **Code** **Meaning** X'40' Successful completion X'41' Permanent error X'42' End of file X'48' Overflow on the printer
$DFOPC	F	1	Operation code **Bit On** **Meaning** 1 Print
$DFSKB	10	1	Line number to skip to before the print operation
$DFSPB	11	1	Number of lines to space before the print operation
$DFSKA	12	1	Line number to skip to after the print operation
$DFSPA	13	1	Number of lines to space after the print operation

Figure 20 (Part 1 of 2). Line Printer DTF

Field Name	Displacement	Length	Contents
$DFPQ	14	1	Q-byte (device address)
$DFPR	15	1	R-byte
$DFPIB	17	2	IOB address[1]
$DFPIO	19	2	I/O area address[1]
$DFPRL	1B	2	Record length
$DFOVF	1C	1	Overflow line number[1]
$DFLP	1D	1	Lines per page[1]
$DFPOS	1E	1	Position counter
$DFMSK	1F	1	Maximum skip value[1]
$DFPGS	20	1	Page size save area

[1] Indicates field is used for pre-open DTF.

Figure 20 (Part 2 of 2). Line Printer DTF

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code X'38'
$DFUPS	1	1	External indicators
$DFAT1	2	1	Attribute byte 1 **Bit On** — **Meaning** 0 — Read 3 — Card image mode
$DFAT2	3	1	Attribute byte 2 **Bit On** — **Meaning** 2 — Device is system reader 3 — Multiple buffers are being used 4 — End-of-file on multiple buffers 6 — /. or /& was read 7 — DTF has been opened
$DFCHA	5	2	Address of next DTF in backward chain
$DFCHB	7	2	Address of next DTF in forward chain
$DFARR	9	2	Address recall register save area (return address)
$DFXRS	B	2	XR1 save area (contents of calling program register 1)
$DFLRA	D	2	Logical record address
$DFCMP	E	1	Completion code **Code** — **Meaning** X'40' — Successful completion X'41' — Permanent error X'42' — End of file
$DFOPC	F	1	Operation code **Bit On** — **Meaning** 0 — Read 3 — Defer operation 4 — Card image mode 5 — No feed

Figure 21 (Part 1 of 2). 2501 DTF

Field Name	Displacement	Length	Contents
$DFCIA	12	3	IOB address[1]
$DFRDA	14	2	Read I/O area address[1]
$DFNIO	19	5	Number of IOBs[1]
$DFQ	1A	1	Q-byte (device address)
$DFR	1B	1	R-byte
$DFRDL	1C	1	Read length

[1] Indicates field is used for pre-open DTF.

Figure 21 (Part 2 of 2). 2501 DTF

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code:[1] R1 (removable disk – 5444, drive 1) = X'A0' F1 (fixed disk – 5444, drive 1) = X'A8' R2 (removable disk – 5444, drive 2) = X'B0' F2 (fixed disk – 5444, drive 2) = X'B8' D1 (5445, drive 1) = X'C0' D2 (5445, drive 2) = X'C8' D3 (5445, drive 3) = X'D0' D4 (5445, drive 4) = X'D8'
$DFUPS	1	1	External indicators[1]
$DFAT1	2	1	Attribute byte 1[1] **Bit On** **Meaning** 0 Indexed 1 Consecutive 2 Direct 3 Multivolume 4 Input 5 Output 6 Update 7 Add
$DFAT2	3	1	Attribute byte 2[1] **Bit On** **Meaning** 3 Multiple buffers 7 DTF has been opened
$DFCHA	5	2	DTF chain pointer A (backward)
$DFCHB	7	2	DTF chain pointer B (forward)[1]
$DFARR	9	2	ARR save area (return address)
$DFXRS	B	2	XR1 save area (contents of object program XR1)
$DFLRA	D	2	Address of logical record (shared I/O address of logical input record)[1]

[1] Indicates field is used for pre-open DTF.

Figure 22 (Part 1 of 7). Disk DTF (5704-SC1 only)

Field Name	Displacement	Length	Contents
$DFCMP	E	1	Completion code **Code** **Meaning** X'40' Normal completion X'41' Controlled cancel taken on permanent I/O error X'42' End of file (input) X'44' No record found (out of extent for direct files) X'50' Key field does not match key in update record X'60' Duplicate key on attempted load or add X'62' Keys out of sequence (attempted load or add) X'64' Key low for this volume or high for any volume X'68' Key low for this volume X'70' End of extent or end of read (output) X'72' Key high for last volume Completion codes other than X'40' are returned before the data management function is actually completed.
$DFOPC	F	1	Operation Code: **Bit(s) On** **Meaning** 0 Get 1 Put/add or put/load 2 Put/update 0,3 Set new limits
$DFIOB	11	2	Pre-open address of I/O area[1] Post open address of IOB
$DFPRB	13	2	Address of current process IOB (dual I/O only; shared I/O—address of logical output record).
$DFBKL	15	2	Block length (length of data buffer)[1]
$DFRCL	17	2	Logical record length[1]
$DFPTR	19	2	Data block index (address of next record)
$DFR01	1A	1	Reserved
$DFXTA	1C	2	Data start extent
$DFMVF	1C	(2)	Address of direct MVF extent table[1]
$DFR02	1D	1	Reserved

[1] Indicates field is used for pre-open DTF.

Figure 22 (Part 2 of 7). Disk DTF (5704-SC1 only)

Field Name	Displacement	Length	Contents
$DFXTB	1F	2	Data end extent (disk address)
$DFNUM	1F	(2)	Number of extents (direct MVF)[1]
$DFSWA	20	1	Scheduler work area format—1 label sequence number
$DFNAM	28	8	File name[1]
$DFAT3	29	1	Attribute byte 3[1] **Bit On — Meaning** 0 — Addrout 1 — Ordered load 2 — Random 3 — Limits 4 — End of limits – call to open 5 — Force end of volume – call to close 6 — Online multivolume 7 — Reserved
$DFAT4	2A	1	Attribute byte 4[1] **Bit On — Meaning** 0 — Compiler access method 1-7 — Reserved
$DFAT5	2B	1	Attribute byte 5 (Bits 0-7 are reserved)[1]
$DFSEC	2C	1	Number of sectors to write (split)
$DFWAA	2D	1	Work area A (disk)
$DFWAB	2E	1	Work area B (disk)
$DFWAC	2F	1	Work area C (disk)
$DFWAD	30	1	Work area D (disk)
$DFR04	31	1	Reserved
$DFRMA	34	3	Work area, length of first part of overlap record
$DFR06	35	1	Reserved
$DFRMB	38	3	Work area, length of second part of overlap record

[1] Indicates field is used for pre-open DTF.

Figure 22 (Part 3 of 7). Disk DTF (5704-SC1 only)

Field Name	Displacement	Length	Contents
$DFND1	39	1	Indicator byte 1 **Bit On — Meaning** 0-3 — Reserved 4 — Pseudo-get 5 — High key loaded (indexed output – MVF) 6 — Low key was found 7 — Buffer has been written
$DFND2	3A	1	Indicator byte 2 **Bit On — Meaning** 0 — Indexed random update – first time indicator 1 — First record on new volume 2 — MVF end of file 3 — End of all MVF files 4 — Empty file (skip initial index search) 5 — Previous operation was add 6 — End of file has been reached 7 — EOF on this get (limits not set)
$DFND3	3B	1	Indicator byte 3 **Bit On — Meaning** 0 — Index contains adds or new entries 1-3 — Reserved 4 — Records added 5 — Out of sequence add (key sort needed) 6 — Current process buffer contains adds 7 — Current process buffer contains updates
$DFRO7	3C	1	Reserved
$DFR08	3D	1	Reserved area 1 (5444 only)
$DFNXR	41	4	Disk address of next record (CSDD)
$DFSPC	42	1	Number of tracks in cylinder
$DFR09	43	1	Reserved
$DFIOA	46	3	Disk address save area
$DFDAT	48	2	Pointer to current index entry.
End of Disk DTF for consecutive output			

Figure 22 (Part 4 of 7). Disk DTF (5704-SC1 only)

Field Name	Displacement	Length	Contents
$DFR10	49	1	Reserved
$DFR11	4A	1	Reserved area 1 byte (5444 only)
$DFEOF	4D	3	Disk address of logical end of file (CSD)
End of Disk DTF for consecutive input and update			
$DFNXK	4D	(3)	Disk address of logical end of index (CSD)
$DFKPR	4F	2	Pointer written index (pointer to next buffer entry)
$DFKAD	51	2	Address of user's key area[1]
$DFKCR	53	2	Address of key in core (direct)
End of Disk DTF for direct			
$DFCUR	53	(2)	Address of current key (index)
$DFHI	53	(2)	Address of high key (limits)
$DFR12	54	1	Reserved
$DFKXA	56	2	Start extent of index (disk address of first track)
$DFKBF	58	2	Address of index IOB
$DFKL	5A	2	Key length[1]
$DFR13	5B	1	Reserved
$DFKXB	5E	3	Disk address of end of index (CSD)
$DFKD	60	2	Displacement of key in record[1]
$DFBLX	61	1	Index block size[1]
$DFR14	62	1	Reserved
$DFFLX	64	2	Disk address of start of index (5445)
$DFR15	65	1	Reserved

[1] Indicates field is used for pre-open DTF.

Figure 22 (Part 5 of 7). Disk DTF (5704-SC1 only)

Field Name	Displacement	Length	Contents
$DFDTX	67	2	Start of disk track index (5445) disk address
$DFABF	69	2	Address of add index IOB
$DFAPR	6B	2	Add index buffer pointer
End of Disk DTF for index sequential input output, and update			
$DFMIX	6D	2	Address of in-core index (random)[1]
$DFLOW	6F	2	Address of low key (limit)
End of Disk DTF for input or update with limits			
$DFLST	6F	(2)	Address of last key (indexed sequential)
$DFHAD	71	2	Address of high add key bucket[1]
$DFLOT	71	(2)	Address of save area for current key[1]
$DFHPK	73	2	Address of high primary key
$DFBYT	75	2	Number of bytes in in-core index[1]
End of Disk DTF for index random input and update			
$DFAPT	77	2	Pointer to next record in add buffer
$DFR16	78	1	Reserved
$DFR17	79	1	Reserved area 1 byte (5444 only)
$DFKXP	7C	3	Disk address of end of primary index
End of Disk DTF for index random add, retrieve add, and update add			
$DFSLP	7E	2	Save area for last index disk pointer
$DFR18	7F	1	Reserved
$DFSLA	82	3	Save area for last index disk address
End of Disk DTF for index sequential add and update add			

[1] Indicates field is used for pre-open DTF.

Figure 22 (Part 6 of 7). Disk DTF (5704-SC1 only)

Field Name	Displacement	Length	Contents
$DFSTA	84	2	Multivolume file table pointer
$DFSEQ	85	1	Volume sequence number of current volume
$DFNXT	86	1	Volume sequence number of next volume
$DFF1S	87	1	Start of format-1
$DFF1	C6	63	Format-1 save area
$DFAR1	C8	2	EOV save area
$DFXR1	CA	2	EOV save area
$DFKEY	CC	2	Address of high key from Format-7
$DFTAB	CE	2	Address of multivolume information table
$DFENT	D0	2	Number of entries in in-core index
$DFVOL	D1	1	Number of volumes online
End of Disk DTF for multivolume file processing			

Figure 22 (Part 7 of 7). Disk DTF (5704-SC1 only)

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code:[1] R1 (simulation area, drive 1) = X'A0' F1 (simulation area, drive 1) = X'A8' R2 (simulation area, drive 2) = X'B0' F2 (simulation area, drive 2) = X'B8' D1 (3340 drive 1) = X'C0' D2 (3340 drive 2) = X'C8' D3/D31 (3340 drive 3) = X'D0' or (3344 drive 3, logical unit 1) = X'D0' D4/D41 (3340 drive 4) = X'D8' or (3344 drive 4, logical unit 1) = X'D8' D32 (3344 drive 3, logical unit 2) = X'D1' D33 (3344 drive 3, logical unit 3) = X'D2' D34 (3344 drive 3, logical unit 4) = X'D3' D42 (3344 drive 4, logical unit 2) = X'D9' D43 (3344 drive 4, logical unit 3) = X'DA' D44 (3344 drive 4, logical unit 4) = X'DB'
$DFUPS	1	1	External indicators[1]
$DFAT1	2	1	Attribute byte 1[1] **Bit On — Meaning** 0 Indexed 1 Consecutive 2 Direct 3 Multivolume 4 Input 5 Output 6 Update 7 Add
$DFAT2	3	1	Attribute byte 2[1] **Bit On — Meaning** 0 Device independent DTF 1 DTF allocated 2 Device is SYSIN 3 Multiple buffers data 4 Deferred open 5 EOV -- Close ignored bit 6 EOV -- Call close 7 Opened
$DFCH	5	2	DTF chain pointer A (backward)

[1] Indicates field is for pre-open DTF.

Figure 22a (Part 1 of 7). Disk DTF (5704-SC2 only)

Field Name	Displacement	Length	Contents
$DFCHB	7	2	DTF chain pointer B (forward)[1]
$DFARR	9	2	ARR save area (return address)
$DFXRS	B	2	XR1 save area (contents of object program XR1)
$DFLRA	D	2	Address of logical record (shared I/O address of logical input record)[1]
$DFCMP	E	1	Completion code: **Code** — **Meaning** X'00' — Record not found (requested key low) X'40' — Successful completion X'41' — Permanent error X'42' — End of file X'44' — No record found X'50' — Update key not equal to key X'60' — Duplicate key on load or add X'62' — Keys out of sequence on load or add X'64' — Key low for this volume or high for any volume X'68' — Key low for this volume X'70' — End of extent or end of reel X'72' — Key high for last volume Completion codes other than X'40' are returned before the data management function is actually completed.
$DFOPC	F	1	Operation code: **Bit(s) On** — **Meaning** X'30' — Get X'40' — Put/update X'60' — Put/add or put/load X'90' — Set new limits
$DFIOB	11	2	Pre-open address of I/O area[1] Post open address of IOB
$DFPRB	13	2	Address of current process IOB (dual I/O only; shared I/O — address of logical output record).
$DFBKL	15	2	Block length (length of data buffer)[1]
$DFRCL	17	2	Logical record length[1]
$DFPTR	19	2	Data block index (address of next record)

[1] Indicates field used for pre-open DTF.

Figure 22a (Part 2 of 7). Disk DTF (5704-SC2 only)

Field Name	Displacement	Length	Contents
$DFSEG	1A	1	File share segment number
$DFXTA	1C	2	Disk address of start of data
$DFMVF	1C	(2)	Address of direct MVF extent table[1]
$DFQB	1D	1	Device-independent Q code
$DFXTB	1F	2	Disk address of end of data
$DFNUM	1F	(2)	Number of extents (direct MVF)[1]
$DFSWA	20	1	Scheduler work area format-1 label sequence number
$DFNAM	28	8	File name[1]
$DFAT3	29	1	Attribute byte 3[1]: Bit On — Meaning 0 — Addrout 1 — Ordered load 2 — Random 3 — Limits 4 — End of limits – call to open 5 — Force end of volume – call to close 6 — Online multivolume 7 — Reserved
$DFAT4	2A	1	Attribute byte 4[1]: Bit On — Meaning 0 — Compiler access method 1 — Bypass direct file clear 2 — 3340s supported 3 — No verify (main data area) 4 - 6 — Reserved 7 — Consecutive add from start of file
$DFAT5	2B	1	Attribute byte 5[1]: Bit On — Meaning 0 — Format-1 for this DTF 1 — File share 2 — Copying a PTAM file 3 — External buffers 4 — External buffers closed 5 — Multivolume file share support 6 — CCP task 7 — Reserved

[1] Indicates field used for pre-open DTF.

Figure 22a (Part 3 of 7). Disk DTF (5704-SC2 only)

Field Name	Displacement	Length	Contents
$DFSEC	2C	1	Number of sectors to write (split)
$DFWAA	2D	1	Work area A (disk)
$DFWAB	2E	1	Work area B (disk)
$DFWAC	2F	1	Work area C (disk)
$DFWAD	30	1	Work area D (disk)
$DFR04	31	1	Reserved
$DFR06	32	1	Reserved
$DFRMA	35	3	Work area, length of first part of overlap record
$DFRMB	38	3	Work area, length of second part of overlap record
$DFND1	39	1	Indicator byte 1: Bit On — Meaning 0 — Buffer is result of scan read 1 — Switch buffers or pointers in subroutine 2 — Update pointers in share DTF 3 — Partially filled add buffer written 4 — Pseudo get 5 — High key loaded (indexed output – MVF) 6 — Low key was found 7 — Buffer has been written
$DFND2	3A	1	Indicator byte 2: Bit On — Meaning 0 — Indexed random update – first time indicator 1 — First record on new volume 2 — MVF end of file 3 — End of all MVF files 4 — Empty file (skip initial index search) 5 — Previous operation was add 6 — End of file has been reached 7 — EOF on this get (limits not set)
$DFND3	3B	1	Indicator byte 3: Bit On — Meaning 0 — Current process buffer contains update 1 — Added entries in input index buffer 2 — Records in add buffer were updated 3 — End of data in input buffer – indexed 4 — Records added 5 — Out of sequence add (key sort needed) 6 — Current process buffer contains adds 7 — New entries in add index buffer

[1] Indicates field is used for pre-open DTF.

Figure 22a (Part 4 of 7). Disk DTF (5704-SC2 only)

Field Name	Displacement	Length	Contents
$DFWAE	3C	1	Work area E (disk)
$DFPUN	3D	1	Physical unit simulation area only
$DFNXR	41	4	Disk address of next record (CSDD)
$DFSHR	43	2	Address of file share DTF
$DFIOA	46	3	Disk address save area
$DFDAT	48	2	Pointer to current index entry
$DFND5	49	1	Indicator byte 5 **Code** — **Meaning** 0 — Get has not been done 1 — Do not flush buffers in close 2 - 7 — Reserved
$DFR11	4A	1	Reserved (simulation area only)
End of Disk DTF for consecutive output			
$DFEOF	4D	3	Disk address of logical end of file (CSD)
End of Disk DTF for consecutive input and update			
$DFNXK	4D	(3)	Disk address of logical end of index (CSD)
$DFKPR	4F	2	Pointer written index (pointer to next buffer entry)
$DFKAD	51	2	Address of user's key area[1]
$DFKCR	53	2	Address of key in core (direct)
End of Disk DTF for direct			
$DFCUR	53	(2)	Address of current key (index)
$DFHI	53	(2)	Address of high key (limits)
$DFR12	54	1	Reserved
$DFKXA	56	2	Start extent of index (disk address of first track)
$DFKBF	58	2	Address of index IOB

[1] Indicates field is used for pre-open DTF.

Figure 22a (Part 5 of 7). Disk DTF (5704-SC2 only)

Field Name	Displacement	Length	Contents
$DFKL	5A	2	Key length[1]
$DFND4	5B	1	Indicator byte 4 Code — Meaning 0 — First key invalid in index buffer 1 — First add buffer not filled 2 — Add area already formatted 3-7 — Reserved
$DFKXB	5E	3	Disk address of end of index (CSD)
$DFKD	60	2	Displacement of key in record[1]
$DFBLX	61	1	Index block size[1]
$DFR14	62	1	Reserved
$DFR15	63	1	Reserved
$DFFLX	65	2	Disk address of start of index (main data area)
$DFDTX	67	2	Start of disk track index disk address (main data area)
$DFABF	69	2	Address of add index IOB
$DFAPR	6B	2	Add index buffer pointer
End of Disk DTF for index sequential input output, and update			
$DFMIX	6D	2	Address of in-core index (random)[1]
$DFLOW	6F	2	Address of low key (limit)
End of Disk DTF for input or update with limits			
$DFLST	6F	(2)	Address of last key (indexed sequential)
$DFHAD	71	2	Address of high add key bucket[1]
$DFHPK	73	2	Address of high primary key
$DFBYT	75	2	Number of bytes in in-core index[1]
End of Disk DTF for index random input and update			

[1] Indicates field is used for pre-open DTF.

Figure 22a (Part 6 of 7). Disk DTF (5704-SC2 only)

Field Name	Displacement	Length	Contents
$DFAPT	77	2	Pointer to next record in add buffer
$DFR16	78	1	Reserved
$DFR17	79	1	Reserved area (simulation area only)
$DFKXP	7C	3	Disk address of end of primary index
End of Disk DTF for index random add, retrieve add, and update add			
$DFSLP	7E	2	Save area for last index disk pointer
$DFR18	7F	1	Reserved
$DFSLA	82	3	Save area for last index disk address
End of Disk DTF for index sequential add and update add			
$DFSTA	84	2	Multivolume file table pointer
$DFSEQ	85	1	Volume sequence number of current volume
$DFNXT	86	1	Volume sequence number of next volume
$DFF1S	87	1	Start of format-1
$DFF1	C6	63	Format-1 save area
$DFAR1	C8	2	EOV save area
$DFXR1	CA	2	EOV save area
$DFKEY	CC	2	Address of high key from Format-7
$DFTAB	CE	2	Address of multivolume information table
$DFENT	D0	2	Number of entries in in-core index
$DFVOL	D1	1	Number of volumes online
End of Disk DTF for multivolume file processing			

Figure 22a (Part 7 of 7). Disk DTF (5704-SC2 only)

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code: T1 (tape unit 1) = X'60' T2 (tape unit 2) = X'68' T3 (tape unit 3) = X'70' T4 (tape unit 4) = X'78'
$DFUPS	1	1	External indicators
$DFAT1	2	1	Attribute byte 1 **Bit On** **Meaning** 1 Consecutive 2 Direct 5 Output
$DFAT2	3	1	Attribute byte 2 **Bit On** **Meaning** 3 Multiple buffers (data) 7 DTF has been opened
$DFCHA	5	2	Address of next DTF in backward chain
$DFCHB	7	2	Address of next DTF in forward chain
$DFARR	9	2	Address recall register save area (return address)
$DFXRS	B	2	XR1 save area (contents of calling program register 1)
$DFLRA	D	2	Logical record address
$DFCMP	E	1	Completion code **Code** **Meaning** X'40' Successful completion X'41' Permanent error X'45' Skip a bad input block X'70' End of extent or end of reel X'90' Wrong length record on input – short X'91' Wrong length record on input – long X'F0' Option 0 to error message X'F1' Option 1 to error message

Figure 23 (Part 1 of 4). Tape DTF

Field Name	Displacement	Length	Contents
$DFOPC	F	1	Operation code **Code** **Meaning** X'04' Read forward X'06' Read backward X'08' Write X'0A' Write tape mark X'0B' Rewind X'0C' Backspace file X'0D' Rewind and unload X'0E' Forward space file X'1C' Backspace block X'1E' Forward space block X'40' Variable length X'80' Fixed length
$DFIOB	11	2	Pre-open address of I/O area[1] Post-open address of IOB
$DFPRB	13	2	Address of process IOB
$DFLIO	13	(2)	Length of I/O area[1]
$DFBKL	15	2	Block length[1]
$DFRCL	17	2	Record length[1]
$DFPTR	19	2	Pointer to logical record in buffer
$DFCRL	1B	2	Current record length
$DFBCT	1D	2	Block count
$DFR03	1F	2	Reserved
$DFSWA	20	1	Scheduler work area format—1 index number
$DFNAM	28	8	File name[1]

[1] Indicates field is used for pre-open DTF.

Figure 23 (Part 2 of 4). Tape DTF

Field Name	Displacement	Length	Contents
$DFAT3	29	1	Attribute byte 3 Bits 0-5 are record format bits **Bit On** — **Meaning** 0 — Fixed length 1 — Variable length 2 — Unblocked records 3 — Blocked records 4 — Spanned records 5 — ASCII format D. 6 — ASCII data management present (pre-open) 7 — ASCII file (post-open)
$DFAT4	2A	1	Attribute byte 4 **Bit On** — **Meaning** 0,1 — Rewind at close 0 — Unload at close 1 — Leave at close 2 — Basic access method 3 — Standard label tape 4 — Locate mode 5 — No multivolume output 6 — DTF closed by EOV 7 — Reel opened on this file
$DFAT5	2B	1	Attribute byte 5 (Bits 0-7 are reserved)
$DFHTC	2C	1	Error halt code **Code** — **Meaning** X'10' — No get operation code X'11' — Wrong length record on input X'12' — Wrong length block X'20' — No put operation code X'21' — Wrong length record on output X'40' — Permanent tape write error at close X'41' — Wrong block count X'50' — Permanent tape read error at close X'60' — No file statement X'61' — File not allocated X'62' — Not enough buffer space X'70' — No basic operation code

Figure 23 (Part 3 of 4). Tape DTF

Field Name	Displacement	Length	Contents
$DFSQU	2D	1	Format–1 sequence number
$DFOST	2E	1	ASCII buffer offset byte **Code** **Meaning** X'80' Offset is 4 bytes for block length X'00' No offset specified Bits 1-7 are reserved for offsets of 0-99.
$DFWRA	30	2	Work area A
$DFWRB	32	2	Work area B
$DFWRC	34	2	Work area C
$DFWRD	36	2	Work area D
$DFR05	38	2	Reserved
$DFND1	39	1	Indicator byte 1 **Bit On** **Meaning** 0 IOB not waited on 1 Truncated block 2 Empty variable block 3 Error reading trailer label 4 Write trailer label 5 DTF closed by EOV 6 EOV return via open 7 No end of file reached
$DFND2	3A	1	Indicator byte 2 (bits 0-7 are reserved)
$DFHSA	3C	2	Halt routine save area
$DFSR1	3E	2	EOV register 1 save area
$DFSVD	40	2	EOV chain DTF address save area
$DFSVE	42	2	EOV save area

Figure 23 (Part 4 of 4). Tape DTF

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code X'40'
$DFUPS	1	1	External indicators
$DFAT1	2	1	Attribute byte 1 **Bit On** — **Meaning** 0 — Indexed 1 — Consecutive 2 — Direct 3 — Multivolume 4 — Input 5 — Output 6 — Update 7 — Add
$DFAT2	3	1	Attribute byte 2 **Bit On** — **Meaning** 0 — Device independent DTF 3 — Multiple buffers (data) 5 — /& read on last input operation 7 — DTF has been opened
$DFCHA	5	2	Address of next DTF in backward chain
$DFCHB	7	2	Address of next DTF in forward chain
$DFARR	9	2	Address recall register save area (return address)
$DFXRS	B	2	XR1 save area (contents of calling program register 1)
$DFLRA	D	2	Logical record address
$DFCMP	E	1	Completion code **Code** — **Meaning** X'40' — Successful completion X'41' — Permanent error X'42' — End of file X'70' — End of extent or end of reel

Figure 24 (Part 1 of 5). Device Independent DTF

Field Name	Displacement	Length	Contents
$DFOPC	F	1	Operation code **Bit On — Meaning** 0 — Get 1 — Put
$DFIOB	11	2	Pre-open address of I/O area[1] Post-open address of IOB
$DFPRB	13	2	Address of process IOB
$DFBKL	15	2	Block length[1]
$DFRCL	17	2	Record length[1]
$DFPTR	19	2	Pointer to logical record in buffer
$DFPOV	1A	1	Printer overflow
$DFR01	1A	(1)	Reserved
$DFXTA	1C	2	Disk address of start of data
$DFQB	1D	1	Q-byte (device address)
$DFR02	1D	(1)	Reserved
$DFBCT	1D	(1)	Block count
$DFXTB	1F	2	Disk address of end of data
$DFSWA	20	1	Scheduler work area format—1 index number
$DFNAM	28	8	File name[1]
$DFAT3	29	1	Attribute byte 3 Bits 0-5 are record format bits **Bit On — Meaning** 0 — Fixed length 1 — Not used 2 — Unblocked records 3 — Blocked records 4 — Spanned records 5 — ASCII format D 6 — ASCII data management present (pre-open) 7 — ASCII file (post-open)

[1] Indicates field is used for pre-open DTF.

Figure 24 (Part 2 of 5). Device Independent DTF

Field Name	Displacement	Length	Contents
$DFAT4	2A	1	Attribute byte 4 **Bit On** — **Meaning** 0,1 — Rewind at close 0 — Unload at close 1 — Leave at close 2 — Basic access method 3 — Standard label tape 4 — Locate mode 5 — No multivolume output 6 — DTF closed by EOV 7 — Reel opened on this file
$DFAT5	2B	1	Attribute byte 5 (Bits 0-7 are reserved)
$DFHTC	2C	1	Error halt code **Code** — **Meaning** X'10' — No get operation code X'11' — Wrong length record on input X'12' — Wrong length block X'20' — No put operation code X'21' — Wrong length record on output X'40' — Permanent tape write error at close X'41' — Wrong block count X'50' — Permanent tape read error at close X'60' — No file statement X'61' — File not allocated X'62' — Not enough buffer space X'70' — No basic operation code
$DFSEC	2C	(1)	Number of sectors to write (split)
$DFWAA	2D	1	Work area A (disk)
$DFSQU	2D	(1)	Format—1 sequence number (tape)
$DFWAB	2E	1	Work area B (disk)
$DFWAC	2F	1	Work area C (disk)
$DFWAD	30	1	Work area D (disk)
$DFR04	31	1	Reserved
$DFRMA	34	3	Work area (disk)

Figure 24 (Part 3 of 5). Device Independent DTF

Field Name	Displacement	Length	Contents
$DFR06	35	1	Reserved
$DFRMB	38	3	Work area (disk)
$DFND1	39	1	Indicator byte 1 **Bit On** — **Meaning** 0 — IOB not waited on 1 — Truncated block 2 — Empty variable block 3 — Error reading trailer label 4 — Write trailer label 5 — DTF closed by EOV 6 — EOV return via open 7 — No end of file reached
$DFND2	3A	1	Indicator byte 2 **Bit On** — **Meaning** 0 — Device=5444 Disk 1 — Device=5445 Disk 2 — Device=Tape 3 — Device=MFCM 4,5 — Reserved 6 — IOB waited on indicator 7 — Write indicator
$DFND3	3B	1	Indicator byte 3 **Bit On** — **Meaning** 0-3 — Reserved 4 — Short record on unit record device 5-7 — Reserved
$DFR07	3C	1	Reserved
$DFHSA	3C	(1)	Halt routine save area
$DFR08	3D	1	Reserved area 1 byte (5444 only)
$DFSR1	3F	2	EOV register 1 save area
$DFSVD	41	2	EOV chain DTF address save area
$DFNXR	42	1	Disk address of next record

Figure 24 (Part 4 of 5). Device Independent DTF

Field Name	Displacement	Length	Contents
$DFSPC	43	1	Number of tracks in cylinder
$DFSVE	43	(1)	EOV save area
$DFR09	44	1	Reserved
$DFIOA	47	3	Disk address save area
$DFDAT	49	2	Pointer to current index entry buffer pointer save area (split)
$DFR10	4A	1	Reserved
$DFR11	4B	1	Reserved area minus 1 byte (5444 only)
$DFEOF	4E	3	Disk address of logical end of file
$DFDVI	58	10	Reserved for independent DTF

Figure 24 (Part 5 of 5). Device Independent DTF

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code X'10'
$DFUPS	1	1	External indicators
$DFAT1	2	1	Attribute byte 1 Not used
$DFAT2	3	1	Attribute byte 2 **Bit On — Meaning** 0 — Device independent DTF 1 — DTF has been allocated 2 — Device is system input device 3 — Multiple buffers (if this bit is off, it indicates the first entry to CRT data management) 7 — DTF has been opened
$DFCHA	5	2	Address of the next DTF in backward chain
$DFCHB	7	2	Address of next DTF in forward chain
$DFARR	9	2	Address recall register save area (return address)
$DFXRS	B	2	XR1 save area (contents of calling program register 1)
$DFLRA	D	2	Logical record address
$DFCMP	E	1	Completion code **Code — Meaning** X'40' — Successful completion X'41' — Permanent error X'42' — End of file indicator

Figure 25 (Part 1 of 2). CRT/Keyboard DTF

Field Name	Displacement	Length	Contents
$DFOPC	F	1	Operation code **Bit On** — **Meaning** None — No operation 0 — Input only 0,1 — Output/Input, Put/Get 0,3 — Input on request 0,1,2,3 — Clear zone portion of OPC 1 — Output only 5 — Do not blank buffer before operation 4,5,6,7 — Clear numeric portion of OPC
$DFCLB	11	2	Length of data, output
$DFCSB	13	2	Start position, output
$DFCRA	15	2	Address of data, input
$DFCLI	17	2	Length of data, input
$DFCSI	19	2	Start position of input
$DFCQE	1A	1	Console queue element (CQE)
$DFECB	1A	(1)	Console queue element ECB
$DFQCB	21	7	Console queue element Q-code
$DFREQ	22	1	Console queue element request code
$DFADR	24	2	Address of DTF

Figure 25 (Part 2 of 2). CRT/Keyboard DTF

Field Name	Displacement	Length	Contents
$DFDEV	0	1	Device code:[1] 3741 = X'40'
$DFUPS	1	1	External indicators[1]
$DFAT1	2	1	Attribute byte 1[1] Bit On — Meaning 1 — Consecutive 4 — Input 5 — Output
$DFAT2	3	1	Attribute byte 2[1] Bit On — Meaning 0 — Device independent 1 — Allocated 2 — System input 3 — Multiple buffers 7 — DTF has been opened
$DFCHA	5	2	DTF chain pointer A (backward)
$DFCHB	7	2	DTF chain pointer B (forward)[1]
$DFARR	9	2	ARR save area (return address)
$DFXRS	B	2	XR1 save area (contents of object program XR1)
$DFLRA	D	2	Address of logical record[1]
$DFCMP	E	1	Completion code Code — Meaning X'40' — Normal completion X'41' — Controlled cancel taken on permanent I/O error X'42' — End of file (input)

[1] Indicates field is used for pre-open DTF.

Figure 26 (Part 1 of 2). 3741 DTF

Field Name	Displacement	Length	Contents
			Completion codes other than X'40' are returned before the data management function is actually completed.
$DFOPC	F	1	Operation Code: Bit(s) On — Meaning 0 — Get 1 — Put
$DFIOB	11	2	Pre-open address of I/O area[1] Post-open address of IOB
$DFPRB	13	2	Address of current process IOB
$DFBKL	15	2	Block length (length of data buffer)[1]
$DFRCL	17	2	Logical record length[1]
$DFPTR	19	2	Reserved
$DFR01	1A	1	Reserved
$DFXTA	1C	2	Reserved
$DFMVF	1C	(2)	Reserved
$DFR02	1D	1	Reserved
$DFXTB	1F	2	Reserved
$DFNUM	1F	(2)	Reserved
$DFSWA	20	1	Reserved
$DFNAM	28	8	File name[1]

[1] Indicates field is used for pre-open DTF.

Figure 26 (Part 2 of 2). 3741 DTF

Appendix C: Input/Output Blocks

DISK INPUT/OUTPUT BLOCK

You build the disk IOB by issuing the $IOBD macro instruction. If you use $RDD, $WRTD, or $WAIT in your program, you must use the $IOED macro instruction to assign the offset in the IOB. The format of the IOB and the labels assigned to the fields are shown in Figure 27. IOBs for the 5444 Disk Storage Drive, the 5445 Disk Storage, the 3340 Direct Access Storage Facility, and the 3344 Direct Access Storage are 30 bytes long.

Label	Displacement	Length	Contents
IOB common equates for all devices			
SIBECB	0	1	Wait/post byte — byte 1 of three-byte ECB
SIBCOM	1	1	Completion code — byte 2 of ECB.
SIBCMP	2	1	The third byte of ECB. A one-byte completion code indicating the status of the operation just performed. You should check this byte before assuming that the data transfer has occurred. Before the wait routine is called, each bit in this byte has the following meaning: **Bit On — Meaning** 0 — Seek has been started on the operation requested by the IOB. 1 — The operation requested is complete 2 — Data transfer is pending on this operation. 3 — Data transfer has been started on this operation. 4 — A wait occurs for this IOB 5 — If bit 7 is also on, there is an error on an associated IOB; if bit 7 is off, a scan equal has been found. 6 — The scan is not satisfied. 7 — A permanent error has occurred on this IOB or an associated IOB. After the wait routine has finished, the code in this byte has the following meaning: **Code — Meaning** X'40' — Successful completion X'41' — Permanent I/O error. X'42' — Scan not satisfied. X'44' — Scan equal found. X'45' — Permanent error on an associated IOB.
SIBCHN	6	4	Address of the next IOB in the chain. IOBs are chained only when the file requires more than one IOB. This area is always present, even when chaining is not used. When the operation specified by this IOB is complete, this area contains the disk address last used (cylinder/sector for the 5444; head/record for the 5445).
SIBQB	7	1	The Q-byte of the start I/O(SIO) command. You set this byte through the $IOBD macro instruction.
SIBRB	8	1	The R-byte of the start I/O command. It further defines the operation requested. Figure 28 shows the possible R-byte settings for the SIO command.

Figure 27 (Part 1 of 4). Disk IOB Format

Label	Displacement	Length	Contents
$IBEID	9	1	ERP module displacement ID.
$IBDAT	B	2	Address of the leftmost byte of your data area. You provide this address Through the $IOBD macro instruction.
$IBSNS	D	2	The area used by the input/output supervisor to contain device status sense information. The contents of this area are described in Figure 29.
$IBFLG	E	1	Indicates special handling required for I/O operations through the various bit settings. You set bits 0 and 4 through the $IOBD macro instruction. If no special handling is required this byte must be set to X'00'. The bit settings are: **Bit On** — **Meaning** 0 — No recovery is to be attempted is a data check, missing address mark, no record found, or track condition check error condition occurs. 1 — No verification is to be done on write operations. 2 — No error logging is to be done if any disk I/O error should occur. Control is to be returned to the calling routine. 3 — Disk IOS should not use the C and S bytes in the IOB, but should pick up the F, C, and S bytes at the end of the IOB for use in this operation. This bit should be used only by the system control program. 4 — The calling routine is not using disk data management; therefore, this IOB is not associated with a DTF. 5 — There is no load I/O of the disk file data register (DFDR). 6 — Error logging is in progress. 7 — A data transfer operation involving an alternate track is in progress.
$IBERR	F	1	The area used by disk IOS to count the number of retries required to complete the I/O request.
$IBTCB	11	2	Task control block (TCB) address.
$IBWRK	12	1	IOS partial completion code.

Figure 27 (Part 2 of 4). Disk IOB Format

Label	Displacement	Length	Contents
$IBFL2	13	1	Flag byte for use with the 5445. The meanings of the bits are: **Bit On** — **Meaning** 0 — Special seek 1 — Super IOB 2 — Not used (5704-SC1) 2-3 — Specify operation (5704-SC2) where: X'11' Scan X'10' Write X'01' Read X'00' Seek 3 — Not used (5704-SC1) 4 — Reserved 5 — Not used (5704-SC1) 5 — C/S address must be converted but not Q-byte (5704-SC2) 6 — No seek 7 — Long form IOB
$IBCC	14	1	The hexadecimal value of the cylinder address where the operation is to begin (5445 only). You set this byte through the $IOBD macro instruction.
$IBHH	15	1	The hexadecimal value of the head address where the operation is to begin (5445 only). You set this byte through the $IOBD macro instruction.
$IBCB	15	(1)	The hexadecimal value of the cylinder address where the operation is to begin (5444 only). You set this byte through the $IOBD macro instruction.
$IBR	16	1	The hexadecimal value of the record address where the operation is to begin (5445 only). You set this byte through the $IOBD macro instruction.
$IBSB	16	(1)	The hexadecimal value of the beginning sector address of the operation (5444 only). You set this byte through the $IOBD macro instruction.
$IBN	17	1	The number of records minus one, in hexadecimal, involved in the data transfer (5445). You set this byte through the $IOBD macro instruction.
$IBNB	17	(1)	The number of sectors minus one, in hexadecimal, involved in the data transfer (5444). You set this byte through the $IOBD macro instruction.

Figure 27 (Part 3 of 4). Disk IOB Format

Label	Displacement	Length	Contents
$IBDAD	19	2	Pointer to 5445 ten-byte disk address.
$IBDCH	1B	2	The disk data management chain pointer (5444 and 5445). It contains the address of the second of the two IOBs used for double buffering.
$IBDTF	1D	2	The address of the DTF associated with this IOB

Figure 27 (Part 4 of 4). Disk IOB Format

I/O Operation	SIO R-Byte Settings (Hex)	Interpretation
5444 Settings		
Control	00	Seek
Read	00 01 02 03	Data Identifier Diagnostic Verify
Write	00 01	Data Identifier
Scan	00 01 02	Equal Low or equal High or equal
5445 Settings		
Control	00 01	Seek Recalibrate
Read	00 03	Key-data Verify key-data
Write	00	Key-data
Scan	00 01 02	Key-data equal Key-data low or equal Key-data high or equal

Figure 28. R-Byte Settings

TIMER INPUT/OUTPUT BLOCK

If you use the $SIT, $TOD, or $RIT macro instructions in your program, you must build the timer IOB by issuing the $TIOB macro instruction. The format of the timer IOB varies depending on whether DATE-Y or DATE-N is specified.

Format of Timer IOB When DATE-Y is Specified

Bytes	Contents
1	Reserved
2-7	Time
8-13	Date

Format of Timer IOB When DATE-N is Specified

Bytes	Contents
1	Reserved
2-7	Time
8	Return Code
9-10	Reserved

Device	Byte	Bit On	Indication
5444	0	0	I/O no-op (single only)
		1	Intervention required
		2	Missing address mark
		3	Equipment check
		4	Data check
		5	No record found
		6	Track condition check
		7	Seek check
	1	0	Scan equal
		1	Access arm at cylinder 0
		2	End of cylinder
		3	Seek busy
		4	Hundred cylinders
		5	Device overrun
		6	Status address A*
		7	Status address B*
5445	0	0	Format error
		1	Intervention required
		2	Missing address mark
		3	Equipment check
		4	Data check
		5	No record found
		6	No-op
		7	Overrun
	1	0	Disk drive error
		1	Unsafe
		2	Seek 1 complete
		3	Seek 2 complete
		4	Data operation complete
		5	End of cylinder has been reached
		6	Scan equal
		7	Disk drive ID**

*Indicates which drive on the 5444 had last data transfer:
Bits 6-7 00—Drive 1
01—Drive 2

** Bit 7 0—Drive 1
1—Drive 2

Figure 29. Device Status Sense Information

Appendix D: Macro Instruction Summary Chart

Figure 30 is a summary chart containing all valid macro processor instructions. The macro instructions are listed in alphabetical order. In addition to the name, three items are given for each macro instruction:

- Format of the instruction with all valid operands
- Function of the macro instruction
- Maximum number of statements generated

For more detailed information on any of the macro instructions, see *Chapter 2: Macro Instruction Statements.*

This page intentionally left blank.

Name of Macro Instruction			Function	Maximum Number of Statements Generated
[Name]	$ALOC	[DTF-address]	Assigns the file indicated by the DTF to your program.	8
[Name]	$CHK	[CKL-address]	Tests for I/O operation completion in the check list.	8
[Name]	$CKL	DTF-address[,SKIP-Y/N] [,REQK-Y/N] [,RTN-Y/N] [,LAST-Y/N]	Generates an entry for the check list to be used by the check routine.	3
[Name]	$CLOS	[DTF-address]	Prepares the device for job termination.	8
	$COMN		Generates equates.	None
[Name]	$CQEP	[KEY-number]	Generates a parameter list for a program function key request.	10
[Name]	$CTLT	[DTF-address] [,OPC-code]	Issues control commands to the tape device.	11
[Name]	$DATE	[LABEL-address]	Retrieves system date.	4
[Name]	$DTFC	IOBA-address, NIOB-number [,DEV-code] [,UP-mask] [,CHN-address] [,RCAD-address] [,OPC-code] [,DEFER-Y/N] [,CARDI-Y/N] [,PRINT4-Y/N] [,FEED-Y/N] [,STACKR-number] [,READA-address] [,PUNCHA-address] [,PRINTA-address] [,READL-number] [,PUNCHL-number] [,PRINTL-number] [,PRHEAD-mask] [,ALIGN-Y/N]	Builds a DTF for a card file.	33
[Name]	$DTFD	AC-code,RECL-number,NAME-filename, BLKL-number,IO-address [,DISK-5444/5445/3340] [,UP-mask] [,BUFNO-1/2] [,MVF-N/Y] [,LIM-N/Y] [,ORD-N/Y] [,BIN-N/Y] [,CHN-address] [,RCAD-address] [,ENT-number] [,MVFN-number] [,KEYL-number] [,KEYD-number] [,KEYA-address] [,MVFT-address] [,MSTX-address] [,IBLKL-number] [,ADKEY-address] [,EOVK-address] [,SHR-Y/N] [,EXTBUF-Y/N]	Builds a DTF for a disk file. The SHR-Y/N and EXTBUF-Y/N parameters apply only to Program Number 5704-SC2.	210

Figure 30 (Part 1 of 6). Macro Instruction Summary Chart

Name of Macro Instruction			Function	Maximum Number of Statements Generated
[Name]	$DTFI	NAME-filename,RECL-number,IO-address [,AC-IN/OUT] [,BLKL-number] [,RCAD-address] [,BUFNO-1/2] [,CHN-address] [,UP-mask]	Builds a DTF for a device independent file.	88
[Name]	$DTFK	NAME-filename,RECL-number,IO-address [,AC-I/O] [,RCAD-address] [,BUFNO-1/2] [,CHN-address] [,UP-mask]	Builds a DTF for a 3741 file.	16
[Name]	$DTFO	[DISK-Y/N] [,TAPE-Y/N] [,IND-Y/N] [,MFCU-Y/N] [,MFCM-Y/N] [,D3741-Y/N] [,D2501-Y/N] [,D1442-Y/N] [,D1403-Y/N] [,D3284-Y/N] [,CRT-Y/N] [,ALL-Y/N] [,FIELD-Y/N]	Establishes labels for fields on a DTF.	None
[Name]	$DTFP	RCAD address,IOBA-address,IOAA-address, OVFL-number,PAGE-number [,DEV-code] [,UP-mask] [HUC-Y/N] [,CHN-address] [,PRINT-Y/N] [,SKIPB-number] [,SPACEB-number] [,SKIPA-number] [,SPACEA-number] [,RECL-number]	Builds a DTF for a printer file.	33
[Name]	$DTFS	[PUTDAT-address] [,PUTLOC-number] [,UP-mask] [,CHN-address] [,PUTLEN-number] [,OPC-code] [,GETDAT-address] [,GETLOC-number] [,GETLEN-number] [,BLANK-Y/N]	Builds a DTF for a CRT file.	37
[Name]	$DTFT	NAME-filename,IO-address,BLKL-number, RECL-number [,UP-mask] [,AC-IN/OUT] [,CHN-address] [,BASIC-Y/N] [,RCAD-address] [,MODE-LOCATE/MOVE] [,MBUFF-Y/N] [,RECFM-code] [,LIOA-number] [,SPAN-Y/N] [,CODE-A/E] [,OSET-B/number] [,END-code] [,MVF-Y/N]	Builds a DTF for a tape file.	67
[Name]	$EOJ	[CANCEL-NORMAL/IMMED/CONTRL]	Terminates the job and returns control to the supervisor.	4
[Name]	$FIND	NAME-module [,FIND-address] [,PACK-P/S]	Provides the disk address of a module in the O library.	23
[Name]	$FTCH	NAME-module name[,PACK-P/S]	Finds and loads an O module and passes control to it.	21

Figure 30 (Part 2 of 6). Macro Instruction Summary Chart

Name of Macro Instruction			Function	Maximum Number of Statements Generated
[Name]	$GETC	[DEV-code] [,DTF-address] [,EOF-address] [,ERR-address] [,OPC-Y/N] [,READL-number] [,CARDI-Y/N] [,STACKR-number]	Gets a record from a card file.	31
[Name]	$GETD	{[AC-code] / [EBAC-code]} [,DTF-address] [,ERR-address] [,EOF-address] [,NFR-address] [LSTV-address] [NOKY-address]	Gets a record from a disk file via disk data management. The EBAC-code parameter applies only to Program Number 5704-SC2.	46
[Name]	$GETI	[DTF-address] [,RCAD-address] [,ERR-address] [,EOF-address]	Gets a record from a device independent file.	32
[Name]	$GETK	EOF-address [,DTF-address] [,ERR-address]	Gets a record from a 3741 file.	
[Name]	$GETS	[DTF-address] [,GETDAT-address] [,GETLEN-number] [,GETLOC-number] [,BLANK-Y/N] [,OPC-IN/INR/N] [,EOF-address]	Gets a record from the CRT keyboard.	48
[Name]	$GETT	MODULE-name[,DTF-address] [,RCAD-address] [,OPC-Y/N] [,ERR-address] [,EOF-address]	Gets a record from a tape file.	35
[Name]	$GPC	[DEV-code] [,DTF-address] [,OPC-code] [,DEFER-Y/N] [,CARDI-Y/N] [,PRINT4-Y/N] [,FEED-Y/N] [,STACKR-number] [,READL-number] [,PUNCHL-number] [,PRINTL-number] [,PRHEAD-mask] [,EOF-address] [,ERR-address]	Constructs a get, then a put interface to a card file.	37
[Name]	$IOBD	[DISK-5444/5445/3340] [,CYL-number] [,SCTR-number] [,HEAD-number] [,NUM-number] [,BUFF-address] [,O-number] [,ERREC-IOS/USER] [,LOG-Y/N] [,VER-Y/N] [,CHN-address]	Builds an IOB for a disk file.	40
	$IOED		Establishes labels for fields in the disk IOB.	None
[Name]	$LMSG	[FORMAT-code] [,COMP-code] [,HALT-code] [,SUBH-code] [,SEV-code] [,DEF-code] [,OPN0-Y/N] [,OPN1-Y/N] [,OPN2-Y/N] [,OPN3-Y/N] [TLEN-number] [,TADR-address]	Generates log list for halt message. Linkage to log may also be generated.	12

Figure 30 (Part 3 of 6). Macro Instruction Summary Chart

Name of Macro Instruction			Function	Maximum Number of Statements Generated
[Name]	$LOAD	NAME-module name [,FIND-address] [,LOAD-2/address] [,USE-R/NR] [,PLIST-address] [,PACK-P/S]	Finds and loads or loads a previously found O module into main storage.	41
[Name]	$LOAD	FIND-address [,LOAD-2/address] [,PACK-P/S]		
[Name]	$LOG	[LIST-address] [,OPN0-address] [,OPN1-address] [,OPN2-address]	Generates linkage to system log.	29
	$LOGD		Generates offsets for log lists.	None
[Name]	$LWTO	TLEN-number, TADR-address [,COMP-code] [,HALT-code] [,SUBH-code] [,REPLY-Y/N] [,RLEN-number] [,RADR-address]	Generates parameter list for write to operator or write to operator with reply function.	15
[Name]	$OPEN	[DTF-address]	Prepares a previously allocated file for data transfer.	8
[Name]	$PFKT	[CQE-address] [,WAIT-Y/N] [,LABEL-address]	Tests whether a program function key that was allocated by $PFKY has been pressed.	14
[Name]	$PFKY	[CQE-address] [,ERR-address]	Request allocation of a program function key.	22
[Name]	$PGS	[DTF-address] [,BLANK-Y/N] [,OPC-Y/N] [,PUTDAT-address] [,PUTLEN-number] [,PUTLOC-number] [,GETDAT-address] [,GETLEN-number] [,GETLOC-number] [,EOF-address]	PUT/GET operation through CRT data management.	66
[Name]	$PUTC	[DEV-code] [,DTF-address] [,OPC-code] [,DEFER-Y/N] [,PRINT4-Y/N] [,FEED-Y/N] [,STACKR-number] [,PUNCHL-number] [,PRINTL-number] [,PRHEAD-mask] [,ERR-address]	Puts a record to a card file.	30
[Name]	$PUTD	{ [AC-code] / [EBAC-code] } [,DTF-address] [,ERR-address] [,EOX-address] [,DUP-address] [,SERR-address] [,KERR-address] [,UPD-Y/N] [,LSTV-address] [,NOKY-address] [,HKER-address]	Writes a record on a disk file via disk data management. The *EBAC-code* parameter applies only to Program Number 5704-SC2.	67
[Name]	$PUTI	[,DTF-address] [,RCAD-address] [,ERR-address] [,EOX-address]	Puts a record to a device independent file.	32

Figure 30 (Part 4 of 6). Macro Instruction Summary Chart

Name of Macro Instruction			Function	Maximum Number of Statements Generated
[Name]	$PUTK	[DTF-address] [,ERR-address]	Puts a record to a 3741.	
[Name]	$PUTP	[DEV-code] [,DTF-address] [,PRINT-Y/N] [,SKIPB-number] [,SPACEB-number] [,SKIPA-number] [,SPACEA-number] [,ERR-address] [,OVFL-address]	Prints a record via printer data management.	37
[Name]	$PUTS	[DTF-address] [,BLANK-Y/N] [,OPC-Y/N] [,PUTDAT-address] [,PUTLOC-number] [,PUTLEN-number]	Puts a record to the CRT via data management.	38
[Name]	$PUTT	MODULE-name [,DTF-address] [,RCAD-address] [,OPC-Y/N] [,LENAD-address] [,ERR-address]	Puts a record to a tape file.	33
[Name]	$RDD	IOB-address,CS-address,NSECT-number [,DISK-5444/5445/3340]	Reads a record from a disk file via the input/output supervisor.	19
[Name]	$RDT	[DTF-address] [,RCAD-address] [,DIRECT-FORW/BACK]	Reads a record from a tape file via basic tape data management.	21
[Name]	$READ	[LIST-address] [,OPC-code] [,EOF-address] [,EOJ-address] [,ERR-address]	Generates linkage to system reader.	32
[Name]	$RIT	[IOB-name] [,CANCEL-Y/N]	Returns the amount of time remaining in a time interval.	4
	$RLSD		Generates offsets in system reader parameter list.	None
[Name]	$RLST	BUF1-address,WORK-address[,BUF2-address]	Generates system reader parameter list, buffers, and work area.	7
[Name]	$ROLL	[INDEX-1/2]	Generates linkage to rollout/rollin. This macro applies only to Program Number 5704-SC1.	17
[Name]	$SIT	[IOB-name] [,TYPE-DEC/BIN/TU/TOD] [,ITYPE-REAL/WAIT/TASK]	Sets a time interval.	5
[Name]	$SNAP	ID-hex,START-address,END-address	Prints the specified area of main storage on the system logging device.	10
[Name]	$TIOB	[DATE-Y/N]	Generates a timer IOB.	1

Figure 30 (Part 5 of 6). Macro Instruction Summary Chart

Name of Macro Instruction			Function	Maximum Number of Statements Generated
[Name]	$TOD	[IOB-name] [,REF-Y/N] [,TYPE-DEC/BIN/TU]	Returns the time of day and the system date to the program.	4
[Name]	$TRAN	[TRL-address]	Translates a record using the system translate routine.	8
[Name]	$TRL	TO-address,FROM-address,LEN-number, TRT-address	Builds a parameter list to pass information to the system translate routine.	9
[Name]	$TRTB	[CODE-E/A] [,HEX-hex]	Generates an EBCDIC to ASCII or an ASCII to EBCDIC translate table.	258
[Name]	$WAIT	[IOB-label] [,ERR-address]	Waits for completion of a disk input/output operation.	15
[Name]	$WRTD	IOB-address,CS-address,NSECT-number [,DISK-5444/5445/3340]	Writes a record on a disk file via the input/output supervisor.	19
[Name]	$WRTT	[DTF-address] [,RCAD-address] [,OPC-Y/N]	Writes a record to a tape file via the basic tape data management.	21
[Name]	$WTT	[DTF-address] [,ERR-address] [,EOF-address] [,EOT-address] [,WLRS-address] [,WLRL-address]	Waits for completion of a basic tape data management I/O operation.	43
[Name]	$XCTL	NAME-module name[,LOAD-2/address] [,PACK-P/S]	Finds and loads a module at a specified address and passes control to it.	22

Figure 30 (Part 6 of 6). Macro Instruction Summary Chart

Technical Newsletter

This Newsletter No. GN21-5649
Date 29 September 1978

Base Publication No. GC21-7608-2
File No. S3-36

Previous Newsletters None

IBM System/3 Model 15
System Control Programming Macros
Reference Manual

This technical newsletter, a part of version 03, modification 00 of IBM System/3 Model 15 System Control Program Number 5704-SC2 (and also applicable to version 06, modification 00 of IBM System/3 Model 15 System Control Program Number 5704-SC1), 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:

Cover, edition notice
21, 22
27, 28
37 through 44
57, 58
61, 62
87 through 90
111, 112
112.1, 112.2 (added)
113 through 118

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

- Miscellaneous technical changes

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. GN21-5700
Date 28 March 1980

Base Publication No. GC21-7608-2
File No. S3-36

Previous Newsletters GN21-5649

IBM System/3 Model 15
System Control Programming Macros
Reference Manual

This technical newsletter applies to version 8, modification 0 of the IBM System/3 Model 15 System Control Programming Macros 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:

7, 8
35, 36
45 through 48

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

Miscellaneous technical changes

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.	GN21-5726
Date	26 September 1980
Base Publication No.	GC21-5207-1
File No.	S3-34
Previous Newsletters	None

IBM System/3
Model 15D
System Measurement Facility
Reference and
Logic Manual

This technical newsletter applies to the current version and modification of the applicable System/3 program 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:

1-5, 1-6

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

Change to PERMANENT DISK ERROR message.

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.

IBM System/3 (File No. S3-36) Printed in U.S.A. GC21-7608-2

International Business Machines Corporation
General Systems Division
5775D Glenridge Drive N.E.
Atlanta, Georgia 30301
(USA Only)

General Business Group / International
44 South Broadway
White Plains, New York 10601
U. S. A.

READER'S COMMENT FORM

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

☐ Check if reply is requested.

Note: All comments and suggestions become the property of IBM.

Name _____

Address _____

_____

● No postage necessary if mailed in the U.S.A.

GC21-7608-2

GC21-7608-2

Cut Along Line

Fold

Fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold

Fold

IBM System/3 (File No. S3-36) Printed in U.S.A. GC21-7608-2

IBM

International Business Machines Corporation
General Systems Division
5775D Glenridge Drive N.E.
Atlanta, Georgia 30301
(USA Only)

General Business Group / International
44 South Broadway
White Plains, New York 10601
U. S. A.
(International)

READER'S COMMENT FORM

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

Note: All comments and suggestions become the property of IBM.

☐ Check if reply is requested.

Name _____

Address _____

_____

● No postage necessary if mailed in the U.S.A.

IBM System/3 Model 15
System Control Programming Macros
Reference

GC21-7608-2

GC21-7608-2

Cut Along Line

Fold

Fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold

Fold

IBM System/3 (File No. S3-36) Printed in U.S.A. GC21-7608-2

International Business Machines Corporation
General Systems Division
5775D Glenridge Drive N.E.
Atlanta, Georgia 30301
(USA Only)

General Business Group / International
44 South Broadway
White Plains, New York 10601
U. S. A.
(International)