XMENU

Subroutine Library Reference

Version 2 Release 2
November 1989

Computer Associates™

030510510901

Preface

The VM Systems Group Internal Tools system is a collection of programs that allow interactive creation and manipulation of 327x display data and menus. This reference manual is provided for users of the VM Systems Group Internal Tools system subroutine library.

Menus can be created immediately by first time users, yet VM Systems Group Internal Tools has many features for the sophisticated user.

This manual provides reference material for the VM Systems Group Internal Tools subroutine library.

The first section of this manual gives general information about how to use the subroutine package. This includes calling the subroutines from your program, loading the subroutines, general usage notes, etc.

The second section of this manual gives specific information for each subroutine. The subroutines are ordered by function - the functions are ordered by the normal order of use in an application. There is an alphabetical listing of the VM Systems Group Internal Tools subroutines and macros following the table of contents to make it easy to find a specific subroutine or macro.

Program macros and technical information are also contained in the second section following the subroutine descriptions.

The final section gives a sample application written in each supported programming language.

Contents

Subroutines Grouped by Function

This section lists VM Systems Group Internal Tools subroutines by logical function. Functions are loosely listed in the order one would expect to use them, for example, the routines used to load a menu come before the routines used to display it.

The subroutine reference material in Chapter 2, "Subroutine Descriptions" on page 2-1 is given in alphabetical order.

Using XMENU Subroutines in a Read/Only, Reentrant Mode

MRENT

Using the MENUEXEC saved menu MRENT area

MSAVE

Determining the XMENU Version

MVERS

Defining an application level

MMAKEA MDROPA

Determining the Terminal Characteristics

MCTYPE MSCRSZ MQRSF

Setting Global XMENU Options

MREMOT MSCMSC MPAFLG MIOLOG MSAPLN MSIZEF

Creating Multiple Terminal Sessions

MSMTO MDEFSC

Creating Virtual Screens and Windows

MDEFVS MDEFW

Loading a Menu

MLOAD MLOADX

Loading a Menu into a Window

MLOADW MAW2M MSETW

Loading Programmable Symbol Sets

MPSLD

Setting Menu Options

MFLAGS MALARM MCLEAR MCURSF MMDT MNULLS MPFMAP MRMDTF MSKIP MTRPA1 MUNLKF MUPCAS MSHARE

Setting Window Options

MPROTW MSETWB MSETWF MTBFLG MWCCF

Positioning the Virtual Screen and Windows

MPOSVS MPOSW

Clear field(s) in the Menu and/or Clear the Screen

MCLRF MCLRUP MCLSCR

Dynamically Adding/Deleting Fields

MADD MDEL

Determining Field Characteristics

MNEXTF MFSIZE

Determining Window Characteristics

MQACTW MQRYW

Saving/Restoring Menu data

MDATSV MDATRS

Moving Data to Menu Fields

MD2SCR MLD2S

Retrieving Data from MENUCTRL Files

MGFILE

Justifying Data in Fields/Buffers

MLJUST MRJUST MCENT

Querying/Setting Field Attributes

MFQRY MFSET MQEXTA MSEXTA MQATTR MQEXTP MATRES MSETAT

Using Set Attribute Orders

MADDSA MCHGSA MDELSA

Positioning the Cursor

MPCUR

Setting Non-terminal Events

MEVENT

Waiting for a Multiple Terminal interrupt

MWAIT

Activating a Window

MADDW

Changing a Window's display priority

MBOTW MTOPW

Displaying a Menu

MRSHOW MDSPLY MDSPRD MRDSPR MRDSPC

Retrieving Cursor Information

MCURP MCURPF MCURPW MCURPS

Determining the Interrupt Key Pressed

MKEYP

Retrieving Selector Pen Information

MGETLP MGTLPF

Checking the Contents of Fields/Strings

MCKDEC MCKHEX MCKNAT MCKALP MCKNBL MCKPAT MCKSTR

Moving Data from Menu Fields

MTSTF MSCR2D MLS2D

Creating/Updating MENUCTRL Files

MUFILE

Updating Output Menu with the Last User Input

MUPSCR MUPSCM

Deactivating a Window

MDELW

Simulating User Input

MSETIN

Trapping Asynchronous Input

MSETAI MQRYAI MCLRAI

Printing the Menu

MPRINT MPRNOH

Purging a Menu

MPURGE

Purging Virtual Screen and Window

MPURVS MPURW

Purging Multiple Terminal Sessions

MRELSC

Purging an XMENU session

MEXIT

One Call XMENU routine

MDXSCR

Displaying Array Data

MARRW MARRWR MARR2W MW2ARR

Dynamic Menu Facility Routines

MPGINT MPGPOS MPGD2S MPGS2D MPGCLR MPGSCR MPGSSC

Writing an Updated Menu to DASD

MWRITE

Performing Custom Full Screen I/O to Windows

MWINRB MWINRD MWINRM MWINW MWINWR

Performing Custom Full Screen I/O

MFSCW MFSCWR MFSRD MFSRB MFSRM MFSWSF MFSWSR

Resetting a terminal's characteristics

MRESET

Retrieving error messages based on subroutine return codes

MGETRC

Using the XMENU Trace Table

MTRACE MGETTR

General Purpose Routines

MCP MCMS MDATE MUSER MPLIST

Printing Data

MPRLNE

General Purpose Numeric Conversion

MCVTBH MCVTBX MCVTBD MCVTBS MCVTBI MCVTCF MCVTHB MCVTXB
MCVTDB MCVTIB MCVTFC

Converting the Case of Data

MCVTUP MCVTLO

Converting Date and Time Values

MTOD2V MV2TOD MCPC2V MCPT2V MJUL2V MV2JUL

Special Purpose Menu Services

MPOINT MFPOS

EXEC Service Routines

MEXQRY MEXGET MEXPUT MEXE2S MS2EXE

Simulating the EXEC environment

MEXECI

XMENU Assembler Macros

MENUFLGS MPRTFLGS MENUATT MPAGINIT MPAGFLGS UBUFFLGS

Attribute Definitions

ATTDEF

Chapter 1. VM Systems Group Internal Tools Subroutine Overview

XMENUSUB TXTLIB is the library containing the subroutines provided by XMENU to enable you to use menus with programs written in a supported language. The languages supported are: Assembler, C/370 FORTRAN and FORTRAN/VS, PL/I, COBOL/VS and PASCAL/VS. The subroutines provide the capabilities to:

- Define a virtual screen and windows.
- Define a multiple terminal application.
- Load a menu.
- Create a columnar format menu.
- Change the attributes or extended attributes of fields.
- Add or delete fields.
- Add, change, or delete set attribute orders.
- Load and use programmed symbol sets.
- Move data to and from of the menu.
- Move data from and to MENUCTRL files.
- Determine which key was pressed, the cursor position, the string the cursor was on, etc.
- Determine what data was input by the user.
- Determine that fields match certain criteria.
- Create a hardcopy snapshot of a menu.
- Display 3270 data streams.
- Display 3270 data streams in windows.
- Get and/or set EXEC, EXEC 2 or REXX variable data.
- Perform useful numeric conversions.
- Additional convenient utility functions, such as providing the date, USERID, etc.

A brief overview of the XMENU subroutines describing how they are to be used is contained in the XMENU User's Guide, number 221-01-2.0.

In the following section of this book each subroutine is listed along with the format of the call and a description of what it does. A list of each subroutine's parameters, their data types, and possible return codes is also provided for each subroutine. All subroutines are contained in the XMENU subroutine library XMENUSHR TXTLIB.

The amount of virtual storage required to use the XMENU subroutines is based on the number of subroutines your applications calls. Normally this storage will be between 24 to 64 kilobytes.

Using XMENU Subroutines

The following facts should be considered when using XMENU subroutines:

- When the word buffer is used in the subroutine descriptions, it refers to the storage space taken up by a variable in your program.
- All numeric parameters are binary fullwords. Some languages default binary values to halfwords. If halfword values, packed or unpacked decimal numbers are passed, unpredictable results will occur.
- All character parameters are packed character (one character to a byte). If calling from PL/I, be sure to code OPTIONS (ASM INTER) in your subroutine declaration as XMENU subroutines do not support PL/I variable length character strings without this parameter.

  If running under C/370, be sure to declare each XMENU subroutine using both a pragma statement to signal the C/370 compiler that XMENU subroutines follow OS subroutine calling conventions, and a C subroutine declaration statement, to identify the subroutine as an external reference.

  If running under PASCAL/VS code FORTRAN as the procedure directive. XMENU subroutines do not support varying length fields. XMENU subroutines also do not directly support languages that transfer a single variable as two passed parameters - address and length. Some FORTRAN/VS and PASCAL/VS parameters are passed this way. If using FORTRAN/VS Versions 1 or 2, FORTRAN/VS language level 77 CHARACTER data type variables may only be used if each XMENU subroutine to be called by the FORTRAN/VS program is listed in the SC() option of a @PROCESS record. This is not required in FORTRAN/VS Version 3 and above.
- Most subroutine parameters are designed so that the first parameter is the menu number first returned by MLOAD, and the second parameter is a return code. The return codes are listed in a XMENU subroutines CMS HELP file, on page 1-12 of this manual and in each subroutine description in the following section.
- XMENU subroutines check that the minimum necessary number of parameters for each subroutine were passed. Be sure when calling XMENU subroutines from Assembler to use the VL parameter of the CALL macro or set the high-order bit in the last passed address.
- Many of the subroutines work slightly differently or return different data depending on the number of parameters passed. If you are coding in a language that requires that subroutine calls have a fixed number of parameters, define the subroutine be called with the maximum number of parameters you need for your particular application. Most routines with optional numbers of parameters can be used by passing default values in the unused parameters. Some XMENU subroutines operate differently if a numeric value or a character string is passed in a parameter. This is done by checking the first bit of the passed parameter. If it is non-zero, it is assumed to be a character.

VM Systems Group Internal Tools Subroutine Relationships

The illustration on the following page gives a pictorial representation of the order in which VM Systems Group Internal Tools subroutines are normally used by an application. Of course this is not the only way VM Systems Group Internal Tools subroutines can be used. A more complete description is given below, with the appropriate subroutines in parenthesis.

- If writing a read/only program, a 4096 byte work area is obtained by the application program and filled by VM Systems Group Internal Tools (MRENT).
- The program insures that it is running on a 327x type terminal (MCTYPE) and that the terminal size is big enough for the screens used by the application (MSCRSZ).
- The screen is cleared in preparation for entry into full screen mode (MCLSCR).
- Global flags are set (MREMOT, MPAFLG).
- Menu(s) are loaded (MLOAD) and if necessary programmed symbol sets are loaded (MPSLD).
- Menu related flags are set (MFLAGS or MALARM, MTRPA1, MPFMAP, etc.).
- If necessary, data is loaded from MENUCTRL files (MGFILE).
- Data is moved from program buffers to menu fields (MD2SCR, MLD2S).
- The cursor is repositioned if necessary (MPCUR).
- The menu is displayed, and optionally input data is received from the user (MDSPLY, MDSPRD).
- If data was input, terminal specific information is retrieved from VM Systems Group Internal Tools, for example, the cursor position (MCURP), the data the cursor pointed to (MCURPF, MCURPW) etc.
- If requested, the menu is printed (MPRINT, MPRNOH).
- Input data is moved to program buffers (MTSTF, MSCR2D, MLS2D).
- Input data is verified if necessary (user routines, MCKxxx).
- If input errors occurred, input data can be moved to the menu (MUPSCR, MUPSCM), an error message can be output (MD2SCR), the alarm can be sounded (MALARM), and the menu redisplayed.
- If no input error occurred, application program processing of user input can begin.
- If necessary updated data can be written to MENUCTRL files (MUFILE).
- New input can be prompted for, etc.
- At program exit, the menu(s) are freed (MPURGE, MEXIT).

VM Systems Group Internal Tools Data Movement Map

The illustration on the following page gives a pictorial representation of how data is moved from your application program to and from VM Systems Group Internal Tools menus and buffers.

In general, there are eight data areas:

1. Your application program buffers.
2. The file system.
3. The menu to be output to the screen.
4. A buffer containing the data input by the user at the last display of the menu.
5. Data physically on the terminal.
6. On a virtual printer (or a print image in a DASD file).
7. VM Systems Group Internal Tools control block(s) for a specific menu.
8. VM Systems Group Internal Tools control block(s) for all menus.

This diagram shows which VM Systems Group Internal Tools subroutines move data from one area to another. It is important to understand that data input by a user is not automatically placed into the output image of a menu.

XMENU Subroutines Data Types

The following table lists the various XMENU data types and their declarations in the various supported programming languages.

Language	Fullword	Char length "n"	Byte	Pointer
Assembler	F' '	CLn' '	X' '	A( )
COBOL/VS	PICTURE 9(8) COMPUTATIONAL	PICTURE X(n)	n/a	n/a
C/370	long int	char[n]	char	*type
FORTRAN/VS	INTEGER*4	CHARACTER*n	LOGICAL*1	n/a
PASCAL/VS	INTEGER	PACKED ARRAY [1..n] of CHAR	BOOLEAN	@type
PL/I	FIXED BINARY(31)	CHARACTER(n)	BIT(8) ALIGNED	POINTER

XMENU Subroutine Calling Sequences

The following are example calling sequences for the various supported programming languages.

Assembler

```
         LA     R1,PARMLIST          Get parm list in R1
         L      R15,=V(subname)      Get routine address in R15
         BALR   R14,R15              Go to it
*                          Non-zero condition code if error
         BNZ    ERROR                If so take care of it
            ...
PARMLIST DC     A(PARM1)             Parameter list
         DC     A(PARM2)
            ...
         DC     XL1'80',AL3(PARMn) Last Parm
```

Another way if using the OS CALL MACRO (contained in OSMACRO MACLIB) is to replace the first three lines above by the following statement:

```
         CALL   subname,(PARM1,PARM2...PARMn),VL
*                          Non-zero condition code if error
         BNZ    ERROR                If so take care of it
            ...
```

COBOL/VS

```
   CALL "subname" USING PARM1 PARM2 ... PARMn.
```

C/370

```
   subname(parm1,parm2,...parmn)
```

Because VM Systems Group Internal Tools subroutines are called by reference rather than by value, most subroutine parameters must be prefixed by the "&" parameter denoting the address of the parameter. For example, to call MLOAD, one might code:
`MLOAD(&number,&retcode,'MENUNAME')` Note that character strings and arrays by their nature are passed by address and thus do not need the "&" prefix operator.

A declaration list of subroutines is provided in the library XMENU MACLIB. The member is called XMEC370.

FORTRAN/VS (or FORTRAN H, G, etc.)

```
CALL subname(PARM1,PARM2,...,PARMn)
```

Under Versions 1 and 2 of FORTRAN/VS character constants are passed internally as two parameters. This form of parameter passing is not supported by XMENU. To support calls to XMENU subroutines under FORTRAN/VS versions 1 and 2, you should provide @PROCESS statements for XMENU subroutines that pass character data as parameters. A complete list of @PROCESS statements for the XMENU subroutines is provided in XMENU MACLIB as member XMENUFVS. These statements can be included in your program via a FORTRAN INCLUDE statement.

A sample @PROCESS statement is shown below:

```
@PROCESS SC(MLOAD,MDSPRD,MD2SCR...)
```

These statements are not necessary in FORTRAN/VS Version 3 and above.

PASCAL/VS

```
subname(PARM1,PARM2,...,PARMn);
```

All XMENU subroutines used in a PASCAL program must be declared. The library XMENU MACLIB contains a member which declares all of the VM Systems Group Internal Tools subroutines to PASCAL. This member is called XMENUPAS.

PL/I

```
CALL subname(PARM1,PARM2,...,PARMn);
```

Declaring Subroutines

In C/370, PASCAL/VS and PL/I each XMENU subroutine must be previously declared.

C/370 Subroutine Declarations

XMENU MACLIB contains all of the XMENU subroutines previously defined in member XMEC370. An example of a proper declaration follows:

```
#pragma linkage(mload,os)
void mload();
```

It is also possible to provide a skeleton set of parameters to allow the compiler to perform type checking of subroutine parameters.

PASCAL/VS Subroutine Declarations

XMENU MACLIB contains all of the XMENU subroutines previously defined in member XMENUPAS. An example of a proper declaration follows:

```
PROCEDURE mload (VAR   number  : INTEGER;
                 VAR   retcode : INTEGER;
                 VAR   menuname: ALFA);  FORTRAN;
```

PL/I Subroutine Declarations

XMENU MACLIB contains all of the XMENU subroutines previously DECLARED in member XMENUPLI. An example of a proper declaration follows:

```
DECLARE
    MLOAD  EXTERNAL ENTRY (BINARY FIXED(31),
                           BINARY FIXED(31),
                           CHAR(8))     OPTIONS(ASM INTER);
```

Virtual Storage Size of an XMENU menu

The virtual storage size of the first record of any menu is:

(number of fields * 24) + 40

The virtual storage size of the second record of a menu created without the XMENU program "EXT" option is:

(menu-line-size * menu-column-size) + (number of fields) + 8

The virtual storage size of the second record of a menu created with the XMENU program "EXT" option is:

(menu-line-size * menu-column-size) + (number of fields * 9) + 8

The maximum possible DASD record length of a menu is 65,535 although the usual size is between 1,600 and 5,000 bytes. XMENU menus are kept in variable format files.

Loading your program and XMENU subroutines

The following statements show loading your program and XMENU subroutines under CMS:

```
GLOBAL TXTLIB XMENUSHR (plus other run-time libraries needed)
LOAD your-program-name
START * your-program-parameters
```

Note that some languages such as C/370 and PASCAL/VS may provide EXECs for loading, starting, and creating modules from your application programs.

See the CMS Command Reference (SC19-6209 for SP/HPO or SC23-0354 for XA) for more information on the CMS GLOBAL, LOAD and START commands.

XMENU Subroutine return codes

CODE	REASON
0	Subroutine completed successfully.
1	Your terminal is not a 327x.
2	MLOAD, MLOADX - The menu file you are attempting to load is not formatted properly.
2	MQEXTP - The attribute data passed is improperly formatted.
2	MFQRY, MFSET, MQEXTA, MSEXTA The field attribute could not be found.
3	The menu position requested is invalid (greater than the screen size).
4	The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).
5	Data transfer - the field size is greater than the buffer size.
6	Data transfer - the buffer size is greater than the field size.
7	Insufficient parameters were passed to the called subroutine.
8	Insufficient storage available to fulfill the request.
9	No user input yet (on calls for input information).
10	No user input made to the requested field (MTSTF, MLS2D).
11	The menu is too large for your terminal's screen size.
12	MLOAD, MLOADX - The menu file you are attempting to load is not formatted properly.
13	The last user input contained no cursor position (a PA or the CLEAR key was pressed).
14	The field name specified was not found in the menu.
15	Attributes cannot be set on a menu without any fields.
16	An attribute parameter passed was invalid.
17	The requested numeric conversion failed.
18	The offset value (MSCR2D, MD2SCR) exceeds the fields size.
19	The field the cursor is on has no field name (MCURPF).
20	MQEXTP - The attribute data passed in improperly formatted.
21	The cursor is not on a non-blank word (MCURPW).
22	The cursor is not within a non-blank string (MCURPS).
23	A field name specified in MGFILE or MUFILE is invalid or does not exist on the menu.
25	The MENUCTRL file referenced in MGFILE or MUFILE is improperly formatted.
26	A badly formatted data record was found in the MENUCTRL file.
27	No updates have been applied to the MENUCTRL file (MUFILE).
28	The menu file requested does not exist (MLOAD, MLOADX).
29	The MENUCTRL file requested does not exist (MGFILE).
30	The CMS XMENULIB is improperly formatted.
31	The CMS XMENULIB menu member does not exist.
32	Invalid MM/DD/YY passed to MV2TOD or MV2JUL, invalid YY.DDD to MJUL2V.

33 Invalid HH:MM:SS passed to MV2TOD.

35 The menu does not contain extended attribute data.

37 The programmable symbol set number passed is invalid.

38 The field name to be added already exists in the menu.

39 The offset into the menu of the field to be added exceeds the size of the menu.

40 A field already starts at the offset into the menu passed to MADD.

41 The name of the field to be added contains invalid characters.

42 XMENU encountered a strange control block error - notify program support.

43 Field "zero" cannot be deleted from the menu.

44 Your program is not running under an EXEC (MEXGET, MEXPUT).

45 The EXEC variable name does not exist (MEXGET).

46 There is insufficient storage available to create the EXEC variable.

47 The EXEC variable name or data length passed is invalid.

48 The menu to be redisplayed is not the one last displayed.

49 The last user interrupt was not a selector pen attention.

50 A selector pen attention interrupt occurred but no attention field was returned by the hardware.

51 The MUFILE temporary work file already exists (FILETYPE "CMSUT1").

52 An invalid set attribute value was passed.

53 The start attribute to be deleted or changed does not exist in the menu (must be added with MADDSA first).

54 (no longer exists).

55 (no longer exists).

56 A field or buffer character check failed (MCKALP, MCKDEC, MCKHEX, MCKNAT).

57 The MCKPAT pattern size is not equal to the field or buffer size, or the passed pattern size was zero.

58 No strings were passed to MCKSTR.

59 None of the strings passed to MCKSTR matched the field or buffer.

60 Unable to load the PSLOAD file (not found, I/O error, etc.).

61 The PSLOAD file is improperly formatted.

62 Titles + command + bottom line count is greater than the number of lines on the screen.

63 Column lengths exceed terminal linesize.

64 First column number greater than "ZZ" and row greater than 5 digits (4 digits if column greater than "Z").

65 The menu number passed is not a page mode menu.

66 A specific row/column was specified with BUFENT greater than 1.

67 BUFCNT is less than or equal to zero.

68 Fewer parameters were passed than BUFCNT implies.

69 Row/column numbers passed are not within the menu.

70 No terminal query data available (MQRSF).

71 User input already exists for this field. No overlay was done.

72 User input already exists for this field. Previous input is overlaid.

73 Line number in a line mode menu exceeds the number of lines saved.

74 MADD or MDEL cannot be used on a page mode menu or a menu created for DMS/CMS compatibility.

75 MPRLNE - the line cannot be printed (too long, etc.).

76 MQRYAI - user data is pending at the terminal.

77 MPGS2D cannot return data from a line mode menu.

78 An invalid attribute was passed to MPGINT or MPGD2S.

79 A title/bottom line passed to MPGINT containing multiple fields exceeds the screen's column size.

80 The count of fields in a title/bottom line is not the same as the actual line's field count.

81 The buffer passed to MRENT is not on a doubleword boundary.

82 MWRITE - a menu with the same name already exists as a DASD file.

83 MCVTFC, MCVTCF Character string passed is not valid floating point format.

84 MCVTFC, MCVTCF Bad type specified.

85 MCVTFC, MCVTCF Error in MULTHEX4 routine.

86 MCVTFC, MCVTCF Error in DEC8DEC4 routine.

87 MCVTFC, MCVTCF Characteristic outside range

88 MCVTFC, MCVTCF Error in HEX42DEC routine.

89 MEXE2S, MS2EXE EXECCOMM not supported.

90 ":R" area not available to save MRENT data.

91 No more field names found or offset number is invalid.

92 No saved data to restore.

93 MENUSFE block passed is not formatted properly

94 Logical path name passed does not exist.

95 Logical device number passed does not exist.

96 Subroutine only supported under XMENU 2 MTF.

97 The device address passed MDEFSC does not exist.

98 The device address passed MDEFSC is an invalid device type.

99 The referenced device is disconnected.

100 The referenced device is not attached/offline.

101 MPSLD could not load a symbol set (no R/W sets available).

102 MPSLD could not load a symbol set (no color sets available).

103 Zero/non-positive buffer length passed to a subroutine.

104 Retcode passed to MGETRC doesn't have a message.

1xx Error in CMS FSSTATE (xx is the FSSTATE code).

120 Invalid character in the file name.

124 Invalid CMS FILEMODE.

136 The disk is not accessed.

4xx Error in CMS FSREAD (xx is the FSREAD code).

402 Invalid input buffer address.

403 Permanent I/O error occurred.

408 Incorrect length record read.

412 End of file detected.

425 Insufficient free storage.

5xx Error in CMS FSWRITE (xx is the FSWRITE code).

502 Invalid output buffer address.

503 Permanent I/O error occurred.

508 Attempt to write an incorrect length record.

512 Attempt to write on a read only disk.

513 The disk is full.

522 Virtual storage capacity exceeded.

525 Insufficient free storage.

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

700 Insufficient storage to get virtual screen control block(s).

701 Insufficient storage to get window control block(s).

702 The virtual screen already exists.

703 A window with this name already exists.

704 A window with this name does not exist.

705 The virtual screen does not exist.

706 A vertical size was passed without a horizontal size.

707 The vscreen sizes are invalid or too big (VSIZE*HSIZE > 2**32).

708 Insufficient storage to get virtual screen data array(s).

709 Window vertical size > virtual screen vertical size.

710 Window horizontal size > virtual screen horizontal size.

711 Insufficient storage to get window data array(s).

712 The vertical position is too large.

713 The horizontal position is too large.

714 A window border definition contains invalid data.

715 There was no window active at the last user input.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

718 Negitive number(s) passed to MPOSW or MPOSVS.

719 The window size must be equal to the menu size.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.
754 There are no active (visible) windows to display.
758 Insufficient storage to allocate a window control block (CVTA).
759 Insufficient storage to allocate a window control block (CVTI).
760 An error was detected in an input datastream directed to a window.
762 Insufficient storage to return a datastream (CVTJ).
766 Insufficient storage to return a datastream (CVTB).
768 Insufficient storage to allocate the character array (CVTA).
769 Insufficient storage to allocate the character array (CVTI).
770 Insufficient storage to allocate a window input buffer (WINF).
771 Insufficient storage to allocate terminal character array (WINF).
772 Insufficient storage to allocate terminal attribute array (WINF).
778 Insufficient storage to allocate the attribute array (CVTA).
779 Insufficient storage to allocate the attribute array (CVTI).
788 Insufficient storage to allocate the extnd. attribute array (CVTA).
789 Insufficient storage to allocate the extnd. attribute array (CVTI).
798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).
799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).
1000 CP command input buffer length greater than 8192.
1001 CP command output buffer length greater than 132.
1002 Insufficient storage to get R13 save area.
1003 Bad return code from subroutine loader.
1004 Unable to initialize MRENT area.
1005 Plist > 50 items.
1006 Subroutine version number mismatch.
1007 Subroutine branch table value too large.

Chapter 2. Subroutine Descriptions

This section gives detailed descriptions of each of the subroutines provided by VM Systems Group Internal Tools. The subroutines are divided into sections by general function - loading a menu, displaying a menu, etc. The sections are provided in the order one would use them in developing an application.

See "Declaring Subroutines" on page 1-9 for information about declaring the parameters of each routine and "XMENU Subroutine Calling Sequences" on page 1-8 declaring the routine itself to your programming language. Information on declaring your variable's data types is at "XMENU Subroutines Data Types" on page 1-7.

If there are several calling sequences for a particular subroutine, the additional sequences are listed below the first calling sequence. For example, the MLOAD routine has three possible calling sequences:

- (number, retcode, menuname)
- (number, retcode, menuname, libname)
- (number, retcode, addr-rec1, addr-rec2)

Each routine may be located quickly by consulting the alphabetical list of subroutine names following the Table of Contents.

MADD - Add a new named field to the menu

```
MADD        (number,retcode,fieldname,offset)
```

MADD dynamically adds a new field to an existing menu. This routine is used for special purposes, for example, to intensify parts of a menu such as a phrase for a text editor. While it is possible to add as many new fields as desired, one should normally create most fields with the XMENU editor and only add fields when necessary for a special case.

When a new field is added, it takes on the 327x default hardware attributes (UNPROT, DIM, NOSKIP, NOMDT, default color, highlight and symbol set). Use MFSET and MSEXTA to customize the new field's attribute data.

Any subroutine calls referencing the new field must be made following the MADD call.

MADD calls require menu buffer relocation to insert the new field attributes. If MPOINT or MFPOS calls exist in your program, these routines must be re-called to refresh your copy of the VM Systems Group Internal Tools control block addresses following any MADD calls.

MADD cannot be used to add fields to a menu created by MPGINT or a menu in use by the DMS compatibility subroutines.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD" or "MLOADX" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field to be added. The name must not already exist on the menu. The name must be alphanumeric and its first character must be alphabetic.

offset **(fullword binary)**
The offset within the menu to add the new field. The offset is calculated from the upper left corner of the menu where that corner is position zero. A new field cannot be defined at the same position as an existing old field.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

38 The field name to be added already exists in the menu.

39 The offset into the menu of the field to be added exceeds the size of the menu.

40 A field already starts at the offset into the menu passed to MADD.

41 The name of the field to be added contains invalid characters.

42 XMENU encountered a strange control block error - notify program support.

74 MADD or MDEL cannot be used on a page mode menu or a menu created for DMS/CMS compatibility.

MADDSA - Add Set Attribute order to the menu

```
MADDSA      (number,retcode,value,offset)
            (number,retcode,value,row,column)
            (number,retcode,value,fieldname)
            (number,retcode,value,fieldname,field-offset)
```

MADDSA dynamically adds a set attribute order to the menu. Set attribute characters override the extended attribute data contained in extended field attributes. They can be used to change single or contiguous characters to a different color, highlight or symbol set than the rest of the field or fields. Set attributes of different generic types (color, highlight and/or symbol set) can be placed at the same menu position by multiple MADDSA calls.

Under VM Systems Group Internal Tools, set attribute orders are placed into the menu when the menu is output to the terminal. Since set attribute orders are kept separate from the menu, data may be moved into and out of a field without affecting the set attribute orders. Data retrieved from a field does not contain the set attribute orders.

Even though set attribute orders require a type and value pair, VM Systems Group Internal Tools will derive the type of the attribute from the data supplied in "value" (e.g. "BLUE" is a color, "BLINK" is a highlight, etc.).

Due to the way the set attribute order hardware functions, a set attribute order will override all field definitions from the position of the set attribute order to the bottom of the screen unless another set attribute of the same type resets the set attribute value(s). This will occur even if other fields follow the set attribute character. Therefore you probably want to add set attribute characters in pairs; one to override the field attribute, and another to restore the field attribute (a set attribute with default value(s) does this).

If an existing set attribute character of the same type exists at the requested offset, it is replaced by the new set attribute.

If the terminal does not support set attribute characters, the request is ignored.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

value **(character length 10 or fullword)**
The type/value of the character attribute to be placed. If a color it must be one of "BLUE," "RED," "PINK," "GREEN," "TURQUOISE," "YELLOW," "WHITE" or "DEFCOL" (to reset color to the field's color value). If a highlight attribute it must be one of "NOHILIGHT" (to reset highlighting to the field's highlight value), "BLINK," "REVERSE,"

or "UNDERSCORE." If a logical programmed symbol set number it must be a binary value, located in the rightmost byte of a fullword and must contain zero, X'40' through X'EF' or X'F1' (zero resets the symbol set to the field's default symbol set). To reset all character attributes types to their default field values, type must contain "DEFAULTS."

offset **(fullword binary)**
The offset in the menu at which the set attribute order is added. The offset is calculated from the upper left corner of the menu where that corner is position zero. The set attribute order takes effect beginning with and including the character at the supplied offset.

row **(fullword binary)**
The row in the menu at which the set attribute order is added. The row is calculated from the top row of the menu where that row is row one.

column **(fullword binary)**
The column on the menu at which the set attribute order is added. The column is calculated from the left column of the menu where that column is column one.

fieldname **(character length 7)**
The field within which to add the set attribute order.

field_offset **(fullword binary)**
The offset within the field where the set attribute order is added. The first position within the field (following the field attribute) is position zero.

Return Codes

0 Subroutine completed successfully.

3 The menu position requested is invalid (greater than the screen size).

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

37 The programmable symbol set number passed is invalid.

52 An invalid set attribute value was passed.

MADDW - Activates a window (makes it visible)

```
MADDW          (number,retcode,wname)
```

MADDW is used to make a hidden window visible on the virtual screen.

A window can be hidden by either not specifying the WINACTIV flag when the window was first defined, or by calling the MDELW subroutine.

Windows can be created, loaded, and then hidden until needed, such as windows used for command selection, help, action bars, or pop-ups.

Note that this routine makes a window visible on the virtual screen. If it is positioned on a part of the virtual screen that is not displayed on the terminal, the window will not be seen. It also won't be seen if it is overlaid by a higher priority menu.

This routine flags the window as active. It does not cause the terminal screen to be updated.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

Return Codes

0 The window is made active/visible.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

704 A window with this name does not exist.

705 The virtual screen does not exist.

MALARM - Sound the audible alarm

```
MALARM        (number,retcode,flag)
```

MALARM sets or resets a flag that determines whether the audible alarm is sounded each time the menu is displayed. If the flag is non-zero, the alarm is sounded at each output. If the flag is zero, the alarm is not sounded. If this routine is never called, the default setting for the alarm flag is that set by the XMENU program when the menu was created. Each menu has its own alarm flag.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MARRW - Display Array Data in a Window

```
MARRW     (number,retcode,wname,cpos,dbuff,abuff)
          (number,retcode,wname,cpos,dbuff,abuff,exabuff)
          (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff)
          (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
              exa2buff)
          (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
              exa2buff,exc2buff)
          (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
              exa2buff,exc2buff,dbcbuff)
```

MARRW allows you to write data from application program arrays to an VM Systems Group Internal Tools window. This can be used to display "free form" data, such as engineering designs, or small arrays, such as lists of correctly spelled words in pop-up windows. It is particularly useful for creating and displaying windows that are dynamically created at run time.

There are four subroutines associated with this facility:

MARRW displays the array data on the terminal, then returns without waiting for user input.

MARRWR displays the array data on the terminal, waits for user input, then returns this input into the passed arrays.

MARR2W moves array data to a window without any display. This is used to preload a group of windows before they are displayed.

MW2ARR moves window data to your arrays based on user input during the last display. This routine allows you to fetch input made to multiple windows made in one user input operation.

You must use the MDEFW subroutine to first define your window before using these routines. The size of your arrays is implicitly defined as the size of the window (number of rows multiplied by the number of columns in a row).

The cursor position and the initial Write Control Character are passed in the cpos parameter. This parameter also returns the key pressed in two forms.

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'00'	Issue Write command.
X'80'	Issue Write/Erase command.
X'C0'	Issue Write/Erase alternate.
X'10'	Trap PA1 if pressed at next input.
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

cpos **(character length 24)**
Passes and returns data for the array operation. The following data is kept in cpos:

Bytes	Use
0-3	Passes the current cursor position, returns the cursor position if it has been moved. The upper left corner of the window is position zero.
4	Passes the Write Control Character (WCC) for window output.
5	Returns the Attention Identifier (AID) byte of the key pressed.
6	Passes flags - X'80' means don't position the cursor,
7	Reserved for future use
8-15	Maps the programmable symbol set bits into logical symbol set numbers. Each byte corresponds to the programmable symbol set numbers passed in the exabuff and excbuff arrays, and contains the symbol set number for this symbol set.
16-23	Returns the key pressed, i.e. the character equivalent of the AID byte.

dbuff **(character array)**
Passes and returns character data for the window. The array size must be equal to the window size in bytes. Each position must be either valid displayable data, the code X'1D' which identifies the position of a 3270 attribute, or the code X'08' which identifies a graphics escape character whose second byte is located at the same array offset in the dbcbuff array.

abuff **(character array)**
Passes and returns 3270 attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

In general, the bits in each byte are as follows:

Bits	Use
0-1	Encoded based on the contents of bits 2-7.
2-3	Define a field's protection - 00 is unprotected, 01 is numeric, 02 is protected no skip, 03 is protected, skip.
4-5	Define a field's intensity - 00 is dim, 01 is light pen detectable, 02 is bright, 03 is dark.

6	is unused.
7	is the Modified Data Tag (MDT) bit.

See the 3270 Datastream Programmer's Reference for the actual definitions of the bits in these attributes.

exabuff **(character array)**
Passes and returns 3270 extended attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

The bit definitions of each byte in this array are as follows:

Bits	Use
0-2	Correspond to the programmable symbol sets passed in the cpos parameter. Values 0 through 7 correspond to bytes 0 to 7 in cpos respectively.
3-4	Correspond to the extended highlighting value. 00 is no highlighting, 01 is blink, 02 is reverse video, and 03 is underscore.
5-7	Correspond to the extended color value. 00 through 07 correspond to blue, red, pink, green, turquoise, yellow, and white respectively.

If the terminal does not support extended attributes, the data in this array has no effect.

excbuff **(character array)**
Passes and returns character attribute data for the window. The array size must be equal to the window size in bytes. Each defined byte acts as a field attribute override for the character at that window position.

The bytes in excbuff are in the same format as the bytes in exabuff.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

If the terminal does not support extended attributes, the data in this array has no effect.

exa2buff **(character array)**
Passes and returns secondary 3270 extended attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

The bit definitions of each byte in this array are as follows:

Bits	Use
0-3	Correspond to the field's outlining data.
4	Corresponds to the field's background transparency.
5-7	Correspond to the background color value. 00 through 07 correspond to blue, red, pink, green, turquoise, yellow, and white respectively.

If the terminal does not support these extended attributes, the data in this array has no effect.

exc2buff **(character array)**
Passes and returns secondary character attribute data for the window. The array size must be equal to the window size in bytes. Each defined byte acts as a field attribute override for the character at that window position.

The bytes in exc2buff are in the same format as the bytes in exa2buff.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

If the terminal does not support extended attributes, the data in this array has no effect.

dbcbuff **(character array)**
Passes and returns compound (graphic escape) character data for the window. The array size must be equal to the window size in bytes. Each position must contain valid displayable data.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no compound characters.

If the terminal does not support graphic escape, the data in this array has no effect.

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
99 The referenced device is disconnected.
100 The referenced device is not attached/offline.
103 Zero/non-positive buffer length passed to a subroutine.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.
644 Unit check occurred - sense data returned.
648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MARRWR - Display Array Data in a Window, Get User Input

```
MARRWR      (number,retcode,wname,cpos,dbuff,abuff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
                exa2buff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
                exa2buff,exc2buff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
                exa2buff,exc2buff,dbcbuff)
```

MARRWR allows you to write data from application program arrays to an VM Systems Group Internal Tools window. Following the write, VM Systems Group Internal Tools waits for user input. This input is then used to update the application program arrays.

This subroutine can be used to display "free form" data, such as engineering designs, or small arrays, such as lists of correctly spelled words in pop-up windows. It is particularly useful for creating and displaying windows that are dynamically created at run time.

There are four subroutines associated with this facility:

MARRW displays the array data on the terminal, then returns without waiting for user input.

MARRWR displays the array data on the terminal, waits for user input, then returns this input into the passed arrays.

MARR2W moves array data to a window without any display. This is used to preload a group of windows before they are displayed.

MW2ARR moves window data to your arrays based on user input during the last display. This routine allows you to fetch input made to multiple windows made in one user input operation.

You must use the MDEFW subroutine to first define your window before using these routines. The size of your arrays is implicitly defined as the size of the window (number of rows multiplied by the number of columns in a row).

The cursor position and the initial Write Control Character are passed in the cpos parameter. This parameter also returns the key pressed in two forms.

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'00'	Issue Write command.
X'80'	Issue Write/Erase command.
X'C0'	Issue Write/Erase alternate.
X'10'	Trap PA1 if pressed at next input.
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

cpos **(character length 24)**
Passes and returns data for the array operation. The following data is kept in cpos:

Bytes	Use
0-3	Passes the current cursor position, returns the cursor position if it has been moved. The upper left corner of the window is position zero.
4	Passes the Write Control Character (WCC) for window output.
5	Returns the Attention Identifier (AID) byte of the key pressed.
6	Passes flags - X'80' means don't position the cursor,
7	Reserved for future use
8-15	Maps the programmable symbol set bits into logical symbol set numbers. Each byte corresponds to the programmable symbol set numbers passed in the exabuff and excbuff arrays, and contains the symbol set number for this symbol set.
16-23	Returns the key pressed, i.e. the character equivalent of the AID byte.

dbuff **(character array)**
Passes and returns character data for the window. The array size must be equal to the window size in bytes. Each position must be either valid displayable data, the code X'1D' which identifies the position of a 3270 attribute, or the code X'08' which identifies a graphics escape character whose second byte is located at the same array offset in the dbcbuff array.

abuff **(character array)**
Passes and returns 3270 attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

In general, the bits in each byte are as follows:

Bits	Use
0-1	Encoded based on the contents of bits 2-7.
2-3	Define a field's protection - 00 is unprotected, 01 is numeric, 02 is protected no skip, 03 is protected, skip.
4-5	Define a field's intensity - 00 is dim, 01 is light pen detectable, 02 is bright, 03 is dark.

6	is unused.
7	is the Modified Data Tag (MDT) bit.

See the 3270 Datastream Programmer's Reference for the actual definitions of the bits in these attributes.

exabuff **(character array)**
Passes and returns 3270 extended attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

The bit definitions of each byte in this array are as follows:

Bits	Use
0-2	Correspond to the programmable symbol sets passed in the cpos parameter. Values 0 through 7 correspond to bytes 0 to 7 in cpos respectively.
3-4	Correspond to the extended highlighting value. 00 is no highlighting, 01 is blink, 02 is reverse video, and 03 is underscore.
5-7	Correspond to the extended color value. 00 through 07 correspond to blue, red, pink, green, turquoise, yellow, and white respectively.

If the terminal does not support extended attributes, the data in this array has no effect.

excbuff **(character array)**
Passes and returns character attribute data for the window. The array size must be equal to the window size in bytes. Each defined byte acts as a field attribute override for the character at that window position.

The bytes in excbuff are in the same format as the bytes in exabuff.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

If the terminal does not support extended attributes, the data in this array has no effect.

exa2buff **(character array)**
Passes and returns secondary 3270 extended attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

The bit definitions of each byte in this array are as follows:

Bits	Use
0-3	Correspond to the field's outlining data.
4	Corresponds to the field's background transparency.
5-7	Correspond to the background color value. 00 through 07 correspond to blue, red, pink, green, turquoise, yellow, and white respectively.

If the terminal does not support these extended attributes, the data in this array has no effect.

exc2buff **(character array)**
Passes and returns secondary character attribute data for the window. The array size must be equal to the window size in bytes. Each defined byte acts as a field attribute override for the character at that window position.

The bytes in exc2buff are in the same format as the bytes in exa2buff.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

If the terminal does not support extended attributes, the data in this array has no effect.

dbcbuff **(character array)**
Passes and returns compound (graphic escape) character data for the window. The array size must be equal to the window size in bytes. Each position must contain valid displayable data.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no compound characters.

If the terminal does not support graphic escape, the data in this array has no effect.

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
99 The referenced device is disconnected.
100 The referenced device is not attached/offline.
103 Zero/non-positive buffer length passed to a subroutine.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.
644 Unit check occurred - sense data returned.
648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MARR2W - Move Application Array Data to a Window

```
MARR2W      (number,retcode,wname,cpos,dbuff,abuff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
                exa2buff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
                exa2buff,exc2buff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
                exa2buff,exc2buff,dbcbuff)
```

MARR2W allows you to write data from application program arrays to an VM Systems Group Internal Tools window. The window is not displayed, the data is simply staged for future display.

This subroutine can be used to display "free form" data, such as engineering designs, or small arrays, such as lists of correctly spelled words in pop-up windows. It is particularly useful for creating and displaying windows that are dynamically created at run time.

There are four subroutines associated with this facility:

MARRW displays the array data on the terminal, then returns without waiting for user input.

MARRWR displays the array data on the terminal, waits for user input, then returns this input into the passed arrays.

MARR2W moves array data to a window without any display. This is used to preload a group of windows before they are displayed.

MW2ARR moves window data to your arrays based on user input during the last display. This routine allows you to fetch input made to multiple windows made in one user input operation.

You must use the MDEFW subroutine to first define your window before using these routines. The size of your arrays is implicitly defined as the size of the window (number of rows multiplied by the number of columns in a row).

The cursor position and the initial Write Control Character are passed in the cpos parameter. This parameter also returns the key pressed in two forms.

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'00'	Issue Write command.
X'80'	Issue Write/Erase command.
X'C0'	Issue Write/Erase alternate.
X'10'	Trap PA1 if pressed at next input.
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

cpos **(character length 24)**
Passes and returns data for the array operation. The following data is kept in cpos:

Bytes	Use
0-3	Passes the current cursor position, returns the cursor position if it has been moved. The upper left corner of the window is position zero.
4	Passes the Write Control Character (WCC) for window output.
5	Returns the Attention Identifier (AID) byte of the key pressed.
6	Passes flags - X'80' means don't position the cursor,
7	Reserved for future use
8-15	Maps the programmable symbol set bits into logical symbol set numbers. Each byte corresponds to the programmable symbol set numbers passed in the exabuff and excbuff arrays, and contains the symbol set number for this symbol set.
16-23	Returns the key pressed, i.e. the character equivalent of the AID byte.

dbuff **(character array)**
Passes and returns character data for the window. The array size must be equal to the window size in bytes. Each position must be either valid displayable data, the code X'1D' which identifies the position of a 3270 attribute, or the code X'08' which identifies a graphics escape character whose second byte is located at the same array offset in the dbcbuff array.

abuff **(character array)**
Passes and returns 3270 attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

In general, the bits in each byte are as follows:

Bits	Use
0-1	Encoded based on the contents of bits 2-7.
2-3	Define a field's protection - 00 is unprotected, 01 is numeric, 02 is protected no skip, 03 is protected, skip.
4-5	Define a field's intensity - 00 is dim, 01 is light pen detectable, 02 is bright, 03 is dark.

6	is unused.
7	is the Modified Data Tag (MDT) bit.

See the 3270 Datastream Programmer's Reference for the actual definitions of the bits in these attributes.

exabuff **(character array)**
Passes and returns 3270 extended attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

The bit definitions of each byte in this array are as follows:

Bits	Use
0-2	Correspond to the programmable symbol sets passed in the cpos parameter. Values 0 through 7 correspond to bytes 0 to 7 in cpos respectively.
3-4	Correspond to the extended highlighting value. 00 is no highlighting, 01 is blink, 02 is reverse video, and 03 is underscore.
5-7	Correspond to the extended color value. 00 through 07 correspond to blue, red, pink, green, turquoise, yellow, and white respectively.

If the terminal does not support extended attributes, the data in this array has no effect.

excbuff **(character array)**
Passes and returns character attribute data for the window. The array size must be equal to the window size in bytes. Each defined byte acts as a field attribute override for the character at that window position.

The bytes in excbuff are in the same format as the bytes in exabuff.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

If the terminal does not support extended attributes, the data in this array has no effect.

exa2buff **(character array)**
Passes and returns secondary 3270 extended attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

The bit definitions of each byte in this array are as follows:

Bits	Use
0-3	Correspond to the field's outlining data.
4	Corresponds to the field's background transparency.
5-7	Correspond to the background color value. 00 through 07 correspond to blue, red, pink, green, turquoise, yellow, and white respectively.

If the terminal does not support these extended attributes, the data in this array has no effect.

exc2buff **(character array)**
Passes and returns secondary character attribute data for the window. The array size must be equal to the window size in bytes. Each defined byte acts as a field attribute override for the character at that window position.

The bytes in exc2buff are in the same format as the bytes in exa2buff.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

If the terminal does not support extended attributes, the data in this array has no effect.

dbcbuff **(character array)**
Passes and returns compound (graphic escape) character data for the window. The array size must be equal to the window size in bytes. Each position must contain valid displayable data.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no compound characters.

If the terminal does not support graphic escape, the data in this array has no effect.

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
99 The referenced device is disconnected.
100 The referenced device is not attached/offline.
103 Zero/non-positive buffer length passed to a subroutine.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.
644 Unit check occurred - sense data returned.
648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MATRES - Restore all field attributes to their initial values

```
MATRES        (number,retcode)
```

MATRES restores all menu field attributes to their values when the menu was first loaded. This subroutine can be used to automatically reset all attributes that were changed during the menu's use.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MAW2M - Associates a window with a menu

```
MAW2M          (number,retcode,wname)
```

MAW2M is used to explicitly associate a window with an existing menu.

This routine allows you to begin displaying a menu in a given window.

The window must be exactly the same row and column size as the menu.

Alternately, you could use MLOADW to load a menu and associate it with a window in one step. This routine lets you display multiple menus in a single window, one at a time.

Parameters

number **(two binary fullwords)**
The first fullword contains the number of the menu to be associated with window wname. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

Return Codes

0 The menu is associated with this window.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

704 A window with this name does not exist.

705 The virtual screen does not exist.

719 The window size must be equal to the menu size.

MBOTW - Gives a window the lowest display priority

```
MBOTW          (number,retcode,wname)
```

MBOTW is used to give a window the lowest display priority.

As each window is defined, it implicitly is given the highest display priority, that is, it is placed on the virtual screen after all older windows. This can cause lower priority windows to be partially or completely overlaid by higher priority windows.

MBOTW lets you reorder the display priority of windows, so that an newer window can be placed beneath older windows.

When you are displaying an older, hidden window, it is best to call MBOTW and MADDW before use, and MDELW after use.

This routine only orders the window queue. It does not cause the terminal screen to be updated.

MTOPW can be used to make a window highest priority.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

Return Codes

0 The window is given the lowest display priority.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

704 A window with this name does not exist.

705 The virtual screen does not exist.

MCENT - Center a field's data

```
MCENT       (number,retcode,fieldname)
            (number,retcode,fieldname,zero,fill)
            (number,retcode,fieldname,zero,fill,type)
            (number,retcode,buffer,length)
            (number,retcode,buffer,length,fill)
            (number,retcode,buffer,length,fill,type)
```

MCENT centers data within a field or a user's buffer. Normally, all blanks or nulls (binary zeros) before or after the non-blank data in the field or string are truncated. Unused positions are changed to nulls.

If "fill" is specified, this character is used to fill unused positions instead of nulls. If "type" is specified and non-zero, only nulls are truncated.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The field to be centered.

buffer **(character length "length")**
The buffer to be centered. The centered data is returned to this buffer.

length **(fullword)**
The length of the buffer to be centered.

zero **(fullword)**
A filler parameter that must be passed as a fullword of binary zero in order to distinguish the "fieldname" call vs. the "buffer"/"length" call. This is only necessary when you want to pass "fill" and/or "type."

fill **(character length 1)**
The character used to fill positions freed by the shifting of data. If not specified, unused positions are filled with nulls (binary zeros).

type **(fullword)**
A flag used to determine whether to truncate blanks and nulls or only nulls. If the value is non-zero, only nulls are truncated.

Return Codes

0 Data left justified.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

MCHGSA - Change an existing Set Attribute order

```
MCHGSA      (number,retcode,value,offset)
            (number,retcode,value,row,column)
            (number,retcode,value,fieldname)
            (number,retcode,value,fieldname,field-offset)
```

MCHGSA changes the value of an existing set attribute order on the menu. The set attribute order of the same type at the same position must have been previously defined by "MADDSA."

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

value **(character length 10 or fullword)**
The value to which the existing character attribute order is to be changed. If a color is to be changed it must be one of "BLUE," "RED," "PINK," "GREEN," "TURQUOISE," "YELLOW," "WHITE" or "DEFCOL" (to reset color to the field's default value). If a highlight attribute is to be changed it must be one of "NOHILIGHT" (to reset highlight to the field's default value), "BLINK," "REVERSE," or "UNDERSCORE." If a logical programmed symbol set number is to be changed it must be a binary value, located in the rightmost byte of a fullword and must contain zero, X'40' through X'EF' or X'F1' (zero resets the programmed symbol set to the field's default value). A previous type of "DEFAULTS" can only be "changed" to "DEFAULTS" ("DEFAULTS" resets all existing set attribute definitions to the field's attribute values).

offset **(fullword binary)**
The offset within the menu where the set attribute to be changed exists. The offset is calculated from the upper left corner of the menu where that corner is position zero.

row **(fullword binary)**
The row in the menu where the set attribute to be changed exists. The row is calculated from the top row of the menu where that row is row one.

column **(fullword binary)**
The column on the menu where the set attribute to be changed exists. The column is calculated from the left column of the menu where that column is column one.

fieldname **(character length 7)**
The field within which the set attribute to be changed exists.

field_offset **(fullword binary)**
The offset within the field where the set attribute character to be changed exists. The first position within the field (following the field attribute) is position zero.

Return Codes

0 Subroutine completed successfully.

3 The menu position requested is invalid (greater than the screen size).

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

37 The programmable symbol set number passed is invalid.

52 An invalid set attribute value was passed.

53 The start attribute to be deleted or changed does not exist in the menu (must be added with MADDSA first).

MCKALP - Check that a field or string is alphabetic

```
MCKALP      (number,retcode,offset,fieldname)
            (number,retcode,offset,buffer,length)
```

MCKALP verifies that the contents of a field or buffer consist solely of upper and/or lower case letters and/or blanks. Special symbols and numbers are not valid. The field or buffer is blank and null (binary zero) suppressed from both ends before checking begins. If the check fails, "offset" returns the offset within the field or buffer which contains the invalid character (the first position is position zero).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If checking a program buffer the value passed in "number" is ignored. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

offset **(fullword binary)**
The offset into the field or buffer containing the invalid character.

fieldname **(character length 7)**
The name of the field within the menu to be checked.

buffer **(character length "length")**
The buffer to be checked.

length **(fullword binary)**
The length of "buffer" in bytes. If this parameter is specified, a "buffer"/"length" type of call is assumed.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

56 A field or buffer character check failed (MCKALP, MCKDEC, MCKHEX, MCKNAT).

MCKDEC - Check that a field or string is decimal

```
MCKDEC      (number,retcode,offset,fieldname)
            (number,retcode,offset,buffer,length)
```

MCKDEC verifies that the contents of a field or buffer consist solely of the decimal digits zero through nine. The field or buffer is blank and null (binary zero) suppressed from both ends before checking begins. If the check fails, "offset" returns the offset within the field or buffer which contains the invalid character (the first position is position zero).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If checking a program buffer the value passed in "number" is ignored. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

offset **(fullword binary)**
The offset into the field or buffer containing the invalid character.

fieldname **(character length 7)**
The name of the field within the menu to be checked.

buffer **(character length "length")**
The buffer to be checked.

length **(fullword binary)**
The length of "buffer" in bytes. If this parameter is specified, a "buffer"/"length" type of call is assumed.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

56 A field or buffer character check failed (MCKALP, MCKDEC, MCKHEX, MCKNAT).

MCKHEX - Check that a field or string is hexadecimal

```
MCKHEX      (number,retcode,offset,fieldname)
            (number,retcode,offset,buffer,length)
```

MCKHEX verifies that the contents of a field or buffer consist solely of the decimal digits zero through nine and the upper or lower case letters "A" through "F." The field or buffer is blank and null (binary zero) suppressed from both ends before checking begins. If the check fails, "offset" returns the offset within the field or buffer which contains the invalid character (the first position is position zero).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If checking a program buffer the value passed in "number" is ignored. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

offset **(fullword binary)**
The offset into the field or buffer containing the invalid character.

fieldname **(character length 7)**
The name of the field within the menu to be checked.

buffer **(character length "length")**
The buffer to be checked.

length **(fullword binary)**
The length of "buffer" in bytes. If this parameter is specified, a "buffer"/"length" type of call is assumed.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

56 A field or buffer character check failed (MCKALP, MCKDEC, MCKHEX, MCKNAT).

MCKNAT - Check that a field or string is national characters

```
MCKNAT      (number,retcode,offset,fieldname)
            (number,retcode,offset,buffer,length)
```

MCKNAT verifies that the contents of a field or buffer consist solely of upper case letters and/or the symbols "#," "$," and/or "@." The field or buffer is blank and null (binary zero) suppressed from both ends before checking begins. If the check fails, "offset" returns the offset within the field or buffer which contains the invalid character (the first position is position zero).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If checking a program buffer the value passed in "number" is ignored. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

offset **(fullword binary)**
The offset into the field or buffer containing the invalid character.

fieldname **(character length 7)**
The name of the field within the menu to be checked.

buffer **(character length "length")**
The buffer to be checked.

length **(fullword binary)**
The length of "buffer" in bytes. If this parameter is specified, a "buffer"/"length" type of call is assumed.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

56 A field or buffer character check failed (MCKALP, MCKDEC, MCKHEX, MCKNAT).

MCKNBL - Check that a field or string is non-blank

```
MCKNBL      (number,retcode,offset,fieldname)
            (number,retcode,offset,buffer,length)
```

MCKNBL verifies that the contents of a field or buffer contains at least one character that is neither a blank nor a null (binary zero). If the check fails, "offset" is set to zero (the first character in the field or buffer).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If checking a program buffer the value passed in "number" is ignored. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

offset **(fullword binary)**
The offset into the field or buffer containing the invalid character.

fieldname **(character length 7)**
The name of the field within the menu to be checked.

buffer **(character length "length")**
The buffer to be checked.

length **(fullword binary)**
The length of "buffer" in bytes. If this parameter is specified, a "buffer"/"length" type of call is assumed.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

56 A field or buffer character check failed (MCKALP, MCKDEC, MCKHEX, MCKNAT).

MCKPAT - Check that a field or string matches a pattern

```
MCKPAT      (number,retcode,offset,pattern,plen,fieldname)
            (number,retcode,offset,pattern,plen,buffer,length)
```

MCKPAT verifies that the contents of a field or buffer match a supplied pattern. The pattern is similar in concept to high-level language "picture statements." Each character in the pattern is compared against the corresponding character in the field or buffer. If they match, the next character pair is checked, and so on. The field or buffer is blank and null (binary zero) suppressed from both ends before checking begins; the first pattern character is checked against the first non-blank character in the field or buffer. If the check fails, "offset" returns the offset within the field or buffer which contains the invalid character (the first position is position zero).

The control characters in the pattern must be supplied in upper case. The following table defines the pattern characters and their meanings:

Character	Meaning
A	Any alphabetic character allowed in this position.
C	Any character allowed in this position (don't care character).
D	Any decimal character allowed in this position.
N	Any national character allowed in this position.
X	Any hexadecimal character allowed in this position.
Anything else	Must match the contents of this position exactly.

Some examples:

Pattern	Use
DD/DD/DD	Date (MM/DD/YY format).
DDD-DD-DDDD	Social security number.
(DDD) DDD-DDDD	United States/Canadian telephone number.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If checking a program buffer the value passed in "number" is ignored. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

offset **(fullword binary)**
The offset into the field or buffer containing the invalid character.

pattern **(character length "plen")**
The pattern string to be matched.

plen **(fullword binary)**
The length of the pattern string.

fieldname **(character length 7)**
The name of the field within the menu to be checked.

buffer **(character length "length")**
The buffer to be checked.

length **(fullword binary)**
The length of "buffer" in bytes. If this parameter is supplied, a "buffer"/"length" type of call is assumed.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

56 A field or buffer character check failed (MCKALP, MCKDEC, MCKHEX, MCKNAT).

57 The MCKPAT pattern size is not equal to the field or buffer size, or the passed pattern size was zero.

MCKSTR - Check that a field or string matches a set of strings

```
MCKSTR      (number,retcode,matched-string-number,number-of-strings,
               string-list,string-size-list,minimum-size-list,
               fieldname)
            (number,retcode,matched-string-number,number-of-strings,
               string-list,string-size-list,minimum-size-list,
               buffer,length)
```

MCKSTR verifies that the contents of a field or buffer match one of a set of supplied strings. The field is compared against each string supplied. If a match is found, the number of the matched string is returned in "matched-string-number" (the first string in the list is string zero). A string will match if it is:

1. greater than or equal to the minimum-length value for that string, and
2. if every character matches every corresponding character in the string.

Nulls (binary zeros) and blanks are suppressed from both ends of the field or buffer before comparison begins. Upper and lower case can be supplied in the comparison strings; case has significance during checking. If no string matches, a non-zero return code is provided.

This routine is provided to support command-list scanning applications.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If checking a program buffer the value passed in "number" is ignored. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

matched_string_number (fullword binary)
Returns the number of the string that matched the field or buffer.

number_of_strings (fullword binary)
The number of comparison strings in the list.

string_list **(array of fullword pointers)**
A list of the addresses of each comparison string.

string_size_list **(array of fullword values)**
A list of the maximum lengths of each comparison string.

minimum_size_list (array of fullword values)
A list of the minimum lengths of each comparison string.

fieldname **(character length 7)**
The name of the field within the menu to be checked.

buffer **(character length "length")**
The buffer to be checked.

length **(fullword binary)**
The length of "buffer" in bytes. If this parameter is supplied a "buffer"/"length" type call is assumed.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

58 No strings were passed to MCKSTR.

59 None of the strings passed to MCKSTR matched the field or buffer.

MCLEAR - Clear screen before output of this menu

```
MCLEAR        (number,retcode,flag)
```

MCLEAR sets or resets a flag that determines whether the screen is cleared each time the menu is output. MCLEAR or MCLSCR should be called before the first menu output by a program to prevent the CMS "MORE..." condition. It should then only be used if non-full screen output is sent to the screen between full screen output and you do not want the user to see the "MORE..." condition (this may cause the non-full screen data to be cleared before the user gets a chance to read it). If MCLEAR is left set, additional unnecessary screen I/O is performed. This causes additional overhead, especially in remote terminals.

If the MCLEAR flag is non-zero, the screen is cleared before each output. If the flag is zero, the screen is not cleared before each output. If this routine is never called, the default clear flag is off. Each loaded menu has its own clear flag.

We recommend that MCLSCR be used to completely clear the screen. MCLEAR is only retained in VM Systems Group Internal Tools for compatibility with XMENU.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCLRAI - Clear asynchronous terminal interrupt detection

```
MCLRAI        (number,retcode)
```

MCLRAI is called to turn off asynchronous terminal interrupt detection.

You must call "MCLRAI" when you no longer wish to trap asynchronous terminal interrupts. If you exit your application without clearing asynchronous interrupt trapping, unpredictable results can occur.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCLRF - Clear a field in the menu

```
MCLRF       (number,retcode,fieldname)
            (number,retcode,fieldname,fill)
```

MCLRF sets a particular field in the menu to nulls (binary zeros). If "fill" is specified, the field is filled with that fill character.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field to be cleared.

fill **(character length 1)**
An optional character used to fill the field. If not passed, the field is filled with nulls (binary zeros).

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

MCLRUP - Clear unprotected fields in the menu

```
MCLRUP      (number,retcode)
            (number,retcode,fill)
```

MCLRUP sets all unprotected fields in the menu to nulls (binary zeros) or to an optional fill character.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fill **(character length 1)**
An optional fill character to fill unprotected fields.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCLSCR - Clear the screen

```
MCLSCR        (number,retcode)
```

MCLSCR clears the terminal screen. The value supplied in "number" is ignored. This routine can be called to prevent the CP "MORE..." condition on entry to a full screen program or at the transition of non-full screen mode to full screen mode.

Parameters

number **(fullword binary)**
Ignored. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Subroutine completed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.
644 Unit check occurred - sense data returned.
648 Full screen routine called before initialization.
652 Terminal type changed between or during full screen call.

MCMS - Issue CMS command

```
MCMS        (number,retcode,buffer,length)
            (number,retcode,buffer,length,flag)
            (number,retcode,buffer,length,flag,fblock)
```

MCMS issues a CMS command from your program. Any CMS command requested will be executed; therefore it is your responsibility to be sure that the called CMS routine will not overlay your program's storage. Commands are searched for in the same way CMS command level does - EXEC's followed by modules. Abbreviations are supported. No CMS, OS, DOS or VSAM clean-up is performed. Both a tokenized and an extended parameter list are passed to CMS. An optional "FBLOCK" address and register one high order byte value can be passed to this subroutine.

Parameters

number **(fullword binary)**
Ignored.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer containing the command.

length **(fullword binary)**
Contains the length of "buffer."

flag **(fullword binary)**
If passed, contains data used to modify the function of "MCMS." Presently, only the two rightmost (low order) bytes (2 and 3) are examined. The rightmost (low order) byte contains the value to be passed as the high order byte of register one when the CMS SVC 202 is issued. Byte 2 contains MCMS flags. Normally, MCMS respects the values of "SET ABBREV," "SET IMPCP" and "SET IMPEX." The flags in byte 2 can "override" these CMS settings for this one call. If bit X'80' is set, this is equivalent to "SET IMPEX OFF." If bit X'40' is set, this is equivalent to "SET IMPCP OFF." If bit X'20' is set, this is equivalent to "SET ABBREV OFF." Other bits in this byte are reserved and should be set to zero.

fblock **(structure)**
If passed, points to a CMS compatible "FBLOCK."

Return Codes

0 CMS command completed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

xxx Otherwise the CMS error return code.

MCP - Issue CP command

```
MCP          (number,retcode,buffer,length)
             (number,retcode,buffer,length,inbuff,inlen,inres)
```

MCP issues a CP command from your program. If "inbuff," "inlen" and "inres" are specified, any CP responses to your command are moved to "inbuff." Multiple commands and multiple responses are separated by X'15' characters.

Parameters

number **(fullword binary)**
Ignored.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer containing the command.

length **(fullword binary)**
Contains the length of "buffer."

inbuff **(character length "inlen")**
The buffer to receive the command response if any.

inlen **(fullword binary)**
Contains the length of "inbuff." If more data is received than will fit into "inbuff," the data is truncated.

inres **(fullword binary)**
Contains the amount of data moved to "inbuff" if the response fits into the buffer. If the response is too big, "inres" contains the size of the buffer needed to fit all of the response (i.e. if "inres" is greater than "inlen" then the response did not fit entirely into the buffer).

Return Codes

0 CP command completed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
1000 CP command input buffer length greater than 8192.
1001 CP command output buffer length greater than 132.
xxx Otherwise the CP error return code.

MCPC2V - Convert decrementing CPU timer value to printable format

```
MCPC2V        (cpuval,retcode,time,length)
```

MCPC2V converts a CPU timer value decrementing from '7FFFFFFF...'X into a printable value in the form "H...H:MM:SS.tht." The value returned is the delta time between '7FFFFFFFF...'X and the current value. The CPU timer value is assumed to be in standard 370 format. Values less than '05ED0000...'X are assumed to be '05ED0000...'X.

Parameters

cpuval **(doubleword binary)**
Input decrementing CPU timer value.

retcode **(fullword binary)**
Return code for the conversion.

time **(character length 16)**
Output time in "H...H:MM:SS.tht" format. Leading zeros are suppressed. The data is returned left justified in the buffer.

length **(fullword binary)**
Returns the length of data moved to "time."

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCPT2V - Convert incrementing CPU timer value to printable format

```
MCPT2V        (cpuval,retcode,time,length)
```

MCPT2V converts a CPU timer value incrementing from zero into a printable value in the form "H...H:MM:SS.tht." The value returned is the delta time between zero and the current value. The CPU timer value is assumed to be in standard 370 format. Values greater than '7A120000...'X are assumed to be '7A120000...'X.

Parameters

cpuval **(doubleword binary)**
Input incrementing CPU timer value.

retcode **(fullword binary)**
Return code for the conversion.

time **(character length 16)**
Output time in "H...H:MM:SS.tht" format. Leading zeros are suppressed. The data is returned left justified in the buffer.

length **(fullword binary)**
Returns the length of data moved to "time."

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCTYPE - Get terminal characteristics

```
MCTYPE      (address,retcode,type,model)
            (address,retcode,type,model,colors,highlight,
               prog-symbols,symbol-flags)
```

MCTYPE returns data about your terminal's characteristics.

Parameters

address **(fullword binary)**
Returns the virtual device address of your terminal. If using VM Systems Group Internal Tools in a read/only mode, address should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

type **(character length 4)**
Returns the type of terminal. It can be "3277," "3278," or "3279." "3279" is set if the terminal supports seven colors.

model **(character length 2)**
Returns the model number of the terminal.

colors **(fullword binary)**
Returns the number of colors supported (one, four, or seven).

highlight **(fullword binary)**
Returns the number of highlighting types supported (one or three).

prog_symbols **(byte length 8)**
Returns the physical symbol set numbers present in your terminal.

symbol_flags **(byte length 20)**
Returns various information regarding the symbol sets available in the terminal as described below:

Byte	Contents
0	Number of symbol sets present
1	Symbol sets flag (byte 4 of the query reply character sets))
2	Usable area data flag (byte 4 the query reply (usable area))
4,5	Character matrix size
8,9	Size of screen (columns)
10,11	Size of screen (rows)
12-19	Flags for each symbol set (byte 2 of each character set descriptor from the query reply (character sets))

This information is only valid if the terminal supports query reply structured fields. If not, the data returned will be zero.

Return Codes

0 Subroutine completed successfully.

1 Your terminal is not a 327x.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCURP - Return cursor position at last input

```
MCURP        (number,retcode,curoff)
             (number,retcode,curline,curcol)
```

MCURP returns the position of the cursor within the menu at the last menu input.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

curoff **(fullword binary)**
Specifies the cursor position as an absolute offset from the upper left corner of the menu where that corner is position zero.

curline **(fullword binary)**
Specifies the cursor line position as an offset from the top line of the menu where the top line is line one.

curcol **(fullword binary)**
Specifies the cursor column position as an offset from the left side of the menu where the left side is column one.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

13 The last user input contained no cursor position (a PA or the CLEAR key was pressed).

MCURPF - Return the name of the field containing the cursor

```
MCURPF      (number,retcode,fieldname)
            (number,retcode,fieldname,offset)
            (number,retcode,fieldname,offset,value)
```

MCURPF returns the name of the field containing the cursor at the last input for this menu.

Optionally, the position within the field can also be returned. This value is one less than the offset value used in "MPCUR." This is because under "MCURPF" the first data position of the field is returned as position one (the actual attribute character is considered to be within the field and is returned as position zero).

If "value" is specified, it is used as the cursor position rather than the last input cursor position. This type of call is used to return a field's name and offset given an absolute menu position.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
Returns the name of the field the cursor was within after the last input. If the cursor was not in a named field, this parameter is left unchanged. If "value" is passed, "fieldname" returns the name of the field at offset "value" into the menu.

offset **(fullword binary)**
Specifies the offset within the field "fieldname." Note that unlike "MPCUR" the first data position in the field is returned as one.

value **(fullword binary)**
Specifies a value to be used to find the field name and offset. If specified, the actual cursor position is not used in this call.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

13 The last user input contained no cursor position (a PA or the CLEAR key was pressed).

19 The field the cursor is on has no field name (MCURPF).

MCURPS - Return contents of the field containing the cursor

```
MCURPS      (number,retcode,buffer,length,actual-length)
            (number,retcode,buffer,length,actual-length,fill)
```

MCURPS returns the contents of the field containing the cursor at the last input for this menu.

The returned string is blank and null (binary zero) suppressed from both ends. If "fill" is specified any unused portion of "buffer" is filled with the character "fill."

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer into which the string is copied.

length **(fullword binary)**
The length of "buffer."

actual_length **(fullword binary)**
Specifies the amount of data actually moved to "buffer" (not including the "fill" characters).

fill **(character length 1)**
Specifies a character used to fill unused parts of "buffer." If not specified, the default fill character is null (binary zero).

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

13 The last user input contained no cursor position (a PA or the CLEAR key was pressed).

22 The cursor is not within a non-blank string (MCURPS).

MCURPW - Return the non-blank word at the cursor position

```
MCURPW      (number,retcode,buffer,length,actual-length)
            (number,retcode,buffer,length,actual-length,fill)
```

MCURPW returns the non-null and non-blank word at the cursor position (if any) at the last input for this menu.

If "fill" is specified, any unused portion of "buffer" is filled with the character "fill."

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer into which the word is copied.

length **(fullword binary)**
The length of "buffer."

actual_length **(fullword binary)**
Specifies the amount of data actually moved to "buffer" (not including the "fill" characters).

fill **(character length 1)**
Specifies a character used to fill unused parts of "buffer." If not specified, the default fill character is null (binary zero).

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

13 The last user input contained no cursor position (a PA or the CLEAR key was pressed).

21 The cursor is not on a non-blank word (MCURPW).

MCURSF - Set whether the cursor is positioned on terminal output

```
MCURSF        (number,retcode,flag)
```

MCURSF sets or resets a flag that determines whether the cursor is positioned when this menu is displayed. If the flag is zero, the cursor is positioned (the default). If the flag is non-zero the cursor is not positioned.

This routine can be used to write applications that allow input between displays (such as performance monitors). Not positioning the cursor allows a user to keep on typing even if the screen is updating.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCVTBD - Convert binary to a decimal string

```
MCVTBD        (number,retcode,string)
```

MCVTBD converts the fullword value in "number" to a decimal printable value in "string." The absolute value of the number is returned.

Parameters

number **(fullword binary)**
The value to be converted to decimal.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length 8)**
The printable decimal output including leading zeros if any.

Return Codes

0 Conversion performed

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCVTBH - Convert binary to a hexadecimal string

```
MCVTBH        (number,retcode,string)
```

MCVTBH converts the fullword value in "number" to a hexadecimal printable value in "string."

Parameters

number **(fullword binary)**
The value to be converted to hexadecimal.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length 8)**
The printable hexadecimal output including leading zeros if any.

Return Codes

0 Conversion performed

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCVTBI - Convert binary to left justified, zero suppressed signed integer

```
MCVTBI        (number,retcode,string,length)
```

MCVTBI converts the fullword value in "number" to an integer printable value in "string." The returned value is leading zero suppressed and left justified in "string." If negative, a minus sign is prefixed to the value.

Parameters

number **(fullword binary)**
The value to be converted to integer.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length 12)**
The printable integer output, left justified, leading zero suppressed, with minus sign if appropriate.

length **(fullword binary)**
Returns the length of the data moved to "string."

Return Codes

0 Conversion performed

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCVTBS - Convert binary to a zero-suppressed decimal string

```
MCVTBS        (number,retcode,string)
```

MCVTBS converts the fullword value in "number" to a zero-suppressed decimal printable value in "string." The absolute value of the number is returned.

Parameters

number **(fullword binary)**
The value to be converted to decimal.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length 8)**
The printable decimal output with any leading zeroes replaced by blanks.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCVTBX - Convert binary to left justified, zero suppressed hexadecimal

```
MCVTBX        (number,retcode,string,length)
```

MCVTBX converts the fullword value in "number" to a hexadecimal printable value in "string." The returned value is leading zero suppressed and left justified in "string."

Parameters

number **(fullword binary)**
The value to be converted to hexadecimal.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length 12)**
The printable hexadecimal output, left justified, leading zero suppressed.

length **(fullword binary)**
Returns the length of the data moved to "string."

Return Codes

0 Conversion performed

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCVTCF - Convert character floating point to binary

```
MCVTCF        (number,retcode,string,length,type)
```

MCVTCF converts the printable floating point number in "string" to binary.

Parameters

number **(fullword, doubleword or quadword binary)**
The binary value returned. The size returned is based on the type passed (see "type" below).

retcode **(fullword binary)**
The return code for the conversion.

string **(character length "length")**
The floating point input string.

length **(fullword)**
The length of the floating point input string.

type **(fullword)**
The type of floating point number to return. The possible values for "type" are:

Value	Type
1	Single precision
2	Double precision
3	Extended precision

Return Codes

0 Conversion performed.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
84 MCVTFC, MCVTCF Bad type specified.
85 MCVTFC, MCVTCF Error in MULTHEX4 routine.
86 MCVTFC, MCVTCF Error in DEC8DEC4 routine.
87 MCVTFC, MCVTCF Characteristic outside range
88 MCVTFC, MCVTCF Error in HEX42DEC routine.

MCVTDB - Convert a decimal string to binary

```
MCVTDB        (number,retcode,string,length)
```

MCVTDB converts the unsigned, printable decimal digits in "string" to binary.

Parameters

number **(fullword binary)**
The binary value returned.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length "length")**
The decimal input string.

length **(fullword)**
The length of the decimal input string. The length should be between one and ten digits.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

17 The requested numeric conversion failed.

MCVTFC - Convert floating point to character format

```
MCVTFC        (number,retcode,string,length,type)
```

MCVTFC converts the floating point number in "number" to character form. The number is returned in the form "+|-x.yyyyy E+|-zzz" where the number of "y" digits is determined by the precision set by "type."

Parameters

number **(fullword, doubleword or quadword binary)**
The binary floating point value. The size is based on the type passed.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length 16, 25 or 42)**
The floating point output string. This buffer must be large enough for the type of value converted.

length **(fullword)**
Returns the length of the character string.

type **(fullword)**
The type of floating point number to return. The possible values for "type" are:

Value	Type
1	Single precision
2	Double precision
3	Extended precision

Return Codes

0 Conversion performed.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
84 MCVTFC, MCVTCF Bad type specified.
85 MCVTFC, MCVTCF Error in MULTHEX4 routine.
86 MCVTFC, MCVTCF Error in DEC8DEC4 routine.
87 MCVTFC, MCVTCF Characteristic outside range
88 MCVTFC, MCVTCF Error in HEX42DEC routine.

MCVTHB - Convert a hexadecimal string to binary

```
MCVTHB        (number,retcode,string,length)
```

MCVTHB converts printable hexadecimal digits in "string" to binary. Both lower and upper case letters "A" through "F" are accepted.

Parameters

number **(fullword binary)**
The binary value returned.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length "length")**
The hexadecimal input string.

length **(fullword)**
The length of the hexadecimal input string. The length should be between one and eight digits.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

17 The requested numeric conversion failed.

MCVTIB - Convert a signed integer string to binary

```
MCVTIB        (number,retcode,string,length)
```

MCVTIB converts the printable, optionally signed integer digits in "string" to binary.

Parameters

number **(fullword binary)**
The binary value returned.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length "length")**
The signed integer input string. This string can contain leading and trailing blanks/nulls (binary zeros). If no digits are found at all, zero is returned as the value. If more digits than will fit in a fullword result are passed, most significant (leading) data may be truncated.

length **(fullword)**
The length of the decimal input string.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

17 The requested numeric conversion failed.

MCVTLO - Convert a string to lower case

```
MCVTLO        (number,retcode,string,length)
```

MCVTLO converts a string to lower case.

Parameters

number **(fullword binary)**
Ignored.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length "length")**
The character input string. This string itself is converted to lower case and returned character for character to the same buffer.

length **(fullword)**
The length of the character input string.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCVTUP - Convert a string to upper case

```
MCVTUP        (number,retcode,string,length)
```

MCVTUP converts a string to upper case.

Parameters

number **(fullword binary)**
Ignored.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length "length")**
The character input string. This string itself is converted to upper case and returned character for character to the same buffer.

length **(fullword)**
The length of the character input string.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MCVTXB - Convert a hexadecimal string to binary

```
MCVTXB        (number,retcode,string,length)
```

MCVTXB converts the printable, hexadecimal digits in "string" to binary. This call differs from "MCVTHB" in that the string may contain leading and trailing blanks and nulls (binary zeros).

Parameters

number **(fullword binary)**
The binary value returned.

retcode **(fullword binary)**
The return code for the conversion.

string **(character length "length")**
The hexadecimal input string. This string can contain leading and trailing blanks/nulls (binary zeros). If no digits are found at all, return code 17 is set. If more than eight digits are passed, leading digits are truncated.

length **(fullword)**
The length of the hexadecimal input string.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

17 The requested numeric conversion failed.

MDATE - Get current date and time

```
MDATE       (date)
            (date,time)
            (date,time,vtime)
            (date,time,vtime,ttime)
```

MDATE returns current date and time information.

Parameters

date **(character length 8)**
Returns the current date in "MM/DD/YY" format.

time **(character length 8)**
Returns the current time in "HH:MM:SS" format. If a non-zero return code exists, it is placed in the first four bytes of "time."

vtime **(doubleword binary)**
Returns current virtual machine VTIME in microseconds.

ttime **(doubleword binary)**
Returns current virtual machine TTIME in microseconds.

Return Codes

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MDATRS - Restore the contents of a saved menu

```
MDATRS        (number,retcode,cflag)
```

MDATRS restores the current menu contents saved by "MDATSV."

This subroutine, together with "MDATSV" is used to quickly restore a menu to an initial state, without having to purge and reload it from disk or memory.

A menu can be restored many times from one saved copy. A program can save a menu once, make modifications while processing, then restore the menu for the next iteration.

"MDATSV" and "MDATRS" should not be used with "MADD" and "MDEL." If the menu was modified using "MADD" or "MDEL" after being saved, the restore may cause unpredictable results.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

cflag **(fullword binary)**
Determines whether the cursor is restored to the position saved when "MDATSV" was called. If the flag is non-zero, the cursor position is restored. If the flag is zero the cursor position is not restored.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

92 No saved data to restore.

MDATSV - Save the contents of a menu for later restore

```
MDATSV        (number,retcode)
```

MDATSV saves the current menu contents for later restoration by "MDATRS."

This subroutine, together with "MDATRS" is used to quickly reset a menu to an initial state, without having to purge and reload it from disk or memory.

A menu can be restored many times from one saved copy. A program can save a menu once, make modifications while processing, then restore the menu for the next iteration.

"MDATSV" and "MDATRS" should not be used with "MADD" and "MDEL." If the menu was modified using "MADD" or "MDEL" after being saved, the restore may cause unpredictable results.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MDEFSC - Defines a VM Systems Group Internal Tools Multiple Terminal Facility session

```
MDEFSC      (number,retcode,address,status)
            (number,retcode,address,status,pathname)
```

MDEFSC is used to define a VM Systems Group Internal Tools Multiple Terminal Facility session.

A device address is passed, and a logical terminal value is returned in the second fullword of the number parameter.

The return code from the CMS Console Facility OPEN request is returned in the status parameter.

If a specific path name is desired, it can be passed in the pathname parameter. Otherwise, the pathname generated by MSAPLN or by VM Systems Group Internal Tools is used.

This routine simply defines a logical terminal. It does not wait for a user to connect (or dial) to this terminal; this subroutine returns immediately to your application.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. The second fullword is used to return the logical terminal number for this Multiple Terminal Facility logical terminal. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

address **(halfword binary)**
Contains the virtual device address of the logical device to be defined.

status **(fullword binary)**
Returns the return code from the CMS Console Facility OPEN request.

pathname **(character length 16)**
Contains a path name for this logical device. If specified, this path name should be unique.

Return Codes

0 A VM Systems Group Internal Tools MTF logical terminal has been defined.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
96 Subroutine only supported under XMENU 2 MTF.
97 The device address passed MDEFSC does not exist.
98 The device address passed MDEFSC is an invalid device type.

100 The referenced device is not attached/offline.

MDEFVS - Define virtual screen

```
MDEFVS      (number,retcode)
            (number,retcode,vsize,hsize)
```

MDEFVS is used to explicitly define an VM Systems Group Internal Tools virtual screen. If this routine is never called, or if vsize and hsize aren't specified, then the virtual screen size will be identical to the largest size supported by the terminal currently in use.

A virtual screen is the main area used to display VM Systems Group Internal Tools windows.

The virtual screen must be equal or larger in size than any window that you plan to display on it. The virtual screen can be larger, equal to, or smaller than the physical terminal used to display it.

Currently, only one virtual screen can be active on any given terminal.

Note that MDEFVS is implicitly called if a window is defined by MDEFW, MLOADW, or MPGINT before MDEFVS is explicitly called.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

vsize **(fullword binary)**
Specifies the vertical size of the virtual screen, that is the number of rows.

hsize **(fullword binary)**
Specifies the horizontal size of the virtual screen, that is the number of columns per row.

Return Codes

0 The virtual screen is defined.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

700 Insufficient storage to get virtual screen control block(s).

702 The virtual screen already exists.

706 A vertical size was passed without a horizontal size.

708 Insufficient storage to get virtual screen data array(s).

MDEFW - Define window

```
MDEFW          (number,retcode,wname,vsize,hsize,flags)
```

MDEFW is used to define an VM Systems Group Internal Tools window.

A window is basically equivalent to a virtual terminal - it holds one or more full screen displays. Portions or entire windows are displayed in viewports which appear on the virtual screen. A portion or the entire virtual screen appears on a real terminal.

Any number of windows can exist, but each must have a unique name within a given virtual screen.

The created window has vsize rows and hsize columns per row. A window must be smaller or equal in size to the virtual screen, therefore, your virtual screen should be defined to hold the largest window you ever expect to display on it.

Windows must be exactly the column size of menus displayed on them, and must be equal or smaller than the row size of menus displayed on them.

The passed flags are used to determine whether this is an active (displayed) window, whether borders are displayed around its viewport and whether the menu name is used as a title in the top viewport border.

When a window is defined, it implicitly becomes the top (highest display priority) window. The priority of window display can be modified by using MTOPW.

MLOADW implicitly defines a window the size of the loaded menu. MPGINT can also implicitly define a window.

Note that MDEFVS is implicitly called if a window is defined before MDEFVS is explicitly called.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

vsize **(fullword binary)**
Specifies the vertical size of the window, that is the number of rows.

hsize **(fullword binary)**
Specifies the horizontal size of the window, that is the number of columns per row.

flags **(byte)**
Specifies how the window's viewport is displayed. The following bits are currently defined:

WINVBOR X'80' Display vertical borders if possible
WINHBOR X'40' Display horizontal borders if possible
WINTITLE X'20' Display the menu name on the top border if possible
WINACTIV X'01' This viewport is active, that is, it is visible

Return Codes

0 The window is defined.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
700 Insufficient storage to get virtual screen control block(s).
701 Insufficient storage to get window control block(s).
703 A window with this name already exists.
706 A vertical size was passed without a horizontal size.
708 Insufficient storage to get virtual screen data array(s).
709 Window vertical size > virtual screen vertical size.
710 Window horizontal size > virtual screen horizontal size.
711 Insufficient storage to get window data array(s).

MDEL - Delete a field from the menu

```
MDEL        (number,retcode,fieldname)
            (number,retcode,fieldname,fill)
```

MDEL dynamically deletes an existing field from a menu. This routine is used for special purposes, for example, to remove temporary fields created by MADD. Both original fields and fields added by MADD can be deleted by MDEL. Once a field is deleted, all trace of it is gone.

Any subroutine calls referencing the deleted field must be made preceding the MDEL call.

The old field's attribute character is replaced by either a null or by the character specified by "fill."

Data returned by MFPOS prior to the MDEL call may be invalidated. Refresh MFPOS data after a MDEL call.

MDEL cannot be used to delete fields from a menu created by MPGINT or a menu in use by the DMS compatibility subroutines.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD" or "MLOADX" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field to be deleted.

fill **(character length 1)**
The character used to replace the previous field's attribute character. If not specified, the attribute is replaced by a null (binary zero).

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

43 Field "zero" cannot be deleted from the menu.

74 MADD or MDEL cannot be used on a page mode menu or a menu created for DMS/CMS compatibility.

MDELSA - Delete Set Attribute from the menu

```
MDELSA      (number,retcode)
            (number,retcode,value,offset)
            (number,retcode,value,row,column)
            (number,retcode,value,fieldname)
            (number,retcode,value,fieldname,field-offset)
```

MDELSA removes a set attribute order from the menu. The set attribute character must have been previously defined by "MADDSA."

Calling MDELSA with only the number and retcode will remove all of the attributes added with "MADDSA."

To avoid confusion, all set attributes placed at the same time for the same reason should be deleted at the same time (i.e. the pair(s) of set attribute orders "around" a string).

If the terminal does not support set attribute orders, this call is ignored.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

value **(character length 10 or fullword)**
The type of character attribute to be removed. If a color it must be one of "BLUE," "RED," "PINK," "GREEN," "TURQUOISE," "YELLOW," "WHITE" or "DEFCOL." If a highlight attribute it must be one of "NOHILIGHT," "BLINK," "REVERSE," or "UNDERSCORE." If a logical programmed symbol set number it must be a binary value, located in the rightmost byte of a fullword and must contain zero, X'40' through X'EF' or X'F1.' To remove a character attribute that sets all attribute types to their field default values, value must contain "DEFAULTS."

offset **(fullword binary)**
The offset within the menu where the set attribute order exists. The offset is calculated from the upper left corner of the menu where that corner is position zero.

row **(fullword binary)**
The row in the menu where the set attribute order exists. The row is calculated from the top row of the menu where that row is row one.

column **(fullword binary)**
The column in the menu to delete the set attribute. The column is calculated from the left column of the menu where that column is column one.

fieldname **(character length 7)**
The field where the set attribute order exists.

field_offset **(fullword binary)**
The offset within the field where the set attribute order exists. The first position within the field (following the field attribute) is position zero.

Return Codes

0 Subroutine completed successfully.

3 The menu position requested is invalid (greater than the screen size).

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

37 The programmable symbol set number passed is invalid.

52 An invalid set attribute value was passed.

53 The start attribute to be deleted or changed does not exist in the menu (must be added with MADDSA first).

MDELW - Deactivates a window (makes it invisible)

```
MDELW        (number,retcode,wname)
```

MDELW is used to make a displayed window invisible on the virtual screen.

A window can be displayed by calling the MADDW subroutine.

Windows can be created, loaded, and then hidden until needed again, such as windows used for command selection, help, action bars, or pop-ups.

This routine flags the window as inactive. It does not cause the terminal screen to be updated.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

Return Codes

0 The window is made active/visible.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

704 A window with this name does not exist.

705 The virtual screen does not exist.

MDROPA - Remove a new application level

```
MDROPA      (number,retcode,level)
            (number,retcode,level,brkkey)
```

MDROPA removes an application level identifier, and optionally sets the CP TERM BRKKEY.

This call should only be necessary when two different applications share a copy of the VM Systems Group Internal Tools subroutine package and data area passed by "MRENT."

This routine should be called by the routine receiving control after having called a second application. If "MEXIT" is called prior to this call, all menus with an application identifier higher than the current application level are purged.

A menu can be shared between application levels by calling "MSHARE."

If the brkkey parameter is passed, the passed value is used to set the CP TERM BRKKEY. MMAKEA can be used to save and later restore (using MDROPA) the setting of the BRKKEY.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

level **(fullword binary)**
Returns the current (not the dropped) application level identifier.

brkkey **(character length 8)**
Sets the CP TERM BRKKEY. If a single question mark is passed, this value is ignored.

Return Codes

0 Subroutine completed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MDSPLY - Display a menu without waiting for user input

```
MDSPLY        (number,retcode)
```

MDSPLY displays a menu and returns without waiting for user input. It can be used for output-only menus (such as status displays).

If this menu is associated with a window, then the virtual screen image is updated on the terminal. This window's viewport and any other filled, active windows are displayed. You can call MQACTW to determine which viewport contains the cursor and the key pressed.

If you change terminals in the middle of an application, unpredictable results can occur, however, if running a window application, automatic reconfiguration should take place.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine. If using the multiple terminal facility, the second fullword of "number" should be filled with the logical terminal number of the terminal you wish to address.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

11 The menu is too large for your terminal's screen size.

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MDSPRD - Display a menu, then wait for user input

```
MDSPRD      (number,retcode,key)
            (number,retcode,key,curoff)
            (number,retcode,key,curline,curcol)
```

MDSPRD displays a menu, then waits for user input. MDSPRD is the standard way to display menus.

MDSPRD performs the functions requested by MALARM, MCLEAR, MEVENT, MFLAGS, MMDT, MNULLS, MPAFLG, MPFMAP, MREMOT, MSKIP, MTRPA1, MUPCAS, MCURSF, MUNLKF, and MRMDTF.

If this menu is associated with a window, then the virtual screen image is updated on the terminal. This window's viewport and any other filled, active windows are displayed. If the user leaves the cursor outside of this viewport, the key pressed will be returned as "NOAID." You can call MQACTW to determine which viewport contains the cursor and the key pressed.

If you change terminals in the middle of an application, unpredictable results can occur, however, if running a window application, automatic reconfiguration should take place.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine. If using the multiple terminal facility, the second fullword of "number" should be filled with the logical terminal number of the terminal you wish to address.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

key **(character length 8)**
The key the user pressed to signal the end of input. This can be "ENTER," "PF01-PF36," "PA1-PA3," "TESTREQ," "SCANNER," "CLEAR," "NOAID," "LGHTPEN," "CARDRDR," "SMSG," "IUCV," "TIMER," "RDR" or "DEVICE."

curoff **(fullword binary)**
Specifies the cursor position as an absolute offset from the upper left corner of the menu where that corner is position zero.

curline **(fullword binary)**
Specifies the cursor line position as an offset from the top line of the menu where the top line is line one.

curcol **(fullword binary)**
Specifies the cursor column position as an offset from the left side of the menu where the left side is column one.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

11 The menu is too large for your terminal's screen size.

13 The last user input contained no cursor position (a PA or the CLEAR key was pressed).

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MDXSCR - Provide common VM Systems Group Internal Tools services in a single call

```
MDXSCR      (number,retcode,mblock)
            (number,retcode,mblock,flag)
```

MDXSCR is used to provide a set of common services in a single subroutine call, similar to those provided to an EXEC user by MENUEXEC.

This routine does the following:

1. Moves application data to a set of menu fields (like MD2SCR)
2. Optionally sets field attributes and extended attributes (like MFSET and MSEXTA)
3. Displays the menu and waits for user input (like MDSPRD)
4. Returns the key pressed, the name and the offset into the field the cursor was within (like MKEYP and MCURPF)
5. Moves user input data back to the application (like MUPSCR and MSCR2D)

Return codes are provided for each field in the data structure. The first non-zero return code is saved for each field operation. The main return code is the first non-zero return code for the entire operation.

A COBOL example of the data structure is given below.

Two utilities are provided to convert the DSECT output of XMEDIT into structures suitable for COBOL or PL/I programs, XMENUCOB and XMENUPLI respectively.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

mblock **(structure)**
Defines the fields, attributes, and data transferred to this subroutine. This structure is defined below.

flag **(fullword binary)**
Specifies whether extended attribute definitions are present. If zero, or if not specified, extended attribute definitions are not present. If non-zero, extended attribute definitions are present.

Contents of the MBLOCK Structure

This is an example of the mblock structure passed by a COBOL routine:

```
1 MENU UNALIGNED,
  2 FIELDS     FIXED BIN(31),
  2 KEY        CHAR(8),
  2 CURFNAM    CHAR(7),
  2 CURFOFF    FIXED BIN(31),
  2 FIELD1,
    3 NAME     CHAR(7),
    3 LENGTH   FIXED BIN(31),
    3 ATTRS,
      4 PROT   CHAR(7),
      4 BRIGHT CHAR(7),
      4 SKIP   CHAR(6),
      4 MDT    CHAR(6),
    3 RC       FIXED BIN(31),
    3 XATTRS,                  *  These fields are
      4 COLOR  CHAR(10),       *  only present
      4 HILITE CHAR(10),       *  if flag is
      4 PS     FIXED BIN(31),  *  non-zero
    3 BUFFER   CHAR(*),
  2 FIELD2,
      .
      .
      .
```

Return Codes

0 The operation completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

5 Data transfer - the field size is greater than the buffer size.

6 Data transfer - the buffer size is greater than the field size.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

11 The menu is too large for your terminal's screen size.

13 The last user input contained no cursor position (a PA or the CLEAR key was pressed).

14 The field name specified was not found in the menu.

18 The offset value (MSCR2D, MD2SCR) exceeds the fields size.

19 The field the cursor is on has no field name (MCURPF).

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MD2SCR - Move data to a menu field

```
MD2SCR      (number,retcode,fieldname,buffer,length,rlen)
            (number,retcode,fieldname,buffer,length,rlen,offset)
            (number,retcode,fieldname,buffer,length,rlen,offset,fill)
```

MD2SCR moves data from a program buffer to a menu named field. If the buffer is shorter than the field, the remainder of the field is filled with either nulls (binary zeros) or a specified fill character. Several calls can place different data into a single field; however, requests should proceed from left to right to avoid data being over-written by fill characters.

If MSIZEF was called with a non-zero return code, then differing sized buffers do not result in non-zero return codes from this routine.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field data is to be moved to.

buffer **(character length "length")**
The buffer from which the field data is moved.

length **(fullword binary)**
The number of characters of buffer to be moved to the named field.

rlen **(fullword binary)**
Returns the number of characters not moved to the named field if the field is shorter than "length."

offset **(fullword binary)**
If specified, denotes the number of characters from the leftmost part of the field to skip before moving data to the named field. If not specified, data is moved to the field starting with the field's first character.

fill **(character length 1)**
If specified, denotes the character used to fill out the unused rightmost remainder of the named field. If not specified, the remainder of the field is set to nulls (binary zeros).

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

5 Data transfer - the field size is greater than the buffer size.

6 Data transfer - the buffer size is greater than the field size.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

18 The offset value (MSCR2D, MD2SCR) exceeds the fields size.

MEVENT - Set/clear non-terminal interrupt events

```
MEVENT        (number,retcode,tsecs,smsg,iucv,rdr,device)
```

MEVENT is used to signal VM Systems Group Internal Tools to check for events other than terminal interrupts while displaying the menu referred to by "number." These events, in addition to terminal interrupts, will return control to your program during the call to "MDSPRD." If any of the MEVENT conditions occur, a "special key pressed" interrupt type is returned in the "key" parameter of "MDSPRD." MEVENT pertains only to the menu specified in "number" - MEVENT needs to be called for each menu needing additional event interruption.

The table below summarizes the different events and the type of interrupt presented (the interrupt type is presented as the "key pressed"):

Key Pressed Code	Type of Interrupt
TIMER	A specified amount of time has expired
SMSG	An SMSG or VMCF external interrupt
IUCV	An IUCV external interrupt
RDR	A file comes into your virtual reader
DEVICE	A specific device causes an interrupt

If you request a time interval condition, your virtual machine must have "ECMODE ON" to utilize the virtual clock comparator. For the other interrupt classes, it is the responsibility of your calling program to handle the interrupt; MEVENT only signals the VM Systems Group Internal Tools full screen processor to return on these interrupt types.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

tsecs **(fullword binary)**
The amount of time to wait in tenths of seconds. If the passed value is zero, no timer event is signalled (zero can be used to "reset" the previous timer requests).

smsg **(fullword binary)**
Non-zero if SMSG/VMCF interrupts are to be trapped; zero to ignore SMSG or to clear the previous request.

iucv **(fullword binary)**
Non-zero if IUCV interrupts are to be trapped; zero to ignore IUCV or to clear the previous request.

rdr **(fullword binary)**
Non-zero if virtual reader interrupts are to be trapped; zero to ignore reader interrupts or to clear the previous request.

device **(fullword binary)**
The virtual device address of the device for which interrupts are to be trapped; zero to ignore device interrupts or to clear the previous request.

Return Codes

0 Events set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MEXECI - Call CMS routine simulating an EXECCOMM environment

```
MEXECI      (number,retcode,command,length,vnames,vlengths,
              vmaxlths,fdata1,fdata2...fdatan)
```

MEXECI calls a CMS command, and simulates the EXECCOMM environment. This means that you can call a program such as MENUEXEC, GLOBALV, or EXECIO that reads and/or modifies EXEC variables, however, instead of manipulating EXEC variables, it manipulates data within your application.

The CMS command to be run is contained in command - its length is specified in length.

You pass a set of pseudo EXEC variable names, and buffers to contain them. All of these buffers must be in character format. Data to be transferred to the called program must be inside the buffers when MEXECI is called; if the data is modified, it is returned into the same buffer.

The number parameter contains the number of simulated EXEC variables contained in your program. The number of elements in vnames, vmaxlths, and vlengths, and the number of data buffer parameters is based on this value.

Vnames is an array of 32 character buffers, each containing the variable name of a simulated EXEC variable. Vlengths elements contain the length of the data in each data buffer. If the called program modifies the data buffer, the modified data length is returned in vlengths. Vmaxlths contains the maximum length of each data buffer, that is, the size allocated by the program - any data that is longer than this value is truncated.

Parameters

number **(fullword binary)**
Contains the number of simulated EXEC variables. This number represents the size of the name and length arrays and the number of data buffers passed. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

command **(character length "length")**
Contains the CMS command to be executed.

length **(fullword binary)**
Contains the byte length of the command to be executed.

buffer **(character length "length")**
The buffer the variable data is returned to.

length **(fullword binary)**
The length of buffer. If the buffer is smaller than the variable data length, the data is truncated.

vnames **(array of 32 byte character strings)**
Each element of this array contains a pseudo EXEC variable name, left justified, and blank filled to the right.

vlengths **(array of fullword binary values)**
Each element contains the current length of the data in each data buffer. If a buffer is modified, this length is set to the modified data length. A dropped or null variable has its length set to zero.

vmaxlth **(array of fullword binary values)**
Each element contains the maximum length of each data buffer, that is, the amount of storage allocated for it, or the maximum length string that this buffer can contain.

fdata1...fdatan **(character strings)**
Each parameter corresponds to one pseudo EXEC variable data area. These areas must contain character data, one character to a byte.

Return Codes

0 Data passed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

44 Your program is not running under an EXEC (MEXGET, MEXPUT).

45 The EXEC variable name does not exist (MEXGET).

47 The EXEC variable name or data length passed is invalid.

xxx Otherwise the CMS error return code.

MEXE2S - Move data from EXEC variables to menu fields

```
MEXE2S        (number,retcode,flag)
```

MEXE2S transfers data from EXEC variables to menu fields. Each named field on the menu has EXEC variable data moved to it (if any).

This routine can only be used if the EXECCOMM interface exists for this EXEC interpreter. If not, a return code of "89" is returned. Alternatively, a combination of calls to "MEXGET" and "MD2SCR" could be called for each menu field.

"MEXQRY" must be called prior to the invocation of this routine to initialize EXEC variable look up.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Ignored. Currently only significant for "MS2EXE."

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

89 MEXE2S, MS2EXE EXECCOMM not supported.

MEXGET - Get data from an EXEC variable

```
MEXGET      (number,retcode,varname,varlen,buffer,length,actual-length)
            (number,retcode,varname,varlen,buffer,length,actual-length,
              prefix)
```

MEXGET transfers the contents of an EXEC variable into your program buffer. "MEXQRY" must be called before using this subroutine. A return code is set if you are not running under an EXEC.

If prefix is specified, the string in prefix is prepended to the variable name passed in varname.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

varname **(character length "varlen")**
Contains the variable name requested. The leading ampersand for EXEC variables should not be included in the buffer. The variable name must be passed in upper case letters even if running under REXX.

varlen **(fullword binary)**
Contains the length of the variable name. The maximum length allowed for an EXEC name can be retrieved using MEXQRY. If the name exceeds the limit for EXEC 1, the name is truncated before the variable search begins. It is uncertain whether the contents of special variables can be returned by MEXGET - this is based on the operating system version and which interpreter you are running under.

buffer **(character length "length")**
The buffer the variable data is returned to.

length **(fullword binary)**
The length of buffer. If the buffer is smaller than the variable data length, the data is truncated.

actual_length **(fullword binary)**
The actual length of data moved into buffer by MEXGET.

prefix **(character string)**
Specifies a string to prepend to varname. This string can be up to 256 characters long. It must be terminated by a blank or null (binary zero) character.

Return Codes

0 Data passed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

44 Your program is not running under an EXEC (MEXGET, MEXPUT).

45 The EXEC variable name does not exist (MEXGET).

47 The EXEC variable name or data length passed is invalid.

MEXIT - Purge (release) all menus

```
MEXIT        (number,retcode)
```

MEXIT removes all active menus with the same or greater application identifier from storage. Further references to any menu are rejected with a return code of four.

If the current application identifier is zero (the default) the VM Systems Group Internal Tools internal trace table is purged from storage.

If any asynchronous interrupt trapping ("MSETAI") was in effect it is cancelled.

If any calls made to "MPRINT," "MPRNOH" or "MPRLNE" caused output on the virtual printer, the virtual printer is closed if requested (see the "close" parameter below). Closing a virtual printer causes all previous output to be placed into a single spool file.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

close **(fullword binary)**
Specifies whether to close the virtual printer if "MPRINT," "MPRNOH" or "MPRLNE" were used by this application. If not passed or if the value passed is non-zero the printer is closed if necessary. If the value passed is zero, no close is done.

Return Codes

0 Menu(s) purged.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MEXPUT - Put data into an EXEC variable

```
MEXPUT        (number,retcode,varname,varlen,buffer,length)
```

MEXPUT either transfers data into an EXEC variable or destroys an existing EXEC variable. If you are placing data into a variable the variable optionally may previously exist. If it did not previously exist, it is created (assuming that your "length" is positive). "MEXQRY" must be called before using this subroutine. A return code is set if you are not running under EXEC .

Based on the version of the operating system and which interpreter you are running under, unpredictable results may occur if you change the values of "special" variables or EXEC control word variables.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

varname **(character length "varlen")**
Contains the variable name. The leading ampersand for EXEC variables should not be included in the buffer. The variable name must be passed in upper case letters even if running under REXX.

varlen **(fullword binary)**
Contains the length of the variable name. The maximum length allowed for an EXEC name can be retrieved using "MEXQRY." If the name exceeds the limit for EXEC 1, the name is truncated before the variable search begins.

buffer **(character length "length")**
The buffer containing the data to be placed in the variable. If a variable is being destroyed (see "length" below) the contents are immaterial.

length **(fullword binary)**
The length of "buffer." If running under REXX and the value in "length" is negative the REXX variable is dropped. If the value is zero, the variable exists and contains the null string (""). The maximum length allowed for a variable can be retrieved from "MEXQRY." If the data is larger than the maximum supported size the data is truncated.

Return Codes

0 Data passed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
44 Your program is not running under an EXEC (MEXGET, MEXPUT).
45 The EXEC variable name does not exist (MEXGET).

46 There is insufficient storage available to create the EXEC variable.

47 The EXEC variable name or data length passed is invalid.

MEXQRY - Determine if running under an EXEC

```
MEXQRY      (number,retcode)
            (number,retcode,maxname,maxvalue)
```

MEXQRY determines whether your program is running under an EXEC processor. It also initializes VM Systems Group Internal Tools for future EXEC variable manipulations (see MEXGET and MEXPUT).

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
Returns a code if running under an EXEC.

maxname **(fullword binary)**
Returns the maximum number of characters that an EXEC variable name can be for the version of EXEC your program is running under. If the value returned is zero, there is no limit on the number of characters (other than storage size).

maxvalue **(fullword binary)**
Returns the maximum number of characters that an EXEC variable can contain for the version of EXEC your program is running under. If the value returned is zero, there is no limit on the number of characters (other than storage size).

Return Codes

0 Not running under an EXEC.

1 Running under CMS EXEC 1 (old style EXEC).

2 Running under CMS EXEC 2.

3 Running under CMS REXX (system product interperter).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MFLAGS - Set/clear flags for a menu

```
MFLAGS        (number,retcode,flagbyte)
```

MFLAGS simultaneously sets or resets the flags defined by "MALARM, "MCLEAR," "MMDT," "MNULLS," "MPFMAP," "MSKIP," "MTRPA1" and "MUPCAS." The effect of each of these subroutines is more fully described in its description in this manual.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flagbyte **(byte length 1)**
Each bit is set to a one to enable the given function, zero to disable the given function. The bits are defined below:

FLAG	VALUE	RESULT IF SET
MENUALRM	80	MALARM - sound alarm at output.
MENUSKIP	40	MSKIP - set protected field's SKIP bit.
MENUMDT	20	MMDT - set unprotected field's MDT bit.
MENUNULL	10	MNULL - change ending blanks in unprotected fields to nulls (binary zero)
MENUMAP	08	MPFMAP - return PF13-24 as PF01-12.
MENUPA1	04	MTRPA1 - trap the PA1 key.
MENUCLER	02	MCLEAR - clear screen before each menu output.
MENUUPCS	01	MUPCAS - convert input to upper case.

The values are given in hexadecimal. These flags may be combined. To simplify the passing of these flags, the macro "MENUFLGS" is supplied with VM Systems Group Internal Tools. This macro is further described in the VM Systems Group Internal Tools macros section of this manual.

Return Codes

0 Flags set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MFPOS - Return addresses of menu field blocks

```
MFPOS       (number,retcode,fieldname,addr-rec1,addr-rec2,addr-input)
            (number,retcode,fieldname,row,column)
```

MFPOS is used in special applications to return the VM Systems Group Internal Tools pointers to its internal control blocks for a particular menu field. These blocks can be relocated during "MADD"/"MDEL" calls; therefore, if you use this routine be sure to re-call MFPOS to reinitialize the control block pointers after such calls.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field to be located.

addr_rec1 **(fullword pointer)**
Returns the address of the menu control block (menu file record 1) for this field. The significant portions of this record include:

Offset	Length	Type	Value
0	1	Bits	Reserved for VM Systems Group Internal Tools internal use.
1	3	Binary	Field's displacement from the top of the screen. This value is the offset to the field's attribute character.
4	4	Binary	Length of the field (including the field's attribute character.
8	7	Char	The name of the field.
15	1	Byte	Reserved for calling program/MENUEXEC use.
16	1	Byte	Field attribute defined at menu creation.
17	1	Byte	Color attribute defined at menu creation.
18	1	Byte	Highlight attribute defined at menu creation.
19	1	Byte	Symbol set attribute defined at menu creation.
20	4	Binary	Relative offset to output data for this field.

This data is mapped by the "MENUREC" section of "MENUFMT" in "XMENU MACLIB."

addr_rec2 **(fullword pointer)**
Returns the address of the menu output area (menu file record 2) where this field's data begins (not at the field's attribute character).

addr_input **(fullword pointer)**
Returns the address of the input table for this field. This table has two fullword values. The first if non-zero is the address of the last input for this field. The second is the length of this input. If both are zero, no input exists for this field. If only the second is zero, the user pressed ERASE EOF.

row **(fullword pointer)**
Returns the row offset of the field from the top of the menu.

column **(fullword pointer)**
Returns the column offset from the left hand side of the menu.

Return Codes

0 Addresses passed to the caller.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

MFQRY - Query the attributes of a field

```
MFQRY      (number,retcode,fieldname,prot)
           (number,retcode,fieldname,prot,bright)
           (number,retcode,fieldname,prot,bright,skip)
           (number,retcode,fieldname,prot,bright,skip,mdt)
```

MFQRY returns character representations of the attributes of a field in the menu. The attributes returned are those currently in use; previous "MFSET" calls alter the values returned by MFQRY.

To query extended attribute values, use subroutine "MQEXTA."

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field to be queried.

prot **(character length 7)**
The protection attribute of the field. "Prot" may be "PROT" (protected), "UNPROT" (unprotected, any input) or "NUMERIC" (unprotected, numeric input).

bright **(character length 7)**
The intensity attribute of the field. "Bright" may be "BRIGHT" (intensified), "DIM" (normal display), "DARK" (non-display), or "LGHTPEN" (normal display, selector pen detectable).

skip **(character length 6)**
Returns "SKIP" if the field has the SKIP attribute, "NOSKIP" otherwise.

mdt **(character length 6)**
Returns "MDT" if the field's Modified Data Tag is set, "NOMDT" otherwise.

Return Codes

0 Data returned.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

MFSCW - Custom full screen output

```
MFSCW        (number,retcode,buffer,length)
```

MFSCW is used to output your own custom 3270 full screen data streams. It does not provide any additional VM Systems Group Internal Tools services other than transferring the data as is to the screen.

MFSCW causes an implicit "MRSHOW" - the next menu displayed will entirely refresh the screen (rather than send changed fields only).

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'00'	Issue Write command.
X'80'	Issue Write/Erase command.
X'C0'	Issue Write/Erase alternate.
X'10'	Trap PA1 if pressed at next input.
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer to be output to the screen. This buffer should contain all appropriate data and 3270 orders.

length **(fullword binary)**
Contains the length of "buffer."

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

MFSCWR - Custom full screen I/O

```
MFSCWR      (number,retcode,buffer,length,inbuff,inlen,inres)
```

MFSCWR is used to output your own custom 3270 full screen data streams and wait for full screen user input. It does not provide any additional VM Systems Group Internal Tools services other than transferring the data as is to the screen and retrieving the screen's input data.

MFSCW causes an implicit "MRSHOW" - the next menu displayed will entirely refresh the screen (rather than send changed fields only).

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'00'	Issue Write command.
X'80'	Issue Write/Erase command.
X'C0'	Issue Write/Erase alternate.
X'10'	Trap PA1 if pressed at next input.
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer to be output to the screen. This buffer should contain all appropriate data and 3270 orders.

length **(fullword binary)**
Contains the length of "buffer."

inbuff **(character length "inlen")**
The buffer to receive the full screen input.

inlen **(fullword binary)**
Contains the length of "inbuff." If more data is received than will fit into "inbuff," the data is truncated.

inres **(fullword binary)**
Returns the number of bytes of data moved into "inbuff."

Return Codes

0 Data displayed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

MFSET - Set the attributes of a field

```
MFSET      (number,retcode,fieldname,prot)
           (number,retcode,fieldname,prot,bright)
           (number,retcode,fieldname,prot,bright,skip)
           (number,retcode,fieldname,prot,bright,skip,mdt)
```

MFSET changes the current attributes of a field in the menu. The last changes made to a field's attributes are those in effect when the menu is next displayed.

To change a field's extended attributes, use the "MSEXTA" subroutine.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field whose attribute is to be changed.

prot **(character length 7)**
The protection attribute of the field. "Prot" may be "PROT" (protected), "UNPROT" (unprotected, any input) or "NUMERIC" (unprotected, numeric input). If "SAME" is specified, the current protection attribute is not changed. If your terminal does not have the numeric input feature, a field defined as "NUMERIC" acts as "UNPROT."

bright **(character length 7)**
The intensity attribute of the field. "Bright" may be "BRIGHT" (intensified), "DIM" (normal display), "DARK" (non-display), or "LGHTPEN" (normal display, selector pen detectable). "BRIGHT" fields are always potentially selector pen detectable. If "SAME" is specified, the current intensity attribute is not changed.

skip **(character length 6)**
Use "SKIP" to set the field's skip attribute, "NOSKIP" to clear it. If "SAME" is specified, the current skip attribute is not changed.

mdt **(character length 6)**
Use "MDT" to set the field's modified data tag, "NOMDT" to clear it. If "SAME" is specified, the current MDT attribute is not changed.

Return Codes

0 Data Set.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

15 Attributes cannot be set on a menu without any fields.

16 An attribute parameter passed was invalid.

MFSIZE - Determine the size of a menu field

```
MFSIZE        (number,retcode,fieldname,size)
```

MFSIZE returns the size of a menu field in bytes (number of characters).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field within the menu whose size is to be returned.

size **(fullword binary)**
Returns the size of the field in number of bytes/characters.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

MFSRB - Perform an unsolicited read buffer

```
MFSRB        (number,retcode,buffer,length,rlen)
```

MFSRB is used to do an unsolicited read buffer operation from the terminal.

This subroutine is intended for special purposes only.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer to receive the read buffer input.

length **(fullword binary)**
Contains the length of "buffer." If more data is read than can fit into "buffer" it is truncated.

rlen **(fullword binary)**
Contains the actual length of data read into "buffer."

Return Codes

0 The operation was successful.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.
644 Unit check occurred - sense data returned.
648 Full screen routine called before initialization.
652 Terminal type changed between or during full screen call.

MFSRD - Perform unsolicited read modified

```
MFSRD          (number,retcode,buffer,length,rlen)
```

MFSRD is used to perform an unsolicited read modified from the terminal. A keyboard unlock write is performed followed by a hardware read modified without waiting for user input.

If an asynchronous interrupt was detected by the use of "MSETAI" the keyboard unlock is not preformed.

This subroutine is intended for special purposes only.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer to receive the read modified input.

length **(fullword binary)**
Contains the length of "buffer." If more data is read than can fit into "buffer" it is truncated.

rlen **(fullword binary)**
Contains the actual length of data read into "buffer."

Return Codes

0 The operation was successful.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.
644 Unit check occurred - sense data returned.
648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

MFSRM - Wait for input then do read modified

```
MFSRM          (number,retcode,buffer,length,rlen)
```

MFSRM is used to wait for user input, then perform a read modified from the terminal.

This subroutine is intended for special purposes only.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer to receive the read modified input.

length **(fullword binary)**
Contains the length of "buffer." If more data is read than can fit into "buffer" it is truncated.

rlen **(fullword binary)**
Contains the actual length of data read into "buffer."

Return Codes

0 The operation was successful.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.
644 Unit check occurred - sense data returned.
648 Full screen routine called before initialization.
652 Terminal type changed between or during full screen call.

MFSWSF - Output write structured field

```
MFSWSF        (number,retcode,buffer,length)
```

MFSWSF is used to output your own custom 3270 write structured field data streams. It does not provide any additional VM Systems Group Internal Tools services other than transferring the data as is to the screen.

MFSWSF causes an implicit "MRSHOW" - the next menu displayed will entirely refresh the screen (rather than send changed fields only).

This routine is intended for special purposes only.

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer to be output to the screen. This buffer should contain all appropriate structured fields.

length **(fullword binary)**
Contains the length of "buffer."

Return Codes

0 The operation was successful.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

MFSWSR - Output structured field - get response

```
MFSWSR        (number,retcode,buffer,length,inbuff,inlen,inres)
```

MFSWSR is used to output your own custom 3270 structured field data streams and wait for full screen user input. It does not provide any additional VM Systems Group Internal Tools services other than transferring the data as is to the screen and retrieving the screen's input data.

MFSCW causes an implicit "MRSHOW" - the next menu displayed will entirely refresh the screen (rather than send changed fields only).

This subroutine is intended for special purposes only.

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer containing the structured fields to be output to the screen.

length **(fullword binary)**
Contains the length of "buffer."

inbuff **(character length "inlen")**
The buffer to receive the full screen input.

inlen **(fullword binary)**
Contains the length of "inbuff." If more data is received than will fit into "inbuff," the data is truncated.

inres **(fullword binary)**
Returns the number of bytes of data moved into "inbuff."

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

MGETLP - Retrieve the selector pen attention field

```
MGETLP        (number,retcode,buffer,length,actual-length)
```

MGETLP retrieves the contents of the selector pen attention field selected by the user (if any). The attention field is that field which caused an interrupt to be passed back to VM Systems Group Internal Tools. This routine only returns data if the last user interrupt was due to a selector pen attention field. A selector pen attention field begins with a blank or null (binary zero) character.

On some terminals the "CURSOR SEL" key can be used in place of a selector pen.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer to contain the contents of the selector pen attention field.

length **(fullword binary)**
The length of "buffer" in bytes.

actual_length **(fullword binary)**
The actual length of data moved into "buffer." If buffer is smaller than the field's actual length, the data is truncated.

Return Codes

0 Data retrieved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

49 The last user interrupt was not a selector pen attention.

50 A selector pen attention interrupt occurred but no attention field was returned by the hardware.

MGETRC - Given a VM Systems Group Internal Tools subroutine, return a message

```
MGETRC        (number,retcode,buffer,length,actlen)
```

MGETRC is used to retrieve a one line, informative message about an VM Systems Group Internal Tools subroutine return code.

You can use this routine to give a user or application developer an online explanation for an VM Systems Group Internal Tools subroutine error condition.

Parameters

number **(fullword binary)**
Ignored. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
Contains the return code whose message is to be returned. Retcode is then changed to the return code for this subroutine call.

buffer **(character length "length")**
The message text for the passed return code is returned into this buffer.

length **(fullword binary)**
The length of the character buffer used to retrieve the message text.

actlen **(fullword binary)**
Returns the actual length of the message text moved into buffer.

Return Codes

0 The message text has been returned.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

104 Retcode passed to MGETRC doesn't have a message.

MGETTR - Get the VM Systems Group Internal Tools trace table pointers

```
MGETTR        (number,retcode,first,last,current)
```

MGETTR is used to retrieve the VM Systems Group Internal Tools trace table pointers.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

first **(fullword binary pointer)**
Returns the storage address of the start of the trace table. If zero is returned, there is no trace table.

last **(fullword binary pointer)**
Returns the storage address of the byte following the last entry in the trace table.

current **(fullword binary pointer)**
Returns the storage address of the next available trace table entry.

Return Codes

0 Subroutine completed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MGFILE - Retrieve data from MENUCTRL file

```
MGFILE      (number,retcode,filename,all)
            (number,retcode,filename,fname1,...fnamen)
            (number,retcode,filename,null,all)
            (number,retcode,filename,null,fname1,...fnamen)
```

MGFILE moves data saved in a MENUCTRL file to the menu. A MENUCTRL file can have been previously created by MENUEXEC, this program or a previous program. Only data statements within the MENUCTRL file are examined; all others, including MENUEXEC subcommands, are ignored.

Data records pertaining to fields not in the menu, or fields not in the parameter list (unless "ALL" is specified) are ignored.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

filename **(character length 8)**
The file name of the MENUCTRL file.

null **(character length 8)**
Provides a space-holder where the TSO DDNAME would reside. This is provided to simplify migration of a program from CMS to TSO and vice versa. The string within "null" must not be the name of any field in the menu or the word "ALL."

all **(character length 8)**
If the first field name is specified as "ALL," then any and all data records that match names of fields in the menu are updated from the MENUCTRL file. If "ALL" is present field names following "ALL" are ignored.

fname1...fnamen (character length 8)
A list of the fields to be updated from the MENUCTRL file. If a name in the list is not found in the MENUCTRL file it is ignored. Fields in the MENUCTRL file but not in the list or menu are also ignored.

Return Codes

0 MENUCTRL file processed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

23 A field name specified in MGFILE or MUFILE is invalid or does not exist on the menu.

25 The MENUCTRL file referenced in MGFILE or MUFILE is improperly formatted.

26 A badly formatted data record was found in the MENUCTRL file.

29 The MENUCTRL file requested does not exist (MGFILE).

4xx Error in CMS FSREAD (xx is the FSREAD code).

402 Invalid input buffer address.

403 Permanent I/O error occurred.

408 Incorrect length record read.

412 End of file detected.

425 Insufficient free storage.

MGTLPF - Return the selector pen attention field name

```
MGTLPF        (number,retcode,fieldname)
```

MGTLPF returns the field name of the selector pen attention field selected by the user (if any). The attention field is that field which caused an interrupt to be passed back to VM Systems Group Internal Tools. This routine only returns data if the last user interrupt was due to a selector pen attention field. A selector pen attention field begins with a blank or null (binary zero).

On some terminals the "CURSOR SEL" key can be used in place of a selector pen.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The buffer to which the field name of the selector pen attention field is returned.

Return Codes

0 Data retrieved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

49 The last user interrupt was not a selector pen attention.

50 A selector pen attention interrupt occurred but no attention field was returned by the hardware.

MIOLOG - Create CMS file logging terminal I/O

```
MIOLOG        (number,retcode,flag)
```

MIOLOG sets or resets a flag that determines whether all VM Systems Group Internal Tools terminal I/O is logged to a CMS file. If the flag is non-zero, I/O logging is performed. If zero, no logging takes place. If this routine is never called no logging takes place.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MJUL2V - Convert a Julian date to a Gregorian date

```
MJUL2V      (julian,retcode,date)
```

MJUL2V converts a Julian date (YY.DDD) into a Gregorian date (MM/DD/YY).

Parameters

julian **(character length 6)**
Input date in "YY.DDD" format.

retcode **(fullword binary)**
The return code for the conversion (see below).

date **(character length 8)**
Output date in "MM/DD/YY" format.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

32 Invalid MM/DD/YY passed to MV2TOD or MV2JUL, invalid YY.DDD to MJUL2V.

MKEYP - Determine input key pressed

```
MKEYP       (number,retcode,key)
            (number,retcode,key,data)
```

MKEYP retrieves the interrupt condition used to terminate user input and optionally specifies whether any menu fields were changed by the user.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

key **(character length 8)**
The key the user pressed to signal the end of input. This can be "ENTER," "PF01-PF24," "PA1-PA3," "TESTREQ," "SCANNER," "CLEAR," "NOAID," "LGHTPEN," "CARDRDR," "SMSG," "IUCV," "TIMER," "RDR" or "DEVICE."

data **(fullword binary)**
Specifies whether any user data was returned with the interrupting condition. If "data" is zero, only an interrupt key and possibly a cursor position were returned. If data is non-zero, some data was returned with the interrupt; a field or fields were modified by the user.

Return Codes

0 Data retrieved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

MLD2S - Move data to a set of menu fields

```
MLD2S        (number,retcode,count,fields,buffers,lengths,rlens,rcodes)
```

MLD2S moves sets of data from program buffers to menu named fields. If any field is longer than its respective buffer, the remainder of the field is filled with nulls (binary zeros - no fill character can be specified).

The number of fields filled with data is determined by the value in the "count" parameter. For each move operation an element of each array is used to get the field name, the buffer address and the buffer length; the last two arrays are used to return the unmoved data count (if any) and the return code for the step.

This subroutine cannot be called by languages that do not support pointer type variables such as COBOL or FORTRAN.

If MSIZEF was called with a non-zero return code, then differing sized buffers do not result in non-zero return codes from this routine.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

count **(fullword binary)**
The number of distinct field data move requests (the number of elements in each of the arrays defined below).

fields **(array of character length 8 elements)**
A list of the names of each named field to which data is to be moved.

buffers **(array of fullword buffer pointers)**
A list of the addresses of each program buffer moved to a named field.

lengths **(array of fullword binary values)**
The length of each respective buffer to be moved to a named field.

rlens **(array of fullword binary values)**
For each move operation, the length of the data not moved to the named field (if the named field is shorter than the buffer).

rcodes **(array of fullword binary values)**
The return code for each respective move operation.

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

5 Data transfer - the field size is greater than the buffer size.

6 Data transfer - the buffer size is greater than the field size.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

MLJUST - Left justify a field's data

```
MLJUST      (number,retcode,fieldname)
            (number,retcode,fieldname,zero,fill)
            (number,retcode,fieldname,zero,fill,type)
            (number,retcode,buffer,length)
            (number,retcode,buffer,length,fill)
            (number,retcode,buffer,length,fill,type)
```

MLJUST left justifies data within a field or a user's buffer. Normally, all blanks or nulls (binary zeros) in the leftmost part of the field or string are truncated. Unused positions to the right of the data are changed to nulls.

If "fill" is specified, this character is used to fill unused positions instead of nulls. If "type" is specified and non-zero, only leftmost nulls are truncated.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The field to be justified.

buffer **(character length "length")**
The buffer to be justified. The justified data is returned to this buffer.

length **(fullword)**
The length of the buffer to be justified.

zero **(fullword)**
A filler parameter that must be passed as a fullword of binary zero in order to distinguish the "fieldname" call vs. the "buffer"/"length" call. This is only necessary when you want to pass "fill" and/or "type."

fill **(character length 1)**
The character used to fill positions freed by the shifting of data. If not specified, unused positions are filled with nulls (binary zeros).

type **(fullword)**
A flag used to determine whether to truncate blanks and nulls or only nulls. If the value is non-zero, only nulls are truncated.

Return Codes

0 Data left justified.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

MLOAD - Load a menu into storage

```
MLOAD       (number,retcode,menuname)
            (number,retcode,menuname,libname)
            (number,retcode,addr-rec1,addr-rec2)
```

MLOAD loads a menu to be manipulated and displayed by a program. It can be loaded from either a disk file, an VM Systems Group Internal Tools XMENULIB library or from a pre-loaded in-storage copy.

If a menu was loaded into storage using the XMENUINS program this copy will be used to load the working menu even if an XMENULIB was specified.

MLOAD returns a unique value in the parameter "number." Pass this value in all other subroutine "number" parameters to manipulate this loaded menu. The same named menu may be loaded more than once for different uses; each is assigned a unique number and may be used differently without affecting other copies with different numbers.

Other than your virtual storage size, there is no restriction on the number of loaded active menus; each menu and its associated control blocks require approximately 10K bytes of storage.

The third form of the MLOAD subroutine is used for special purposes where you build a menu dynamically in storage or if you have a menu pre-loaded with your program (see the TEXT option of the XMENU program in the "VM Systems Group Internal Tools User's Guide").

Parameters

number **(fullword binary)**
Returns the unique number by which the menu is referenced in subsequent VM Systems Group Internal Tools subroutine calls. The value is assigned by VM Systems Group Internal Tools. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

menuname **(character length 8)**
The name of the menu to be loaded.

libname **(character length 8)**
The name of the XMENULIB library the menu is within. If this parameter is passed as eight blanks, MLOAD acts as if no XMENULIB was specified.

addr_rec1 **(fullword pointer)**
This parameter points to a fullword containing the address of record one of a pre-loaded menu. MLOAD copies the menu to its own buffers; the passed menu is neither used nor modified.

addr_rec2 **(fullword pointer)**
This parameter points to a fullword containing the address of record two of a pre-loaded menu. MLOAD copies the menu to its own buffers; the passed menu is neither used nor modified.

Return Codes

0 Menu loaded successfully.

1 Your terminal is not a 327x.

2 MLOAD, MLOADX - The menu file you are attempting to load is not formatted properly.

2 MQEXTP - The attribute data passed is improperly formatted.

2 MFQRY, MFSET, MQEXTA, MSEXTA The field attribute could not be found.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

11 The menu is too large for your terminal's screen size.

28 The menu file requested does not exist (MLOAD, MLOADX).

30 The CMS XMENULIB is improperly formatted.

31 The CMS XMENULIB menu member does not exist.

1xx Error in CMS FSSTATE (xx is the FSSTATE code).

120 Invalid character in the file name.

124 Invalid CMS FILEMODE.

136 The disk is not accessed.

4xx Error in CMS FSREAD (xx is the FSREAD code).

402 Invalid input buffer address.

403 Permanent I/O error occurred.

408 Incorrect length record read.

412 End of file detected.

425 Insufficient free storage.

MLOADW - Load a menu into a window

```
MLOADW      (number,retcode,wname,flags,menuname)
            (number,retcode,wname,flags,menuname,libname)
            (number,retcode,wname,flags,addr-rec1,addr-rec2)
```

MLOADW loads a menu to be manipulated and displayed by a program into a window. If the window doesn't already exist, it is created.

The menu can be loaded from either a disk file, an VM Systems Group Internal Tools XMENULIB library or from a pre-loaded in-storage copy.

If a menu was loaded into storage using the XMENUINS program this copy will be used to load the working menu even if an XMENULIB was specified.

MLOAD returns a unique value in the parameter number. Pass this value in all other subroutine number parameters to manipulate this loaded menu. The same named menu may be loaded more than once for different uses; each is assigned a unique number and may be used differently without affecting other copies with different numbers.

Other than your virtual storage size, there is no restriction on the number of loaded active menus.

If the window already exists, it must be the same size as the menu to be loaded otherwise a return code of 719 is returned.

If the window does not exist, it is implicitly created. If the virtual screen does not exist, it is also implicitly created.

If the window and/or the virtual screen is implicitly created, it is automatically sized according to the menu size. Both the window and the menu must be equal to or smaller than the virtual screen size.

The passed flags are used to determine whether this is an active (displayed) window, whether borders are displayed around its viewport and whether the menu name is used as a title in the top viewport border.

The third form of the MLOAD subroutine is used for special purposes where you build a menu dynamically in storage or if you have a menu pre-loaded with your program (see the TEXT option of the XMENU program in the "VM Systems Group Internal Tools User's Guide").

Parameters

number **(fullword binary)**
Returns the unique number by which the menu is referenced in subsequent VM Systems Group Internal Tools subroutine calls. The value is assigned by VM Systems Group Internal Tools. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

flags **(byte)**
Specifies how the window's viewport is displayed. The following bits are currently defined:

WINVBOR X'80' Display vertical borders if possible
WINHBOR X'40' Display horizontal borders if possible
WINTITLE X'20' Display the menu name on the top border if possible
WINACTIV X'01' This viewport is active, that is, it is visible

menuname **(character length 8)**
The name of the menu to be loaded.

libname **(character length 8)**
The name of the XMENULIB library the menu is within. If this parameter is passed as eight blanks, MLOAD acts as if no XMENULIB was specified.

addr_rec1 **(fullword pointer)**
This parameter points to a fullword containing the address of record one of a pre-loaded menu. MLOAD copies the menu to its own buffers; the passed menu is neither used nor modified.

addr_rec2 **(fullword pointer)**
This parameter points to a fullword containing the address of record two of a pre-loaded menu. MLOAD copies the menu to its own buffers; the passed menu is neither used nor modified.

Return Codes

0 Menu loaded successfully
window created if necessary.

1 Your terminal is not a 327x.

2 MLOAD, MLOADX - The menu file you are attempting to load is not formatted properly.

2 MQEXTP - The attribute data passed is improperly formatted.

2 MFQRY, MFSET, MQEXTA, MSEXTA The field attribute could not be found.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

28 The menu file requested does not exist (MLOAD, MLOADX).

30 The CMS XMENULIB is improperly formatted.

31 The CMS XMENULIB menu member does not exist.

1xx Error in CMS FSSTATE (xx is the FSSTATE code).

120 Invalid character in the file name.

124 Invalid CMS FILEMODE.

136 The disk is not accessed.

4xx Error in CMS FSREAD (xx is the FSREAD code).

402 Invalid input buffer address.

403 Permanent I/O error occurred.

408 Incorrect length record read.

412 End of file detected.

425 Insufficient free storage.

700 Insufficient storage to get virtual screen control block(s).

701 Insufficient storage to get window control block(s).

703 A window with this name already exists.

706 A vertical size was passed without a horizontal size.

708 Insufficient storage to get virtual screen data array(s).

709 Window vertical size > virtual screen vertical size.

710 Window horizontal size > virtual screen horizontal size.

711 Insufficient storage to get window data array(s).

MLOADX - Load a menu regardless of size

```
MLOADX      (number,retcode,menuname)
            (number,retcode,menuname,libname)
            (number,retcode,addr-rec1,addr-rec2)
```

MLOADX loads a menu to be used by a program. MLOADX differs from MLOAD in that the menu is loaded, even if it is too large to fit on the screen. This subroutine is meant for special purposes only, such as by the XMENU menu editor or to load and print a menu larger than the terminal size without displaying it. If you load a menu with MLOADX and attempt to display it on a screen smaller than the menu size, a terminal I/O error may occur.

Parameters

number **(fullword binary)**
Returns the unique number by which the menu is referenced in subsequent VM Systems Group Internal Tools subroutine calls. The value is assigned by VM Systems Group Internal Tools. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

menuname **(character length 8)**
The name of the menu to be loaded.

libname **(character length 8)**
The name of the XMENULIB library the menu is within. If this parameter is passed as eight blanks, MLOADX acts as if no XMENULIB was specified.

addr_rec1 **(fullword pointer)**
This parameter points to a fullword containing the address of record one of a pre-loaded menu. MLOADX copies the menu to its own buffers; the passed menu is neither used nor modified.

addr_rec2 **(fullword pointer)**
This parameter points to a fullword containing the address of record two of a pre-loaded menu. MLOADX copies the menu to its own buffers; the passed menu is neither used nor modified.

Return Codes

0 Menu loaded successfully.

1 Your terminal is not a 327x.

2 MLOAD, MLOADX - The menu file you are attempting to load is not formatted properly.

2 MQEXTP - The attribute data passed is improperly formatted.

2 MFQRY, MFSET, MQEXTA, MSEXTA The field attribute could not be found.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

28 The menu file requested does not exist (MLOAD, MLOADX).

30 The CMS XMENULIB is improperly formatted.

31 The CMS XMENULIB menu member does not exist.

1xx Error in CMS FSSTATE (xx is the FSSTATE code).

120 Invalid character in the file name.

124 Invalid CMS FILEMODE.

136 The disk is not accessed.

4xx Error in CMS FSREAD (xx is the FSREAD code).

402 Invalid input buffer address.

403 Permanent I/O error occurred.

408 Incorrect length record read.

412 End of file detected.

425 Insufficient free storage.

MLS2D - Retrieve input data from a set of menu fields

```
MLS2D          (number,retcode,count,fields,buffers,lengths,rlens,rcodes)
```

MLS2D moves data from the last user input into a group of menu fields to application program buffers. If any buffer is longer than its respective field, the remainder of the buffer is filled with nulls (binary zeros - no fill character can be specified).

The number of buffers to be filled with data is determined by the value in the "count" parameter. For each move operation an element of each array is used to get the field name, the buffer address and the buffer length; the last two arrays are used to return the transferred data count (if any) and the return code for the step.

This subroutine cannot be called by languages that do not support pointer type variables such as COBOL or FORTRAN.

If MSIZEF was called with a non-zero return code, then differing sized buffers do not result in non-zero return codes from this routine.

Unlike MSCR2D, data is only returned from a field if it was input by the user or if the field's Modified Data Tag (MDT) was set. The following table describes the possible conditions and the data returned by MLS2D:

Field condition	Rlen	Retcode	Buffer contents
User supplied input	non-zero	0	Changed
User pressed ERASE EOF	zero	0	Nulls (binary zeros)
User entered nothing	zero	10	Unchanged

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The return code for this request. This value is the last non-zero return code encountered. If no non-zero return codes occurred, "retcode" is zero.

count **(fullword binary)**
The number of distinct field input move requests (the number of elements in each of the arrays defined below).

fields **(array of character length 8 elements)**
A list of the names of each named field from which data is to be moved.

buffers **(array of fullword buffer pointers)**
A list of the addresses of each program buffer to which user input is moved.

lengths **(array of fullword binary values)**
The length of each respective buffer.

rlens **(array of fullword binary values)**
For each move operation, the length of the user input data actually moved to the respective buffer. The values returned in "rlens" for this subroutine are different than those returned by "MLD2S," "MSCR2D" or "MD2SCR" in that the values are actual data transfer counts rather than counts of data not transferred.

rcodes **(array of fullword binary values)**
The return code for each respective move operation.

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

5 Data transfer - the field size is greater than the buffer size.

6 Data transfer - the buffer size is greater than the field size.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

10 No user input made to the requested field (MTSTF, MLS2D).

14 The field name specified was not found in the menu.

MMAKEA - Create a new application level

```
MMAKEA      (number,retcode,level)
            (number,retcode,level,brkkey)
```

MMAKEA creates a new application level identifier, and optionally returns the current setting of the CP TERM BRKKEY.

This call should only be necessary when two different applications share a copy of the VM Systems Group Internal Tools subroutine package and data area passed by "MRENT."

This routine should be called by the routine giving control to a second application. On return from that application "MDROPA" should be called. This prevents the second application from modifying or purging any menus from an application level lower than its own.

A menu can be shared between application levels by calling "MSHARE."

If the brkkey parameter is passed, the current setting of the CP TERM BRKKEY is returned. This can be used to save and later restore (using MDROPA) the setting of the BRKKEY.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

level **(fullword binary)**
Returns the application identifier for this newly created level.

brkkey **(character length 8)**
Returns the CP TERM BRKKEY setting. If this cannot be determined, a single question mark is returned.

Return Codes

0 Subroutine completed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MMDT - Set Modified Data Tags (MDT) before the output of a menu

```
MMDT          (number,retcode,flag)
```

MMDT sets or resets a flag that determines whether unprotected field's Modified Data Tags (MDT) are set before output. Modified Data Tags are used to "simulate" user input from fields. If the MMDT flag is non-zero, Modified Data Tags are set in all unprotected fields when the menu is next output. If the flag is zero, Modified Data Tags are left unchanged. If never called the default MDT flag is off. Each loaded menu has its own MDT flag.

If this routine is called with a non-zero flag, the MDT bits are set at the next call to a display subroutine (MDSPRD, etc.). Once this is done, the MDT bits remain on for the remaining use of the menu unless individually reset even if MMDT is again called with a flag of zero (since this means the MDT bits are to be left unchanged).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MNEXTF - Return the first/next named field in a menu.

```
MNEXTF        (number,retcode,fieldname,foffset,fflag,fsize,isize)
```

MNEXTF returns information for each named field in a menu. To obtain information about the first named field in a menu, the value of "foffset" should be set to zero. After the subroutine is called, "foffset" is modified so that a subsequent call will return information about the next named field. When no more named fields remain, a return code of "91" is returned.

Based on the setting of "fflag," either all or only unprotected, named fields are returned.

"Fsize" returns the size of the field in question. "Isize" returns "-1" if no input was made into this field, "zero" if "ERASE EOF" was pressed or the input data length if input was made.

This subroutine is used to quickly go through a menu looking for fields to be processed. This subroutine obviates the need for your application to know the field names in the menu.

The order in which fields are returned is undefined.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
Returns the name of the field whose information is returned. This location must be in read/write storage.

foffset **(fullword binary)**
Contains a value used to determine which field information is being returned. To get the first field's information, set this location to zero. Following the call, this location is changed by VM Systems Group Internal Tools so that a subsequent call will return the next field's information. When no more fields can be transferred a return code of "91" is returned and this location remains unchanged. It must be set to zero by your application to process the fields from the beginning. This location must be in read/write storage.

fflag **(fullword binary)**
Contains a flag used to determine which fields are to be returned. If the value of this flag is zero, all named fields are returned. If the value of this flag is non-zero, only unprotected named fields are returned.

fsize **(fullword binary)**
Returns the size of the field in number of bytes/characters.

isize **(fullword binary)**
Returns whether the field was modified, and if so, the size in bytes/characters of the user input data. If no user input was made, a "-1" is returned. If "ERASE EOF" was pressed, zero is returned. If user input was entered, the number of bytes/characters is returned.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

91 No more field names found or offset number is invalid.

MNULLS - Set the ends of input fields to nulls in a menu

```
MNULLS        (number,retcode,flag)
```

MNULLS sets or resets a flag that determines whether blank characters at the end of input areas are set to nulls (binary zeros) before the menu is output. If the flag is non-zero, ends of input fields are set to nulls. If the flag is zero, ends of input fields remain unchanged. If this routine is never called, the default null flag is off. Each loaded menu has its own null flag.

Nulls in unused rightmost positions of a field allow the operator to use the "INSERT" key to insert data between existing characters in a field. In addition, nulls are not returned as input data.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MPAFLG - Capture data if a PA key is pressed

```
MPAFLG        (number,retcode,flag)
```

MPAFLG sets or resets a flag which determines whether VM Systems Group Internal Tools attempts to capture modified data when a PA key is pressed. If the flag is set, VM Systems Group Internal Tools will attempt to automatically capture modified data when a PA key is pressed. If not set (the default) only the key pressed is returned when a PA key is pressed.

PA data capture is a global flag - if it is set, all menus will have their PA key data captured.

Not all operating systems support data capture when PA keys are pressed.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear the PA key flag, non-zero to set the PA key flag.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MPCUR - Position the Cursor

```
MPCUR       (number,retcode,curoff)
            (number,retcode,curline,curcol)
            (number,retcode,fieldname)
            (number,retcode,fieldname,offset)
```

MPCUR sets the position of the cursor in the menu for the next menu output. The cursor may be positioned via absolute menu position, by menu row and column or within a named field.

If the cursor is not specifically positioned before output, the cursor is placed where it was defined in the menu if MPCUR was never called, where MPCUR last placed it, or where it was on last input if MUPSCM was called.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

curoff **(fullword binary)**
Specifies the cursor position as an absolute offset from the upper left corner of the menu where that corner is position zero.

curline **(fullword binary)**
Specifies the cursor line position as an offset from the top line of the menu where the top line is line one.

curcol **(fullword binary)**
Specifies the cursor column position as an offset from the left side of the menu where the left side is column one.

fieldname **(character length 7)**
Specifies that the cursor is to be placed within the field named "fieldname." If "offset" is not specified, it is placed in the first position of the field (one past the field attribute).

offset **(fullword binary)**
Specifies the offset within field "fieldname" to place the cursor. The leftmost position of the field is location zero.

Return Codes

0 Cursor positioned successfully.

3 The menu position requested is invalid (greater than the screen size).

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

MPFMAP - Map input of PF13-24 to PF01-12 from a menu

```
MPFMAP        (number,retcode,flag)
```

MPFMAP sets or resets a flag that determines whether program function keys PF13 through PF24 are returned by MDSPRD, MRDSPR, MRDSPC, MKEYP or MPGD2S as PF01 to PF12. If the flag is non-zero, the program function keys are mapped. If the flag is zero, the program function keys are not mapped. If this routine is never called, the default map flag is off. Each loaded menu has its own map flag.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MPGCLR - Clear a line mode menu line output area

```
MPGCLR      (number,retcode)
            (number,retcode,flags)
```

MPGCLR is used to clear the line output area in a line mode menu. In addition, the next output line location is reset to the first (top) line in the line output area.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flags **(binary length 2)**
Flags passed by your program and/or returned by VM Systems Group Internal Tools to control the action of this subroutine. The following describes the defined flags:

FLAG	VALUE	RESULT IF SET
MPAGNTFG	0080	Don't display data (line mode menu only).

The value defined above is in hexadecimal. If MPAGNTFG is specified, the menu is cleared internally but no display occurs.

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

65 The menu number passed is not a page mode menu.

MPGD2S - Move a row of data to a page mode menu

```
MPGD2S      (number,retcode,flags,key,linedefs,bufcnt,cmdbuf,
               cmdbufl,cmdimpl,BUF1,BUF1L,...BUFn,BUFnL)
            (number,retcode,flags,key,linedefs,bufcnt,
               BUF1,BUF1L,...BUFn,BUFnL)
```

MPGD2S is used to move a row of data from your program buffer(s) to a horizontal row of fields in a page mode menu. The attribute for each field in the row can be changed as well.

All rows can be cleared and/or their attributes can be reset to their values at menu creation before data movement.

Normally the next available row in the menu is used to hold the output. If the bottom row was used at the last call the top row is used at this call. It is also possible to direct output to a specific field by row and column if desired.

Depending on the flags passed, the menu may be displayed at the time of the transfer and/or VM Systems Group Internal Tools may wait for user input.

Also depending on the flags passed, data may be converted from binary to decimal or hexadecimal characters, optionally right justified and/or zero suppressed.

If the menu contains one data field (a "line mode" menu) the single line of data passed by this routine is displayed as soon as it it moved to the menu (unless a specific flag is set - see below). If the menu is scrolled to previous data (see "MPGSCR") the entire menu may be refreshed.

This subroutine uses the same next line pointer as "MPGS2D."

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flags **(binary length 2)**
Flags passed by your program and/or returned by VM Systems Group Internal Tools to control the action of this subroutine. The following describes the defined flags:

FLAG	VALUE	RESULT IF SET
MPAGDFLG	8000	Display the menu now.
MPAGDRFL	4000	Get user input.
MPAGADSP	2000	VM Systems Group Internal Tools displayed the menu automatically.
MPAGCRSP	1000	Output data to a specific row/column.
MPAGWER	0800	Clear all data fields before moving data.
MPAGWERA	0400	Reset all data field attributes.
MPAGDTYP	0200	Data type information has been passed.
MPAGAUTO	0100	Let VM Systems Group Internal Tools display the menu if necessary.
MPAGNTFG	0080	Don't display data (line mode menu only).

The values defined above are in hexadecimal. These flags may be combined. To simplify the passing of these flags, the macro "MPAGFLGS" is supplied with VM Systems Group Internal Tools. This macro is further described in the VM Systems Group Internal Tools macros section of this manual.

MPAGADSP is the only bit modified by VM Systems Group Internal Tools. It is set/cleared only if MPAGAUTO is set. If MPAGAUTO or MPAGDFLG are set VM Systems Group Internal Tools waits for user input only if MPAGDRFL is set.

If MPAGAUTO is set, the menu is automatically displayed if data is moved to the last row of columnar data.

If the page mode menu consists of one field (line mode menu) and if MPAGNTFG is set, the line of data passed is added to the internal copy of the menu and the saved line queue but is not displayed immediately.

key **(character length 8)**
The key the user pressed to signal the end of input. This can be "ENTER," "PF01-PF24," "PA1-PA3," "TESTREQ," "SCANNER," "CLEAR," "NOAID," "LGHTPEN," "CARDRDR," "SMSG," "IUCV," "TIMER," "RDR" or "DEVICE."

If an invalid attribute was passed, the return code is set to 78 and the storage address of the offending attribute is returned in the first four bytes of "key."

linedefs **(2 fullwords binary)**
If using a line mode menu, this area returns in the first fullword the number of this data line in the saved line queue. The second fullword returns the line of the menu this data was placed on.

If bit MPAGCRSP in "flags" is set this area passes data to position the output to one specific field in the row/column array on the menu. The first fullword contains the column number (0=col A, 1=col B,), the second fullword contains the row number (0=row 1, 1=row 2,).

bufcnt **(fullword binary)**
The number of buffers (columns) passed.

cmdbuf **(fullword binary)**
If input is received, this contains the data returned in the command line field of the menu. This parameter is passed only if bit MPAGDRFL in "flags" is set.

cmdbufl **(fullword binary)**
This field contains the length of "cmdbuf." This parameter is passed only if bit MPAGDRFL in "flags" is set.

cmdimpl **(fullword binary)**
This field returns the actual length of data returned into "cmdbuf." This parameter is passed only if bit MPAGDRFL in "flags" is set.

The following parameters are passed in groups of two, one for each field of output to be transferred. The first buffer (column) sets output for the leftmost field on the row. Each following buffer sets data from the field to the right of the preceding field.

BUF1...BUFn **(binary data of varying length)**
Data to be output to a field on this row. If the attribute for this field needs to be overridden, it is placed in the first 10 bytes of "BUF1."

If an attribute is passed, it must be in the following form:

Byte	Contents
0	SFE attribute byte (X'29')
1	Count of following pairs (X'04')
2	Color type (X'42')
3	Color value (X'F1' BLUE, X'F2' RED, X'F3' PINK, X'F4' GREEN, X'F5' TURQUOISE, X'F6' YELLOW, X'F7' WHITE, X'00' Default)
4	Highlighting type (X'41')
5	Highlighting value (X'00' None, X'F1' BLINK, X'F2' REVERSE, X'F3' UNDERSCORE)
6	Logical symbol set type (X'43')
7	Logical symbol set value (X'00' default, X'40-EF' logical symbol set)
8	Standard attribute type (X'C0')
9	Standard attribute value (a bit definition - see the section on 3270 attributes for more information)

The macro "MENUATT" is provided with VM Systems Group Internal Tools to ease in the definition of this attribute. See the macro section in this book for more details.

BUF1L...BUFnL (fullword binary optionally followed by 1 byte flag and character length 7).
The fullword contains the length of the line pointed to by BUF1, including any leading attribute. The following flag and character are only examined if bit MPAGDTYP of "flags" is set. If set, the flag byte is defined as follows:

FLAG	VALUE	RESULT IF SET
UBUFHEX	80	Convert passed data to hexadecimal.

UBUFDEC	40	Convert passed data to decimal.
UBUFIGNA	04	Header of data is not an attribute.
UBUFRJ	02	Right justify data.
UBUFZS	01	Zero suppress data.

The values above are given in hexadecimal. To simplify the passing of these flags, the macro "UBUFFLGS" is supplied with VM Systems Group Internal Tools. This macro is further described in the VM Systems Group Internal Tools macros section of this book.

The seven byte character area is reserved for future extensions. Bit UBUFIGNA is used to distinguish data passed that "looks" like an attribute but should be treated as data to be converted. This bit is only valid if UBUFHEX or UBUFDEC are set.

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

65 The menu number passed is not a page mode menu.

66 A specific row/column was specified with BUFENT greater than 1.

67 BUFCNT is less than or equal to zero.

68 Fewer parameters were passed than BUFCNT implies.

69 Row/column numbers passed are not within the menu.

78 An invalid attribute was passed to MPGINT or MPGD2S.

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

MPGINT - Creating a page mode menu

```
MPGINT      (number,retcode,flags,numlnes,linelen,savecnt,colcnt,
               coldefs,cmdline,numtop,numbot,T1,T1L,...Tn,TnL)
```

MPGINT (Initialize page mode menu) creates and loads a columnar menu tailored to the size of the terminal. This special form of menu, called a "page mode menu," is created dynamically by MPGINT (i.e. it is not created before running your application). No copy is created on DASD storage.

This style of menu has four component parts:

- An optional number of top title lines.
- A two dimensional section containing columns of fields in rows.
- An optional number of bottom title lines.
- An optional command input line with an optional command prefix or suffix area.

In general, this routine is passed the number and contents of top and bottom title lines, the number, lengths and optionally the initial attributes of each field in a columnar row, and flags used to determine how the menu is created.

When the menu is created, MPGINT returns the number of rows available for use by the columnar data - MPGINT creates a menu equal to the largest screen size supported by the terminal. The menu number is also returned - this is used to reference the menu in all other VM Systems Group Internal Tools subroutine calls.

Normally, MPGINT assigns top title lines field names of "TOP.1," "TOP.2," etc. - bottom title lines are named "BOT.1" and so on. The command line is named "CMDLINE" and the command prefix or suffix area is named "CMDPREF."

Starting from the top leftmost part of the menu, each field in the columnar region is assigned a named based on a "spreadsheet" type naming convention. Each row from top to bottom is assigned an ascending number and each column from left to right is assigned an ascending letter. For example, the first field in the first row is named "A.1," the second field in the first row is named "B.1," etc. The first field in the second row is named "A.2" and so on.

Some of these names can be overridden by passed parameters. Specific named fields can be created in the top and bottom titles.

A special type of page mode menu, called a "line mode menu," can also be created. This type of menu can have top and bottom titles and a command line, but has one large field where the columnar rows of fields would be. This style of menu is used for line by line output and is dealt with specially by "MPGD2S."

Page mode menus can be displayed in windows. There are several ways to do this:

1. If MPAGWIND is passed, a 16 character window name follows the flag bytes. This window is used to display this menu. If the window already exists, the page mode

menu is sized to fit the window. If the window doesn't exist, it is implicitly created based on the size data passed or implied by the MPGINT call.

2. If a default window name is set by MSETW, this window is used for this menu. Again, it is created if it doesn't already exist.
3. If MPAGIGNW is set, window support is ignored even if requested by the above two cases.

VM Systems Group Internal Tools provides several other subroutines specifically designed for page mode menus. These are used to move buffers of data to and from the fields in sequential rows. In addition, the entire columnar area can be cleared, specific fields can be updated, etc. See "MPGD2S," "MPGS2D," "MPGPOS," "MPGCLR" and "MPGSCR." With the exception of "MADD" and "MDEL" any VM Systems Group Internal Tools subroutine can be used to manipulate a menu created by MPGINT.

Parameters

number **(fullword binary)**
Returns the unique number by which the menu is referenced in subsequent VM Systems Group Internal Tools subroutine calls. The value is assigned by VM Systems Group Internal Tools. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flags **(binary length 4)**
Flags passed to VM Systems Group Internal Tools to determine the type and format of the page mode menu to be created. The bits in "flags" are defined below:

FLAG	VALUE	RESULT IF SET
MPAGNOCM	80000000	Don't create a command line.
MPAGCTOP	40000000	Place command line before columnar fields.
MPAGNPRE	10000000	Don't create a command line prefix/suffix area.
MPAGRPRE	08000000	Make command line suffix area.
MPAGCCOL	04000000	Center columns on the menu.
MPAGSPTL	01000000	Command line precedes last top title line.
MPAGNOCL	00040000	Create "line mode" menu (one big area).
MPAGSPCN	00020000	Specific starting row/column field names passed.
MPAGTATT	00010000	Title/bottom lines contain specific fields.
MPAGDEFS	00008000	Create menu using 3277 compatibility size.
MPAGWIND	00004000	A window name follows these flags.

MPAGIGNW	00002000	Ignore window support even if MPAGWIND is set.

The values defined above are in hexadecimal. These flags are passed to VM Systems Group Internal Tools. Undefined bits are reserved for future use and should be set to zero. To simplify the passing of these flags, the macro "MPAGINIT" is supplied with VM Systems Group Internal Tools. This macro is further described in the VM Systems Group Internal Tools macros section of this manual.

If the MPGWIND flag is set, these flags are followed by a window name. This window name is 16 characters long, left justified and blank filled.

numlnes **(fullword binary)**
Returns the number of rows assigned for columnar fields.

linelen **(fullword binary)**
This parameter is used to return additional information in the case of a programming error. Normally it returns the terminal line size. If the columnar definitions exceed the terminal's line size, this data can be used to determine how many fields will fit on a line.

If a title or bottom line has more fields than will fit on a line or if the count of fields present on a title line is incorrect a return code of 79 or 80 respectively is returned and this parameter returns the storage address of the offending title parameter pair.

If an invalid attribute is passed, a return code of 78 is returned and this parameter returns the storage address of the invalid attribute.

savecnt **(fullword binary)**
For "line mode" menus (bit MPAGNOCL set in "flags") this parameter sets the number of data lines passed by "MPGD2S" to be saved in an in storage queue. If the value passed is not positive it defaults to one. Subroutine "MPGSCR" can be used to position and display data within this queue. If more lines are passed by "MPGD2S" than fit on the queue, the oldest lines on the queue are purged. "MPGSSC" can be called to change this value.

colcnt **(fullword binary optionally followed by two additional fullwords)**
Passes the number of column definitions.

If "flags" has bit MPAGSPCN on, two additional fullwords follow:

1. Starting column "letter" (0=col A, 1=col B, etc.) The largest column number is 702 (A-Z, AA-ZZ).
2. Starting row "number" (0=row 1, 1=row 2, etc.) The largest row number is 99999 if using single letter column names, or 9999 if using double letter column names.

These numbers define the starting field name to assign to the columnar fields.

coldefs **(array of fullword binary values and optional attributes)**
Passes for each column defined in a row, the length of the column (not counting the attribute field). The number of entries is equal to the value contained in the first fullword of "colcnt." The data area of each columnar field is initialized to blanks. If a specific attribute is to

be passed to define this field, it follows the count value. If present, it must be in the following format:

Byte	Contents
0	SFE attribute byte (X'29')
1	Count of following pairs (X'04')
2	Color type (X'42')
3	Color value (X'F1' BLUE, X'F2' RED, X'F3' PINK, X'F4' GREEN, X'F5' TURQUOISE, X'F6' YELLOW, X'F7' WHITE, X'00' Default)
4	Highlighting type (X'41')
5	Highlighting value (X'00' None, X'F1' BLINK, X'F2' REVERSE, X'F3' UNDERSCORE)
6	Logical symbol set type (X'43')
7	Logical symbol set value (X'00' default, X'40-EF' logical symbol set)
8	Standard attribute type (X'C0')
9	Standard attribute value (a bit definition - see the section on 3270 attributes for more information)
10	Two bytes of filler (total 12 bytes)

To simplify the passing of attributes, the macro "MENUATT" is supplied with VM Systems Group Internal Tools. This macro defines the first ten bytes above. This macro is further described in the VM Systems Group Internal Tools macros section of this book.

If no attribute is passed for a columnar field it defaults to "PROT, DIM, SKIP, NOMDT" with default color, highlight and symbol set. The attribute and size of each columnar field is copied for each row used for columnar data.

cmdline (2 fullword binary values followed by an optional character string of length "cmdline+4")

The first fullword contains the number of contiguous lines to allocate to a command line. The command line is initialized to blanks (if the "MNULLS" subroutine is called, this will be set to nulls (binary zeros) at display time). The second fullword is the length of the command prefix/suffix area optionally followed by the command prefix/suffix data.

If bit MPAGNOCM is set this parameter is ignored. If bit MPAGNPRE is set, the second two parts of this parameter are ignored.

If no prefix string is passed VM Systems Group Internal Tools creates a prefix area containing the string "====>." A leading attribute cannot be passed with the prefix string. The command prefix/suffix field is defined as "PROT, DIM, SKIP" with default color, highlight and symbol set.

The command line is created with an attribute of "UNPROT, BRIGHT" with default color, highlight and symbol set.

numtop (fullword binary)

The number of top title lines the menu is to contain.

numbot (fullword binary)

The number of bottom lines the menu is to contain.

The following two parameters are repeated in pairs, one pair for each top title line followed by one pair for each bottom title line. There should a number of parameters equal to (2*"numtop"+"numbot").

T1...Tn **(character string of length "T1L...TnL")**
The initial contents of the title line for each top and bottom title line in descending row order. This data can contain an initial attribute in the format of the ten byte attribute passed in "coldefs." The macro "MENUATT" can be used to place attributes in the title lines.

If no attribute is passed, the default attribute is "PROT, BRIGHT, SKIP, NOMDT" with default color, highlighting and symbol set.

T1L...TnL **(fullword binary optionally followed by another fullword and an array of character length 8 elements)**
The first fullword contains the character length of the associated top or bottom title line including any leading passed attribute.

If bit MPAGTATT in "flags" is set, additional data must follow each count. The second fullword contains the number of fields minus one contained on the line (a line always contains at least one field - the count here is used to determine the size of the menu allocated in storage). This count is followed by the names to be assigned to each field in the title line. The field names are assigned in field order. A title containing three fields would have a field count of two and three field names. The field names in this array are used instead of "TOP.1" "BOT.1," etc. Fields are defined by attributes imbedded within the title line.

Return Codes

0 Menu created successfully.

1 Your terminal is not a 327x.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

62 Titles + command + bottom line count is greater than the number of lines on the screen.

63 Column lengths exceed terminal linesize.

64 First column number greater than "ZZ" and row greater than 5 digits (4 digits if column greater than "Z").

68 Fewer parameters were passed than BUFCNT implies.

78 An invalid attribute was passed to MPGINT or MPGD2S.

79 A title/bottom line passed to MPGINT containing multiple fields exceeds the screen's column size.

80 The count of fields in a title/bottom line is not the same as the actual line's field count.

700 Insufficient storage to get virtual screen control block(s).

701 Insufficient storage to get window control block(s).

703 A window with this name already exists.

706 A vertical size was passed without a horizontal size.

708 Insufficient storage to get virtual screen data array(s).

709 Window vertical size > virtual screen vertical size.

710 Window horizontal size > virtual screen horizontal size.

711 Insufficient storage to get window data array(s).

MPGPOS - Position next line pointer

```
MPGPOS        (number,retcode,line)
```

MPGPOS (position next line pointer) sets the location of the line pointer to the passed value. The value of "line" must be between zero and "numlnes"-1 (returned by "MPGINT").

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

line **(fullword binary)**
The row to place data at the next MPGD2S request.

Return Codes

0 Row reset successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

65 The menu number passed is not a page mode menu.

69 Row/column numbers passed are not within the menu.

MPGSCR - Scroll within a line mode menu

```
MPGSCR        (number,retcode,linenumb)
```

MPGSCR allows you to scroll within the saved lines associated with a line mode menu. This subroutine positions to the line "linenumb" in the saved line queue, fills the menu with lines beginning at this line number, then displays the menu. Unused lines are cleared to nulls (binary zeros).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

linenumb **(fullword binary)**
The number in the saved line queue which is placed in the first line of the displayed line area. This number should be a value from zero to "n-1" where "n" is the current number of lines in the saved line queue. The number of lines currently saved in the queue is returned on calls to "MPGD2S."

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

65 The menu number passed is not a page mode menu.

73 Line number in a line mode menu exceeds the number of lines saved.

MPGSSC - Set saved line count

```
MPGSSC        (number,retcode,savecnt)
```

MPGSSC allows you to change the saved line count value originally passed by the "MPGINT" parameter "savecnt." If the value is greater than that passed by "MPGINT" the internal value is updated. If the value is less than that passed by "MPGINT" and there are currently more lines saved than the new maximum value, the oldest saved lines are freed until the number of saved lines equals the maximum line count.

This subroutine can only be called referencing a line mode menu.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

savecnt **(fullword binary)**
This parameter sets the number of data lines passed by "MPGD2S" to be saved in an in storage queue. If the value passed is not positive it defaults to one. Subroutine "MPGSCR" can be used to position and display data within this queue. If more lines are passed by "MPGD2S" than fit on the queue, the oldest lines on the queue are purged.

Return Codes

0 Saved line count set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

65 The menu number passed is not a page mode menu.

MPGS2D - Move a row of data from a page mode menu

```
MPGS2D      (number,retcode,flags,key,linedefs,bufcnt,cmdbuf,
               cmdbufl,cmdimpl,BUF1,BUF1L,BUF1RL,
               ...BUFn,BUFnL,BUFnRL)
            (number,retcode,flags,key,linedefs,bufcnt,
               BUF1,BUF1L,BUF1RL,...BUFn,BUFnL,BUFnRL)
```

MPGS2D is used to move a row of data from the last user input in a horizontal row of fields in a page mode menu to your program buffer(s).

Normally the next available row in the menu is used to return the input. If the bottom row was passed at the last call the top row is passed at this call. It is also possible to retrieve input from a specific field by row and column if desired.

The key pressed at the last input is returned. Also, depending on the flags passed, the contents of the command line may be returned.

This subroutine uses the same next line pointer as "MPGD2S."

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flags **(binary length 2)**
Flags passed by your program to control the action of this subroutine. The following describes the defined flags:

FLAG	VALUE	RESULT IF SET
MPAGDRFL	4000	Command line parameters passed.
MPAGCRSP	1000	Input data from a specific row/column.

The values defined above are in hexadecimal. These flags may be combined. To simplify the passing of these flags, the macro "MPAGFLGS" is supplied with VM Systems Group Internal Tools. This macro is further described in the VM Systems Group Internal Tools macros section of this book.

key **(character length 8)**
The key the user pressed to signal the end of input. This can be "ENTER," "PF01-PF24," "PA1-PA3," "TESTREQ," "SCANNER," "CLEAR," "NOAID," "LGHTPEN," "CARDRDR," "SMSG," "IUCV," "TIMER," "RDR" or "DEVICE."

linedefs **(2 fullwords binary)**
Normally ignored.

If bit MPAGCRSP in "flags" is set this area passes data used to determine the specific field in the row/column array on the menu from which the input data will transferred. The first fullword contains the column number (0=col A, 1=col B,), the second fullword contains the row number (0=row 1, 1=row 2,).

bufcnt **(fullword binary)**
The number of buffers (columns) passed.

cmdbuf **(fullword binary)**
If command line input is received, this contains the data returned in the command line field of the menu. This parameter is passed only if bit MPAGDRFL in "flags" is set. If MPAGDRFL is set, and no command line input exists, MPGS2D completes but returns a return code of ten.

cmdbufl **(fullword binary)**
This field contains the length of "cmdbuf." This parameter is passed only if bit MPAGDRFL in "flags" is set.

cmdimpl **(fullword binary)**
This field returns the actual length of data returned into "cmdbuf." This parameter is passed only if bit MPAGDRFL in "flags" is set.

The following parameters are passed in groups of three, one for each field of input to be retrieved. The first buffer gets input for the leftmost field on the row. Each following buffer gets data from the field to the right of the preceding field.

BUF1...BUFn **(binary data of varying length)**
The buffer to receive input from a field on this row. If this buffer is smaller than the input data, the input data is truncated.

BUF1L...BUFnL (fullword binary)
The fullword contains the length of the buffer pointed to by "BUFn."

BUF1RL...BUFnRL (fullword binary)
The fullword contains the actual length of data moved into "BUFn." If there was no input for this field, MPGS2D will fill this parameter with a minus one. If the user pressed "ERASE EOF" in this field this parameter is set to zero.

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

10 Command line data requested but none was input.

65 The menu number passed is not a page mode menu.

66 A specific row/column was specified with BUFENT greater than 1.

67 BUFCNT is less than or equal to zero.

68 Fewer parameters were passed than BUFCNT implies.

69 Row/column numbers passed are not within the menu.

77 MPGS2D cannot return data from a line mode menu.

MPLIST - Get program command line

```
MPLIST        (number,retcode,buffer,length,rlen)
```

MPLIST returns the command line used to invoke your application program. The list is moved in eight-byte parts with a blank separating each piece (consequently the buffer size should be a multiple of nine bytes).

The list returned is identical to the tokenized PLIST passed to your application by CMS in general purpose register 1 assuming your buffer is large enough. The CMS "fence" (eight bytes of X'FF' is not transferred, you can determine the last parameter by the value in "rlen."

Parameters

number **(fullword binary)**
Ignored.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(character length "length")**
The buffer to receive the parameter list.

length **(fullword binary)**
Contains the length of "buffer." If the command line is longer than "length," it is truncated.

rlen **(fullword binary)**
Contains the length of data moved into "buffer." If no command line was present "rlen" will be zero.

Return Codes

0 Subroutine completed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MPOINT - Return addresses of VM Systems Group Internal Tools control blocks

```
MPOINT        (number,retcode,addr-rec1,addr-rec2,addr-ctrl)
```

MPOINT is used for special applications to return the VM Systems Group Internal Tools pointers to its internal control blocks for a particular menu. The first two blocks can be relocated during "MADD" calls; therefore, if you use this routine be sure to re-call it to re-initialize your pointers after each "MADD" call.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

addr_rec1 **(fullword pointer)**
Returns the address of the menu control blocks (menu file record 1).

addr_rec2 **(fullword pointer)**
Returns the address of the menu output area (menu file record 2).

addr_ctrl **(fullword pointer)**
Returns the address of the VM Systems Group Internal Tools MENUBLOK for this menu.

Return Codes

0 Addresses passed to the caller.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MPOSVS - Position virtual screen on terminal

```
MPOSVS      (number,retcode,vpos,hpos)
            (number,retcode,vpos,hpos,vsize,hsize)
```

MPOSVS is used to position the virtual screen on the terminal.

When a virtual screen is first defined, its upper left character is displayed on the upper left character of the terminal. MPOSVS can be used to scroll through a virtual screen that is bigger than the terminal.

If vsize and hsize aren't specified, then as much of the virtual screen as will fit on the terminal is displayed. If vsize and hsize are specified, only their amount of rows and columns are displayed. Vsize and hsize cannot be bigger than the terminal's largest row and column size.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

vpos **(fullword binary)**
The row position of the character on the virtual screen displayed on the upper left corner of the terminal. The uppermost row of the virtual screen is row zero.

hpos **(fullword binary)**
The column position of the character on the virtual screen displayed on the upper left corner of the terminal. The uppermost column of the virtual screen is column zero.

vsize **(fullword binary)**
The number of rows displayed on the terminal. This number cannot exceed the terminal's largest row count. If zero is passed, this parameter is ignored.

hsize **(fullword binary)**
The number of columns displayed on the terminal. This number cannot exceed the terminal's largest column count. If zero is passed, this parameter is ignored.

Return Codes

0 The virtual screen is positioned.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

705 The virtual screen does not exist.

712 The vertical position is too large.

713 The horizontal position is too large.

718 Negitive number(s) passed to MPOSW or MPOSVS.

MPOSW - Position and size a window's viewport on virtual screen

```
MPOSW     (number,retcode,wname,vpos,hpos)
          (number,retcode,wname,vpos,hpos,voff,hoff)
          (number,retcode,wname,vpos,hpos,voff,hoff,vsize,hsize)
```

MPOSW is used to position and size a window's viewport on the virtual screen.

When a window is first defined, it is fully displayed starting in the upper left hand corner of the virtual screen. MPOSW is used to change this position, and also to change the amount and size of data displayed in this viewport.

A viewport is a portion of a window displayed on a virtual screen. Vpos and hpos specify the position that the upper left character of the viewport's data area (not its borders, if any) holds on the virtual screen.

If voff and hoff aren't specified, the upper left character of the window is displayed in the upper left character of the viewport.

If vsize and hsize aren't specified, then as much of the window as will fit in the virtual screen is displayed. If vsize and hsize are specified, only their amount of rows and columns are displayed. Vsize and hsize define the size of the viewport. Vsize and hsize cannot be bigger than the remaining size of the virtual screen (based on where the viewport is positioned). They also cannot be bigger than the window size.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

vpos **(fullword binary)**
The row position of the character on the viewport displayed on the upper left corner of the virtual screen. The uppermost row of the viewport is row zero.

hpos **(fullword binary)**
The column position of the character on the viewport displayed on the upper left corner of the virtual screen. The uppermost column of the viewport is column zero.

voff **(fullword binary)**
The row position of the character of the window displayed on the upper left corner of the viewport. The uppermost row of the window is row zero.

hoff **(fullword binary)**
The column position of the character of the window displayed on the upper left corner of the viewport. The uppermost column of the window is column zero.

vsize **(fullword binary)**
The number of rows displayed in the viewport. This number cannot exceed the remaining row count of the virtual screen or the row size of the window. If zero is passed, this parameter is ignored.

hsize **(fullword binary)**
The number of columns displayed in the viewport. This number cannot exceed the remaining column count of the virtual screen or the column size of the window. If zero is passed, this parameter is ignored.

Return Codes

0 The viewport is positioned and sized.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

705 The virtual screen does not exist.

712 The vertical position is too large.

713 The horizontal position is too large.

718 Negitive number(s) passed to MPOSW or MPOSVS.

MPRINT - Print a menu

```
MPRINT      (number,retcode)
            (number,retcode,pflags)
            (number,retcode,pflags,pfname)
            (number,retcode,pflags,pfname,overst)
```

MPRINT prints the current menu on the VM virtual printer or to a DASD file. The menu, including any current user input is printed, normally within a numbered box. A header line displays the menu name, the date and time of print and the number of lines and columns in the menu.

If "pflags" is zero or if not passed, the output goes to the virtual printer, within the formatted box - DARK fields are not printed. If "overst" is not passed, it defaults to two.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

pflags **(fullword binary)**
Flags used to control the request. The following values control the subroutine output results:

FLAG	VALUE	RESULT IF SET
PRTFILE	1	Output goes to a DASD file.
PRTSCRPT	2	Output is in SCRIPT/VS format.
PRTNOH	8	Output without a box or header line (emulate MPRNOH).
PRTDARK	16	Output the contents of DARK fields.
PRTFNAMP	32	A specific FILENAME is present.
PRTNOINP	64	Output without including the last user input.

The values above are given in decimal. These flags may be combined. To simplify the passing of these flags, the macro "MPRTFLGS" is supplied with VM Systems Group Internal Tools. This macro is further described in the VM Systems Group Internal Tools macros section of this book.

pfname **(character length 8)**
The FILENAME to which the output is sent if bits PRTFILE and PRTFNAMP are set. If only PRTFILE is set the menu name is used as the default name. The output FILETYPE is "LISTING." If PRTSCRPT is set, the FILETYPE is "SCRIPT."

overst **(fullword binary)**
The number of times lines are overprinted to highlight BRIGHT fields in the output. If "overst" is zero no overprinted lines are output. This value is ignored if bit PRTSCRPT is set.

Return Codes

0 Menu printed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

5xx Error in CMS FSWRITE (xx is the FSWRITE code).

502 Invalid output buffer address.

503 Permanent I/O error occurred.

508 Attempt to write an incorrect length record.

512 Attempt to write on a read only disk.

513 The disk is full.

522 Virtual storage capacity exceeded.

525 Insufficient free storage.

MPRLNE - Print a line of data

```
MPRLNE        (number,retcode,pflags,pfname,buffer,buflen)
```

MPRLNE prints the contents of "buffer" to the same output dataset/printer as "MPRINT" and "MPRNOH." It can be used to incorporate program specific comments, etc. into the menu output log.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

pflags **(fullword binary)**
Flags used to control the request. The following values control the subroutine output results:

FLAG	VALUE	RESULT IF SET
PRTFILE	1	Output goes to a DASD file.
PRTSCRPT	2	Output is in SCRIPT/VS format.
PRTFNAMP	32	A specific FILENAME is present.

The values above are given in decimal. These flags may be combined. To simplify the passing of these flags, the macro "MPRTFLGS" is supplied with VM Systems Group Internal Tools. This macro is further described in the VM Systems Group Internal Tools macros section of this book.

pfname **(character length 8)**
The FILENAME to which the output is sent if bits PRTFILE and PRTFNAMP are set. If only PRTFILE is set the menu name is used as the default name. The output FILETYPE is "LISTING." If PRTSCRPT is set, the FILETYPE is "SCRIPT."

buffer **(character length "buflen")**
The data to be output. An ANSI carriage control character should be the first character in your output buffer unless you set PRTSCRPT.

buflen **(fullword binary)**
The length of "buffer" in bytes.

Return Codes

0 Data printed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

75 MPRLNE - the line cannot be printed (too long, etc.).

5xx Error in CMS FSWRITE (xx is the FSWRITE code).

502 Invalid output buffer address.

503 Permanent I/O error occurred.

508 Attempt to write an incorrect length record.

512 Attempt to write on a read only disk.

513 The disk is full.

522 Virtual storage capacity exceeded.

525 Insufficient free storage.

MPRNOH - Print a menu without a header

```
MPRNOH      (number,retcode)
            (number,retcode,pflags)
            (number,retcode,pflags,pfname)
            (number,retcode,pflags,pfname,overst)
```

MPRNOH prints the current menu on the VM virtual printer or to a DASD file. Only the menu is output; no header line or numbered box is printed.

If "pflags" is not passed or is zero, output goes to the virtual printer - DARK fields are not printed. If "overst" is not passed, it defaults to two.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

pflags **(fullword binary)**
Flags used to control the request. The following values control the subroutine output results:

FLAG	VALUE	RESULT IF SET
PRTFILE	1	Output goes to a DASD file.
PRTSCRPT	2	Output is in SCRIPT/VS format.
PRTDARK	16	Output the contents of DARK fields.
PRTFNAMP	32	A specific FILENAME is present.
PRTNOINP	64	Output without including the last user input.

The values above are given in decimal. These flags may be combined. If "pflags" is passed this subroutine functions identically to MPRINT except that the PRTNOH flag (see MPRINT) is always assumed to be set. To simplify the passing of these flags, the macro "MPRTFLGS" is supplied with VM Systems Group Internal Tools. This macro is further described in the VM Systems Group Internal Tools macros section of this book.

pfname **(character length 8)**
The FILENAME to which the output is sent if bits PRTFILE and PRTFNAMP are set. If only PRTFILE is set the menu name is used as the default name. The output FILETYPE is "LISTING." If PRTSCRPT is set, the FILETYPE is "SCRIPT."

overst **(fullword binary)**
The number of times overprinted lines are used to highlight BRIGHT fields in the output. If "overst" is zero no overprinted lines are output. This value is ignored if bit PRTSCRPT is set.

Return Codes

0 Menu printed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

5xx Error in CMS FSWRITE (xx is the FSWRITE code).

502 Invalid output buffer address.

503 Permanent I/O error occurred.

508 Attempt to write an incorrect length record.

512 Attempt to write on a read only disk.

513 The disk is full.

522 Virtual storage capacity exceeded.

525 Insufficient free storage.

MPROTW - Sets single or multiple window input mode

```
MPROTW        (number,retcode,flag)
```

MPROTW is used to set whether windows other than the highest priority window have their input fields protected (inhibited from user input).

If flag is zero, then all displayed windows that have unprotected input fields can be used by a user to input data. That is, it is possible to enter data on more than one window at once.

If flag is non-zero, then only the highest priority (top) window's input fields are left unprotected; all other window's input fields are protected from user input. Thus, only the topmost menu can have data entered.

You might want to use window protection in an application where you are displaying pop-up windows. While the pop-up window appears, the lower windows cannot be modified.

This subroutine only sets an VM Systems Group Internal Tools flag. Protection takes effect at the next terminal display.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Determines whether window protection is used. If flag is zero protection is not used. If non-zero, protection is used.

Return Codes

0 Window protection is set.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

705 The virtual screen does not exist.

MPSLD - Load a Programmable Symbol Set

```
MPSLD       (number,retcode,setnum,setname)
            (number,retcode,setnum,setaddr,setlen)
```

MPSLD loads a programmable symbol set into the terminal for use by subsequent menus. The symbol set may either reside on disk or may be passed from an in-storage copy. Symbol sets on disk may be created/edited using the PSEDIT symbol set editor.

Several different menus, and/or several different displays of the same menu may be performed without reloading the symbol set. Symbol sets should be reloaded if there is any possibility that they were destroyed between VM Systems Group Internal Tools uses (for example, if a user disconnects and reconnects to a different terminal, or if another full screen program was called between menu displays). Attempts to use symbol sets that are not loaded are changed to use the default symbol set.

You assign a logical number to the symbol set. This number is used in the menu and in MSEXTA, MADDSA, MCHGSA and/or MDELSA calls to display particular menu fields using the appropriate loaded symbol set. This number should be between X'40' and X'EF'. If the terminal already contains a symbol set with the same logical number, it is overlaid by the newer request.

If MPSLD is issued on a terminal without loadable symbol set capabilities, the request is ignored. The MCTYPE subroutine can be used to determine the number of symbol sets your terminal has and their capabilities.

Parameters

number **(fullword binary)**
Ignored. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

setnum **(fullword)**
The logical symbol set number to be assigned to this symbol set. The number is contained within the rightmost byte of the fullword and must be a value between X'40' and X'EF'.

setname **(character length 8)**
The name of the PSLOAD file containing the symbol set data.

setaddr **(fullword pointer)**
If the symbol set is contained in storage, this parameter points to a fullword containing its address. An in-storage copy of a symbol set must be identical in format to a PSLOAD file.

setlen **(fullword)**
If the symbol set is contained in storage, this parameter contains its length.

Return Codes

0 Symbol set loaded or ignored (if hardware not present).

1 Your terminal is not a 327x.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

37 The programmable symbol set number passed is invalid.

60 Unable to load the PSLOAD file (not found, I/O error, etc.).

61 The PSLOAD file is improperly formatted.

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

MPURGE - Purge (release) a menu

```
MPURGE      (number,retcode)
            (number,retcode,close)
```

MPURGE removes a menu from storage. Any further references to the menu are rejected with a return code of four.

If any calls to "MPRINT," "MPRNOH" or "MPRLNE" caused output on the virtual printer, the virtual printer is closed if requested (see the "close" parameter below). Closing a virtual printer causes all previous output to be placed into a single spool file.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

close **(fullword binary)**
Specifies whether to close the virtual printer if "MPRINT," "MPRNOH" or "MPRLNE" were used by this application. If not passed or if the value passed is zero no close is done. If the value passed is non-zero, the printer is closed if necessary.

Return Codes

0 Menu purged.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MPURVS - Purge virtual screen

```
MPURVS        (number,retcode)
```

MPURVS is used to purge an VM Systems Group Internal Tools virtual screen, together with any windows associated with it.

This call can be used to quickly purge all existing windows and the virtual screen at the end of an application.

Note that MPURVS can be implicitly called by the MEXIT subroutine.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 The virtual screen and associated windows are purged.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

705 The virtual screen does not exist.

MPURW - Purges a window

```
MPURW          (number,retcode,wname)
```

MPURW is used to purge an VM Systems Group Internal Tools window.

MPURGE can implicitly purge a window, however, MPURW does not implicitly purge a menu associated with it.

Note that MPURVS can be implicitly called by the MEXIT subroutine.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

Return Codes

0 The window is purged.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

704 A window with this name does not exist.

705 The virtual screen does not exist.

MQACTW - Returns the last active window and menu

```
MQACTW        (number,retcode,wname)
```

MQACTW is used to return the window name and potentially the menu number of the window the cursor was within at the last terminal input.

This routine allows you to determine which window the user considers to be active. Armed with this information, you can choose to only process the request the user is most interested in.

Wname is used to return the name of this active window. If this window is closely associated with an VM Systems Group Internal Tools menu, the menu number is returned in number.

By convention, VM Systems Group Internal Tools (and for that matter, most other software) returns the key pressed and the cursor position into the window containing the cursor at the time the user pressed an interrupt producing key.

If no cursor position exists (say, for example, a PA key was pressed), then the highest priority active window is considered to contain the cursor.

Parameters

number **(two binary fullwords)**
Returns the menu number of the menu associated with wname. If no menu is associated with this window, this field is set to zero. If there was no active window, this field is not modified, and a return code of 715 is set. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
Returns the name of the active window. If there was no active window, this field is not modified, and a return code of 715 is set.

Return Codes

0 Active window and menu were returned.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
705 The virtual screen does not exist.
715 There was no window active at the last user input.

MQATTR - Query a passed attribute character

```
MQATTR      (number,retcode,byte,prot)
            (number,retcode,byte,prot,bright)
            (number,retcode,byte,prot,bright,skip)
            (number,retcode,byte,prot,bright,skip,mdt)
```

MQATTR returns character representations of the attributes of a byte passed in parm "byte." This routine is not normally used in user applications; it is provided for special purposes only.

Parameters

number **(fullword binary)**
Ignored.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

byte **(byte length 1)**
The attribute byte to be queried.

prot **(character length 7)**
The protection attribute of the byte. "Prot" may be "PROT" (protected), "UNPROT" (unprotected, any input) or "NUMERIC" (unprotected, numeric input).

bright **(character length 7)**
The intensity attribute of the byte. "Bright" may be "BRIGHT" (intensified), "DIM" (normal display), "DARK" (non-display), or "LGHTPEN" (normal display, selector pen detectable).

skip **(character length 6)**
Returns "SKIP" if the byte has the SKIP attribute, "NOSKIP" otherwise.

mdt **(character length 6)**
Returns "MDT" if the byte's Modified Data Tag is set, "NOMDT" otherwise.

Return Codes

0 Data returned.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MQEXTA - Query the extended attributes of a field

```
MQEXTA        (number,retcode,fieldname,color,hi,ps)
```

MQEXTA returns character representations of the extended attributes of a field in the menu. The attributes returned are those currently in use; previous "MSEXTA" calls alter the values returned by MQEXTA. MQEXTA can be called even if the terminal does not support extended attributes. A non-zero return code will only be returned if the menu itself does not contain extended attribute data.

To query standard attribute values, use subroutine "MFQRY."

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field to be queried.

color **(character length 10)**
The color attribute of the field. The color may be "DEFAULT" (default color for the intensity and protection of the field), "BLUE," "PINK," "RED," "GREEN," "TURQUOISE," "YELLOW" or "WHITE."

hi **(character length 10)**
The highlighting attribute of the field. "Hi" may be "DEFAULT" (no highlighting), "BLINK," "REVERSE," or "UNDERSCORE."

ps **(fullword)**
Returns the logical symbol set number for the field. The value is placed in the rightmost byte of the fullword. A value of zero denotes the standard symbol set.

Return Codes

0 Data returned.

2 MLOAD, MLOADX - The menu file you are attempting to load is not formatted properly.

2 MQEXTP - The attribute data passed is improperly formatted.

2 MFQRY, MFSET, MQEXTA, MSEXTA The field attribute could not be found.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

35 The menu does not contain extended attribute data.

MQEXTP - Query the extended attributes of a passed string

```
MQEXTP        (number,retcode,string,color,hi,ps)
```

MQEXTP returns character representations of the extended attributes of a passed extended attribute string. This routine is not normally used for user applications; it is provided for special purposes only.

Parameters

number **(fullword binary)**
Ignored.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

string **(byte length 10)**
The string to be queried. It should be in the following format:

Byte	Contents
0	SFE attribute byte (X'29')
1	Count of following pairs (X'04')
2	Color type (X'42')
3	Color value
4	Highlighting type (X'41')
5	Highlighting value
6	Logical symbol set type (X'43')
7	Logical symbol set value
8	Standard attribute type (X'C0')
9	Standard attribute value

color **(character length 10)**
Returns the color attribute of the string. The color may be "DEFAULT" (default color for the intensity and protection of the field), "BLUE," "PINK," "RED," "GREEN," "TURQUOISE," "YELLOW" or "WHITE."

hi **(character length 10)**
Returns the highlighting attribute of the string. "Hi" may be "DEFAULT" (no highlighting), "BLINK," "REVERSE," or "UNDERSCORE."

ps **(fullword)**
Returns the logical symbol set number for the string. The value is placed in the rightmost byte of the fullword. A value of zero denotes the standard symbol set.

Return Codes

0 Data returned.

2 MLOAD, MLOADX - The menu file you are attempting to load is not formatted properly.

2 MQEXTP - The attribute data passed is improperly formatted.

2 MFQRY, MFSET, MQEXTA, MSEXTA The field attribute could not be found.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MQRSF - Return query partition structured field data

```
MQRSF       (number,retcode,buffer)
            (number,retcode,buffer,length)
```

MQRSF returns the query partition structured field data. Up to 1024 bytes of data are returned. If the terminal does not support query structured field, a return code of 70 is returned.

The format of this data can be found in the "IBM 3270 Information Display System Data Stream Programmer's Reference," number GA23-0059.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

buffer **(binary length 1024)**
The query structured field data is moved to this buffer. VM Systems Group Internal Tools assumes that this buffer is 1024 bytes long.

length **(fullword binary)**
Returns the actual length of data moved into "buffer."

Return Codes

0 Subroutine completed successfully.

1 Your terminal is not a 327x.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

70 No terminal query data available (MQRSF).

MQRYAI - Query asynchronous terminal interrupt detection

```
MQRYAI        (number,retcode)
```

MQRYAI is called to determine whether an asynchronous terminal interrupt is pending. If pending, the return code is set to "76." Alternately, the byte passed to "MSETAI" in "flagbyte" can be periodically inspected. There is no facility to transfer control to a point in an application at the time of the interrupt. You must occasionally sample the flagbyte or call "MQRYAI" to see if an interrupt is pending.

Asynchronous terminal interrupt detection must be enabled by "MSETAI."

When an interrupt is pending, the data entered can be retrieved by calling "MRDSPR" using the menu number currently displayed. The data associated with the interrupt is not read until "MRDSPR" is called. If you do not care about the data entered with the interrupt, this call does not have to be made.

You must call "MCLRAI" when you no longer wish to trap asynchronous terminal interrupts. If you exit your application without clearing asynchronous interrupt trapping, unpredictable results can occur.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

76 MQRYAI - user data is pending at the terminal.

MQRYW - Returns information about a window

```
MQRYW        (number,retcode,wname,rs,cs,vp,hp,fr,fc,nr,nc,flags,mnumb)
```

MQRYW is used to return information about an existing window.

This routine can be used to develop general purpose routines that handle windows generated by foreign routines.

This routine also allows you to save and restore window characteristics, or display these values so a user could interactively modify them.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

rs **(fullword binary)**
Returns the number of rows in the window.

cs **(fullword binary)**
Returns the number of columns in the window.

vp **(fullword binary)**
Returns the viewport's vertical position on the virtual screen.

hp **(fullword binary)**
Returns the viewport's horizontal position on the virtual screen.

fr **(fullword binary)**
Returns the first row of the window displayed in the viewport.

fc **(fullword binary)**
Returns the first column of the window displayed in the viewport.

nr **(fullword binary)**
Returns the number of rows of the window displayed in the viewport.

nc **(fullword binary)**
Returns the number of columns of the window displayed in the viewport.

flags **(byte)**
Returns how the window's viewport is displayed. The following bits are currently defined:

WINVBOR X'80' Display vertical borders if possible
WINHBOR X'40' Display horizontal borders if possible
WINTITLE X'20' Display the menu name on the top border if possible
WINACTIV X'01' This viewport is active, that is, it is visible

mnumb **(fullword binary)**
Returns the number of the menu associated with this window, if any. If no menu is associated with this window, zero is returned.

Return Codes

0 Information about the window is returned.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
704 A window with this name does not exist.
705 The virtual screen does not exist.

MRDSPC - Get input from a menu that is currently displayed

```
MRDSPC      (number,retcode,key)
            (number,retcode,key,curoff)
            (number,retcode,key,curline,curcol)
```

MRDSPC gets additional input from a currently displayed menu. It is used for special cases only (for example, to modify parts of a menu, or to ignore previous input). MRDSPC differs from "MRDSPR" in that the alarm is sounded if MALARM was set and the cursor position is updated.

If this menu is associated with a window, then the virtual screen image is updated on the terminal. This window's viewport and any other filled, active windows are displayed. If the user leaves the cursor outside of this viewport, the key pressed will be returned as "NOAID." You can call MQACTW to determine which viewport contains the cursor and the key pressed.

If you change terminals in the middle of an application, unpredictable results can occur, however, if running a window application, automatic reconfiguration should take place.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine. If using the multiple terminal facility, the second fullword of "number" should be filled with the logical terminal number of the terminal you wish to address.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

key **(character length 8)**
The key the user pressed to signal the end of input. This can be "ENTER," "PF01-PF36," "PA1-PA3," "TESTREQ," "SCANNER," "CLEAR," "NOAID," "LGHTPEN," "CARDRDR," "SMSG," "IUCV," "TIMER," "RDR" or "DEVICE."

curoff **(fullword binary)**
Specifies the cursor position as an absolute offset from the upper left corner of the menu where that corner is position zero.

curline **(fullword binary)**
Specifies the cursor line position as an offset from the top line of the menu where the top line is line one.

curcol **(fullword binary)**
Specifies the cursor column position as an offset from the left side of the menu where the left side is column one.

Return Codes

0 Menu displayed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

11 The menu is too large for your terminal's screen size.

13 The last user input contained no cursor position (a PA or the CLEAR key was pressed).

48 The menu to be redisplayed is not the one last displayed.

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MRDSPR - Get input from a menu that is currently displayed

```
MRDSPR      (number,retcode,key)
            (number,retcode,key,curoff)
            (number,retcode,key,curline,curcol)
```

MRDSPR gets additional input from a currently displayed menu. It is used for special cases only (for example, to modify parts of a menu, or to ignore previous input).

If this menu is associated with a window, then the virtual screen image is updated on the terminal. This window's viewport and any other filled, active windows are displayed. If the user leaves the cursor outside of this viewport, the key pressed will be returned as "NOAID." You can call MQACTW to determine which viewport contains the cursor and the key pressed.

If you change terminals in the middle of an application, unpredictable results can occur, however, if running a window application, automatic reconfiguration should take place.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine. If using the multiple terminal facility, the second fullword of "number" should be filled with the logical terminal number of the terminal you wish to address.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

key **(character length 8)**
The key the user pressed to signal the end of input. This can be "ENTER," "PF01-PF36," "PA1-PA3," "TESTREQ," "SCANNER," "CLEAR," "NOAID," "LGHTPEN," "CARDRDR," "SMSG," "IUCV," "TIMER," "RDR" or "DEVICE."

curoff **(fullword binary)**
Specifies the cursor position as an absolute offset from the upper left corner of the menu where that corner is position zero.

curline **(fullword binary)**
Specifies the cursor line position as an offset from the top line of the menu where the top line is line one.

curcol **(fullword binary)**
Specifies the cursor column position as an offset from the left side of the menu where the left side is column one.

Return Codes

0 Menu displayed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

11 The menu is too large for your terminal's screen size.

13 The last user input contained no cursor position (a PA or the CLEAR key was pressed).

48 The menu to be redisplayed is not the one last displayed.

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MRELSC - Releases a VM Systems Group Internal Tools Multiple Terminal Facility session

```
MRELSC        (number,retcode)
```

MRELSC is used to release (end)a VM Systems Group Internal Tools Multiple Terminal Facility session.

This subroutine is used to terminate a MTF session and should be used on every MTF logical terminal defined before your application ends.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. The second fullword is the logical terminal number for the Multiple Terminal Facility logical terminal. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 A VM Systems Group Internal Tools MTF logical terminal is released.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

95 Logical device number passed does not exist.

96 Subroutine only supported under XMENU 2 MTF.

MREMOT - Set VM Systems Group Internal Tools output controls

```
MREMOT        (number,retcode,remote,optimize,apltext)
```

MREMOT sets or resets flags which determine how VM Systems Group Internal Tools performs output optimization and data conversion.

"Remote" determines whether only changed fields are transmitted to the terminal (rather than the whole menu each output). If non-zero (the default), only changed fields are transmitted. If zero, the entire menu is transmitted each output.

"Optimize" determines whether output data stream optimization is performed. If non-zero (the default), output optimization is done. This includes compressing of repeating characters, compression of default attribute values, etc. If zero, no optimization is performed.

Each of these flags if set reduce I/O overhead at the expense of CPU utilization.

"Apltext" determines whether the input and output data streams are checked for APL or TEXT characters. If non-zero (the default), checking is done. If zero, no checking is done. Normally, input and output APL/TEXT checking is based on the settings of CP TERM TEXT and TERM APL. Input checking uses APL if both TERM TEXT and TERM APL are OFF and a compound character is detected in the input stream.

If you are sure that your menus will not have any APL or TEXT characters transferred, you can save some CPU time by turning APL/TEXT checking off.

These flags are global flags - they affect all VM Systems Group Internal Tools output.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

remote **(fullword binary)**
If a value of zero is passed, the entire menu is transmitted to the terminal each output. If non-zero, only changed fields are transmitted.

optimize **(fullword binary)**
If a value of zero is passed, no output data optimization is performed. If non-zero, output optimization is performed.

apltext **(fullword binary)**
If a value of zero is passed, input and output data streams are not checked for APL or TEXT characters. If non-zero, standard checking is performed.

Return Codes

0 Flags set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MRENT - Initialize read-only VM Systems Group Internal Tools subroutines

```
MRENT        (number,retcode)
```

MRENT is called to initialize the VM Systems Group Internal Tools subroutines to be used as a part of read/only code (for example, a VM shared segment). MRENT should be called before any other VM Systems Group Internal Tools routine. MRENT copies the read/write data from VM Systems Group Internal Tools into the buffer pointed to by "number." All subsequent calls to VM Systems Group Internal Tools subroutines (except for data conversion and some other general purpose routines) must have this buffer pointed to by the "number" parameter. If not passed by a subroutine (or if data within it is overlaid in error) the VM Systems Group Internal Tools subroutine library will use its internal data area instead (and thus will no longer be read-only).

The first twenty (20) fullwords of this buffer are reserved so that the active menu number can be passed (and for other user defined needs). This way, both read/write and read/only calls to VM Systems Group Internal Tools have the same calling interface.

Parameters

number **(4096 byte buffer, doubleword aligned)**
Pass the address of a buffer for VM Systems Group Internal Tools read/write storage data. All subsequent calls to VM Systems Group Internal Tools should have as their first parameter the buffer passed to this routine. If this buffer was allocated from free storage your program may be included in a shared segment (provided it also follows the other rules of shared segments). Some performance enhancement may result if this buffer is on a page boundary. When this routine is called, up to 4096 bytes of data will be moved to the buffer regardless of its true size. It is the responsibility of your program to allocate this buffer.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Data moved successfully.

81 The buffer passed to MRENT is not on a doubleword boundary.

MRESET - Reset a terminal's characteristics

```
MRESET        (number,retcode)
```

MRESET is used to explicitly reset a terminal's characteristics.

This subroutine is used to explicitly request that VM Systems Group Internal Tools refetch the terminal's characteristics.

You might use this routine if you are using native, full screen I/O routines, and suspect that a user may have switched terminals in the middle of your application.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If you are using the Multiple Terminal Facility, the second fullword is the logical terminal number for the Multiple Terminal Facility logical terminal. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 A VM Systems Group Internal Tools MTF logical terminal is released.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

95 Logical device number passed does not exist.

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

MRJUST - Right justify a field's data

```
MRJUST      (number,retcode,fieldname)
            (number,retcode,fieldname,zero,fill)
            (number,retcode,fieldname,zero,fill,type)
            (number,retcode,buffer,length)
            (number,retcode,buffer,length,fill)
            (number,retcode,buffer,length,fill,type)
```

MRJUST right justifies data within a field or a user's buffer. Normally, all blanks or nulls (binary zeros) in the rightmost part of the field or string are truncated. Unused positions to the left of the data are changed to nulls.

If "fill" is specified, this character is used to fill unused positions instead of nulls. If "type" is specified and non-zero, only rightmost nulls are truncated.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The field to be justified.

buffer **(character length "length")**
The buffer to be justified. The justified data is returned to this buffer.

length **(fullword)**
The length of the buffer to be justified.

zero **(fullword)**
A filler parameter that must be passed as a fullword of binary zero in order to distinguish the "fieldname" call vs. the "buffer"/"length" call. This is only necessary when you want to pass "fill" and/or "type."

fill **(character length 1)**
The character used to fill positions freed by the shifting of data. If not specified, unused positions are filled with nulls (binary zeros).

type **(fullword)**
A flag used to determine whether to truncate blanks and nulls or only nulls. If the value is non-zero, only nulls are truncated.

Return Codes

0 Data right justified.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

MRMDTF - Set whether MDT bits are reset at terminal output

```
MRMDTF        (number,retcode,flag)
```

MRMDTF sets or resets a flag that determines whether the Modified Data Tags are reset at the next terminal output. If the flag is zero, the MDT bits are reset (the default). If the flag is non-zero the MDT bits are not reset.

This routine can be used to write applications that leave fields primed as if a user entered data into them (for example, in the case of requesting that incorrect data be retyped).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MRSHOW - Signal total menu rewrite at next display

```
MRSHOW        (number,retcode)
```

MRSHOW signals VM Systems Group Internal Tools that the entire menu needs to be rewritten at the next display. This subroutine should be called whenever any full screen I/O is done outside the control of VM Systems Group Internal Tools. If MRSHOW is not called VM Systems Group Internal Tools assumes that the screen contents are still those written by VM Systems Group Internal Tools and only field differences are updated. This can cause strange looking screens and/or terminal I/O errors.

If your application calls XEDIT, for example, on return MRSHOW should be called.

You should call MRSHOW as seldom as possible. Otherwise VM Systems Group Internal Tools may output more to the terminal than is necessary.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MSAPLN - Sets the VM Systems Group Internal Tools Application Name

```
MSAPLN        (number,retcode,applname)
```

MSAPLN is used to set a specific VM Systems Group Internal Tools Application name. This name is used as part of the CMS Console Facility path name.

If you never call this subroutine, a random name (based on the time of day clock) is generated.

This subroutine has no specific benefit at present, but is reserved for future VM Systems Group Internal Tools use.

Parameters

number **(fullword binary)**
Ignored. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

applname **(character length 8)**
Is the name of this VM Systems Group Internal Tools application.

Return Codes

0 VM Systems Group Internal Tools application name is set.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MSAVE - Fetch or create MENUEXEC saved MRENT area

```
MSAVE        (number,retcode)
```

MSAVE is called to fetch the address of and/or create the MRENT area used by the MENUEXEC SAVE option. This allows an application to create menus to be passed to MENUEXEC or allows an application to use menus created by MENUEXEC.

The format and contents of the area returned are the same as those returned by the MRENT subroutine.

Parameters

number **(fullword pointer)**
Returns the address of the MENUEXEC saved MRENT buffer for VM Systems Group Internal Tools read/write storage data. See the "MRENT" subroutine for more details concerning this data area.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Data moved successfully.

90 ":R" area not available to save MRENT data.

MSCMSC - Sets whether the CMS Console Facility is used for full screen I/O

```
MSCMSC        (number,retcode,flag)
```

MSCMSC allows you to specify whether your VM Systems Group Internal Tools application will use the CMS Console Facility to perform full screen I/O.

By default, VM Systems Group Internal Tools will use the CMS Console Facility for full screen I/O. The Console Facility is always used in Multiple Terminal applications, and it is always used when your virtual machine is running in XA mode.

You might turn off use of the Console Facility if you suspect that it is causing problems in your application.

Parameters

number **(fullword binary)**
Ignored. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword)**
A value used to determine whether the CMS Console Facility is used. If it is zero, the Console Facility is not used. If it is set to any other value, the CMS Console Facility is used for VM Systems Group Internal Tools full screen I/O.

Return Codes

0 CMS Console Facility usage flag set.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MSCRSZ - Get terminal or menu size

```
MSCRSZ      (number,retcode,lines,chars)
            (number,retcode,lines,chars,cols,deflines,defcols)
```

MSCRSZ returns size information about your terminal or a loaded menu.

Parameters

number **(fullword binary)**
Pass zero if you desire terminal information, otherwise the number of the menu in question. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

lines **(fullword binary)**
Returns the number of lines on the terminal or menu.

chars **(fullword binary)**
Returns the number of characters (lines times columns) on the terminal or menu.

cols **(fullword binary)**
Returns the number of columns on the terminal or menu.

deflines **(fullword binary)**
Returns the number of lines on the terminal in 3277 compatibility mode.

defcols **(fullword binary)**
Returns the number of columns on the terminal in 3277 compatibility mode.

Return Codes

0 Subroutine completed successfully.

1 Your terminal is not a 327x.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MSCR2D - Move data from a menu field to your buffer

```
MSCR2D      (number,retcode,fieldname,buffer,length,rlen)
            (number,retcode,fieldname,buffer,length,rlen,offset)
            (number,retcode,fieldname,buffer,length,rlen,offset,fill)
```

MSCR2D moves data from a named field to a program buffer. If the field was modified by the user, the modified data is returned to the buffer. If the field was not modified by the user, the original unmodified contents of the field are returned to the buffer. If the field is shorter than the buffer, the remainder of the buffer is filled with nulls (binary zeros) or the specified fill character.

If MSIZEF was called with a non-zero return code, then differing sized buffers do not result in non-zero return codes from this routine.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field data is to be moved from.

buffer **(character length "length")**
The buffer into which field data is moved.

length **(fullword binary)**
The number of characters to be moved to the buffer.

rlen **(fullword binary)**
The number of characters not moved to the buffer if the buffer is shorter than the length of the field.

offset **(fullword binary)**
If specified, denotes the number of characters skipped from the leftmost part of the field before moving data to the buffer. If not specified, data is moved from the field starting with the field's first character.

fill **(character length 1)**
If specified, denotes the character used to fill out the unused rightmost remainder of the buffer. If not specified, the remainder of the buffer is set to nulls (binary zeros).

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

5 Data transfer - the field size is greater than the buffer size.

6 Data transfer - the buffer size is greater than the field size.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

18 The offset value (MSCR2D, MD2SCR) exceeds the fields size.

MSETAI - Start asynchronous terminal interrupt detection

```
MSETAI      (number,retcode)
            (number,retcode,flagbyte)
```

MSETAI initializes asynchronous interrupt detection. If "flagbyte" is passed, its address is remembered. When an interrupt is processed, the byte at "flagbyte" is changed to all ones (X'FF').

"MQRYAI" can be used to determine whether an interrupt has occurred. Alternately, the byte passed in "flagbyte" can be periodically inspected. There is no facility to transfer control to a point in an application at the time of the interrupt. You must occasionally sample the flagbyte or call "MQRYAI" to see if an interrupt is pending.

When an interrupt is pending, the data entered can be retrieved by calling "MRDSPR" using the menu number currently displayed. The data associated with the interrupt is not read until "MRDSPR" is called. If you do not care about the data entered with the interrupt, this call does not have to be made.

You must call "MCLRAI" when you no longer wish to trap asynchronous terminal interrupts. If you exit your application without clearing asynchronous interrupt trapping, unpredictable results can occur (for example, CMS does not clear "HNDINT" settings at command completion. When your program exits (or if it abends), console interrupts would branch to application code no longer present). "MEXIT" clears asynchronous interrupt handling if it was set.

Only one asynchronous interrupt handler can be enabled. If MSETAI is called while asynchronous interrupt handling is already in effect, the call is ignored.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flagbyte **(byte)**
A read/write byte in your application that can be used to flag the arrival of a terminal interrupt. MSETAI initializes this byte to zero. It is changed to all ones (X'FF') when an interrupt occurs.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MSETAT - Set a field attribute from a data structure

```
MSETAT        (number,retcode,fieldname,menusfe)
```

MSETAT changes a field's attribute definition to that given by a passed data structure. If the menu does not contain extended attributes, the color, highlighting and symbol set definitions are ignored.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field within the menu whose attribute is to be changed.

menusfe **(character length 10)**
Contains the attribute structure used to change the field's attribute. The attribute structure must be in the following form:

Byte	Contents
0	SFE attribute byte (X'29')
1	Count of following pairs (X'04')
2	Color type (X'42')
3	Color value (X'F1' BLUE, X'F2' RED, X'F3' PINK, X'F4' GREEN, X'F5' TURQUOISE, X'F6' YELLOW, X'F7' WHITE, X'00' Default)
4	Highlighting type (X'41')
5	Highlighting value (X'00' None, X'F1' BLINK, X'F2' REVERSE, X'F3' UNDERSCORE)
6	Logical symbol set type (X'43')
7	Logical symbol set value (X'00' default, X'40-EF' logical symbol set)
8	Standard attribute type (X'C0')
9	Standard attribute value (a bit definition - see the section on 3270 attributes for more information)

The macro "MENUATT" is provided with VM Systems Group Internal Tools to simplify the definition of this attribute. See the macro section in this book for more details.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

93 MENUSFE block passed is not formatted properly

MSETIN - Set pseudo input into a menu input buffer

```
MSETIN        (number,retcode,fieldname,buffer,length)
```

MSETIN (Set Input) simulates user input into a menu input buffer. The data in "buffer" is set into the VM Systems Group Internal Tools input control table as if the user entered this data the last time the menu was displayed. If "length" is zero, the input simulated is "ERASE EOF." If the parameter "buffer" is zero, this routine simulates no input at all in this field. The menu must have been displayed at least once before using this routine. This call allows you to implement functions similar to some editor "SOS" (Simulate on Screen") commands.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field that pseudo input is moved into.

buffer **(character length "length")**
The buffer containing the pseudo input character string. If the content of this parameter is binary zero (meaning a null parameter or a buffer address of zero) the "no input" condition is simulated. To overlay previous user input, this routine must be called twice: first with this parameter as zero to clear the previous input, then again to set the field to specific pseudo input.

length **(fullword binary)**
The number of characters to be used as pseudo input. If the value of this parameter is zero, and "buffer" is non-zero, the "ERASE EOF" condition is simulated. If the number of characters exceeds the field's length, the data used is truncated to the field length.

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

71 User input already exists for this field. No overlay was done.

72 User input already exists for this field. Previous input is overlaid.

MSETW - Defines the default window for page mode menu display

```
MSETW          (number,retcode,wname)
```

MSETW is used to define the default window used to display menus.

This routine allows you to "front-end" an existing VM Systems Group Internal Tools application and cause all of its displays to be presented in a VM Systems Group Internal Tools window.

This routine can be used to implicitly define the name of a window to be used for page mode menus (see MPGINT).

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

Return Codes

0 The window name is set as the default window name.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

704 A window with this name does not exist.

705 The virtual screen does not exist.

MSETWB - Defines the characters used for a viewport's borders

```
MSETWB        (number,retcode,wname,btl,btr,bbl,bbr,bt,bb,bl,br)
```

MSETWB is used to explicitly define the characters used for a viewport's borders.

By default, asterisks are used for border characters.

Different characters, extended colors, highlighting and symbol sets can be used for the four corners, the top and bottom lines, and the left and right lines.

An error code is returned if invalid data is passed in a border definition.

If your terminal does not support one or more extended attributes, these definitions are ignored.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

btl **(byte length 7)**
Defines the top left border character.

btr **(byte length 7)**
Defines the top right border character.

bbl **(byte length 7)**
Defines the bottom left border character.

bbr **(byte length 7)**
Defines the bottom right border character.

bt **(byte length 7)**
Defines the top border character.

bb **(byte length 7)**
Defines the bottom border character.

bl **(byte length 7)**
Defines the left border character.

br **(byte length 7)**
Defines the right border character.

Border definitions

The following table defines the data contained in border definitions:

Byte 0 Data character

Byte 1 Attribute character (must be X'F0')

Byte 2 Extended attribute character (must be X'00')

Byte 3 Character attribute data. Bit format is XXXYYZZZ, which is:

- **XXX** Symbol set number (00-07)
- **YY** Highlighting (00 - none, 01 - blink, 02 - reverse, 03 - underscore)
- **ZZZ** Color (00 - default, 01 - blue, 02 - red, 03 - pink, 04 - green, 05 - turquoise, 06 - yellow, 07 - white)

Byte 4 Second extended attribute character (must be X'00')

Byte 5 Second character attribute data. Bit format is XXXXYZZZ, which is:

- **XXXX** Field outlining
- **Y** Field transparency
- **ZZZ** Background color (see color definitions above)

Byte 6 Second portion of compound data character (if byte 0 = X'08')

Return Codes

0 Information about the window is returned.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

704 A window with this name does not exist.

705 The virtual screen does not exist.

714 A window border definition contains invalid data.

MSETWF - Modifies a window's flags

```
MSETWF        (number,retcode,wname,flags)
```

MSETWF is used to modify the flags set when a window was first defined.

The passed flags are identical to those used when calling MDEFW. These flags are used to determine whether this is an active (displayed) window, whether borders are displayed around its viewport and whether the menu name is used as a title in the top viewport border.

This routine allows you to modify window flags after a window is defined. For example, you could allow a user to turn borders on and off in your application.

This subroutine only sets an VM Systems Group Internal Tools flag. Changes take effect at the next terminal display.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

flags **(byte)**
Specifies how the window's viewport is displayed. The following bits are currently defined:

WINVBOR X'80' Display vertical borders if possible
WINHBOR X'40' Display horizontal borders if possible
WINTITLE X'20' Display the menu name on the top border if possible
WINACTIV X'01' This viewport is active, that is, it is visible

Return Codes

0 Window flags have been changed.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
704 A window with this name does not exist.
705 The virtual screen does not exist.

MSEXTA - Set the extended attributes of a field

```
MSEXTA        (number,retcode,fieldname,color,hi,ps)
```

MSEXTA changes the extended attributes of a field in the menu. MSEXTA can be called even if the terminal does not support extended attributes. A non-zero return code will only be returned if the menu itself does not contain extended attribute data.

To set standard attribute values, use subroutine "MFSET."

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
The name of the field whose extended attributes are to be changed.

color **(character length 10)**
The color attribute of the field. The color may be "DEFAULT" or "DEFCOL" (default color for the intensity and protection attributes of the field), "BLUE," "PINK," "RED," "GREEN," "TURQUOISE," "YELLOW" or "WHITE."

hi **(character length 10)**
The highlighting attribute of the field. "Hi" may be "DEFAULT" or "NOHILIGHT" (no highlighting), "BLINK," "REVERSE," or "UNDERSCORE."

ps **(fullword)**
Sets the logical symbol set number for the field. The value is retrieved from the rightmost byte of the fullword. A value of zero denotes the standard symbol set. When the menu is displayed, if the symbol set in question is not loaded, the field is displayed with symbol set zero to avoid I/O errors. The number passed by "MPSLD" must match the number passed in this routine for the field to be displayed with the named symbol set loaded by "MPSLD." The number must be between X'40' and X'EF.' The value X'F1' cannot be used to set a field's logical symbol set; this value can only be used in "MADDSA," "MCHGSA" and "MDELSA."

Return Codes

0 Data returned.

2 MLOAD, MLOADX - The menu file you are attempting to load is not formatted properly.

2 MQEXTP - The attribute data passed is improperly formatted.

2 MFQRY, MFSET, MQEXTA, MSEXTA The field attribute could not be found.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

14 The field name specified was not found in the menu.

15 Attributes cannot be set on a menu without any fields.

16 An attribute parameter passed was invalid.

35 The menu does not contain extended attribute data.

37 The programmable symbol set number passed is invalid.

MSHARE - Share a menu between application levels

```
MSHARE        (number,retcode,flag)
```

MSHARE sets or resets a flag which determines whether a given menu can be accessed by multiple application levels. If the flag is non-zero, any application level that passes this menu's number in the "number" parameter of a subroutine can manipulate this menu. If the flag is zero (the default) the menu can only be accessed by applications with an application identifier equal to or lower than the identifier of the application which loaded or created the menu. Each loaded menu has its own share flag.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear the share flag, non-zero to set the share flag.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MSIZEF - Set whether to return error code on buffer size mismatch

```
MSIZEF        (number,retcode,flag)
```

MSIZEF sets or resets a flag that determines whether return codes 5 and 6 are returned from MD2SCR, MSCR2D, MLD2S, and MLS2D if the program buffer and the menu field sizes are not equal. If the flag is zero, these return codes are set (the default). If the flag is non-zero these conditions result in a zero return code.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MSKIP - Set SKIP bits before output of a menu

```
MSKIP        (number,retcode,flag)
```

MSKIP sets or resets a flag that determines whether all protected field's SKIP bits are set before output. SKIP bits are used to enable the cursor to automatically skip protected fields. If the MSKIP flag is non-zero, SKIP bits are set in all protected fields. If the flag is zero, SKIP bits are left unchanged. If this routine is never called, the default skip flag is off. Each loaded menu has its own skip flag.

If this routine is called with a non-zero flag, the SKIP bits are set at the next call to a display subroutine (MDSPRD, etc.). Once this is done, the SKIP bits remain on for the remaining use of the menu unless individually reset even if MSKIP is again called with a flag of zero (since this means the SKIP bits are to be left unchanged).

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MSMTO - Sets whether the Multiple Terminal Facility is used

```
MSMTO        (number,retcode,flag)
```

MSMTO allows you to specify whether your VM Systems Group Internal Tools application will use the VM Systems Group Internal Tools Multiple Terminal Facility.

The VM Systems Group Internal Tools Multiple Terminal Facility (MTF) allows you to control multiple terminals from a single CMS application. This subroutine must be called in order to initialize MTF.

Once this subroutine is called, many other XMENU subroutines expect a logical terminal number to be passed in the second fullword of the number parameter (which passes menu numbers in its first fullword).

Parameters

number **(fullword binary)**
Ignored. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword)**
A value used to determine whether the VM Systems Group Internal Tools Multiple Terminal Facility is used. If it is zero, MTF is not used. If it is set to any other value, the VM Systems Group Internal Tools Multiple Terminal Facility will be used.

Return Codes

0 VM Systems Group Internal Tools Multiple Terminal Facility usage flag set.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MS2EXE - Move input from menu fields to EXEC variables

```
MS2EXE        (number,retcode,flag)
```

MS2EXE transfers data from user input to menu fields to EXEC variables. Each named field on the menu sets EXEC variable data if the user modified the field at the last input.

This routine can only be used if the EXECCOMM interface exists for this EXEC interpreter. If not, a return code of "89" is returned. Alternatively, a combination of calls to "MTSTF," "MSCR2D" and "MD2SCR" could be called for each menu field.

"MEXQRY" must be called prior to the invocation of this routine to initialize EXEC variable look up.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
If the low order bit (01) is set, and if running under the system product interpreter (REXX) and if the user pressed "ERASE EOF" at the last input for this field, the EXEC variable is dropped (rather than set to "" as is the default).

Return Codes

0 Data moved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

89 MEXE2S, MS2EXE EXECCOMM not supported.

MTBFLG - Use text borders on window displays if possible

```
MTBFLG        (number,retcode,flag)
```

MTBFLG sets or resets a flag that determines whether text borders are used on window displays if possible.

Text borders are window borders that use the 3270 TEXT character set. This results in single line borders including square corners (similar to the box surrounding the subroutine description above.

If the terminal doesn't support the TEXT character set, vertical lines are used for the sides, and dashes are used for the corners, top and bottom lines.

If the flag is non-zero, text borders are generated. If zero, standard borders are generated. If this routine is never called standard borders are generated.

MSETWB can also be used to specifically change the characters and attributes used in window borders.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MTOD2V - Convert a TOD clock value to character values

```
MTOD2V      (todval,retcode,date)
            (todval,retcode,date,time)
            (todval,retcode,date.time,weekday)
            (todval,retcode,date,time,weekday,julian)
```

MTOD2V converts a time of day clock value to printable date and time values. This conversion is done without any adjustment for local time.

Parameters

todval **(doubleword binary)**
Input time of day clock value.

retcode **(fullword binary)**
The return code for the conversion (see below).

date **(character length 8)**
Output date in "MM/DD/YY" format.

time **(character length 8)**
Output time in "HH:MM:SS" format.

weekday **(character length 9)**
Output day of the week, e.g. "MONDAY," "TUESDAY," etc.

julian **(character length 6)**
Output date in "YY.DDD" format.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MTOPW - Gives a window the highest display priority

```
MTOPW          (number,retcode,wname)
```

MTOPW is used to give a window the highest display priority.

As each window is defined, it implicitly is given the highest display priority, that is, it is placed on the virtual screen after all older windows. This can cause lower priority windows to be partially or completely overlaid by higher priority windows.

MTOPW lets you reorder the display priority of windows, so that an older window can be placed on top of newer windows.

When you are displaying an older, hidden window, it is best to call MTOPW and MADDW before use, and MDELW after use.

This routine only orders the window queue. It does not cause the terminal screen to be updated.

MBOTW can be used to give a window lowest priority.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. If running under the VM Systems Group Internal Tools Multiple Terminal Facility, the second fullword is the logical terminal this virtual screen is associated with. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

Return Codes

0 The window is given the highest display priority.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

704 A window with this name does not exist.

705 The virtual screen does not exist.

MTRACE - Start or stop the VM Systems Group Internal Tools trace table

```
MTRACE        (number,retcode,flag)
```

MTRACE is used to start or stop the VM Systems Group Internal Tools trace table. When an application begins, the default is to start the trace table. This call can be used to "freeze" the table at the point of an error.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
If zero is passed, VM Systems Group Internal Tools tracing stops, if non-zero tracing is resumed.

Return Codes

0 Subroutine completed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MTRPA1 - Trap PA1 if used as input to this menu

```
MTRPA1        (number,retcode,flag)
```

MTRPA1 sets or resets a flag that determines whether PA1 is trapped by VM Systems Group Internal Tools if pressed by the user. If the flag is non-zero, PA1 is trapped and returned as "PA1." Under CMS, if the flag is zero, PA1 places the user into "CP READ." If this routine is never called the default PA1 flag is off. Each loaded menu has its own PA1 flag.

Under VM/SP, once the PA1 flag is set it remains active until the next non-full screen I/O even if not specified in intervening full screen displays. Consequently, the setting of this flag may cause implicit trapping of PA1 on other full screen displays.

Use of this routine may implicitly cause the CP TERM BRKKEY to be set to NONE. If you are concerned that your application saves and restores this value, use the MMAKEA and MDROPA subroutines in your application.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MTSTF - Determine whether a user input data into a field

```
MTSTF       (number,retcode,fieldname)
            (number,retcode,fieldname,length)
```

MTSTF determines whether a field was modified by a user, and if so, optionally returns the length of the user's input data (the length value can be used to dynamically obtain storage of the appropriate size to retrieve the user data). If the user pressed "ERASE EOF" the data length returned is zero. If the field had no input a non-zero return code is set and "length" remains unchanged.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

fieldname **(character length 7)**
Specifies the field to be tested.

length **(fullword binary)**
Returns the length of the field modified by the user. If the user pressed "ERASE EOF," "length" is returned as zero.

Return Codes

0 Data retrieved successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

10 No user input made to the requested field (MTSTF, MLS2D).

14 The field name specified was not found in the menu.

MUFILE - Update data in MENUCTRL files

```
MUFILE      (number,retcode,filename,fname1,...fnamen)
            (number,retcode,filename,null,fname1,...fnamen)
```

MUFILE updates a MENUCTRL file with this menu's user input data and/or fields with Modified Data Tag (MDT) set. A MENUCTRL file can previously exist from MENUEXEC, from this program (or a previous program) calling "MUFILE." Only data statements within the MENUCTRL file are updated; all others, including MENUEXEC commands, are copied into the updated version of the MENUCTRL file.

Data records pertaining to fields not in the menu, fields not in the parameter list, or fields not updated by the user remain unchanged in the MENUCTRL file.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

filename **(character length 8)**
The file name of the MENUCTRL file.

null **(character length 8)**
Provides a space-holder where the TSO DDNAME would reside. This is provided to simplify migration of a program from CMS to TSO and vice versa. The string within "null" must not be the name of any field in the menu.

fname1...fnamen (character length 8)
A list of the fields whose data is to be updated in the MENUCTRL file. If a name in the list is not modified by the user it is ignored and the previous data for that field (if any) in the MENUCTRL file is unchanged.

Return Codes

0 MENUCTRL file updated successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

14 The field name specified was not found in the menu.

23 A field name specified in MGFILE or MUFILE is invalid or does not exist on the menu.

25 The MENUCTRL file referenced in MGFILE or MUFILE is improperly formatted.

26 A badly formatted data record was found in the MENUCTRL file.

27 No updates have been applied to the MENUCTRL file (MUFILE).

51 The MUFILE temporary work file already exists (FILETYPE "CMSUT1").

4xx Error in CMS FSREAD (xx is the FSREAD code).

402 Invalid input buffer address.

403 Permanent I/O error occurred.

408 Incorrect length record read.

412 End of file detected.

425 Insufficient free storage.

5xx Error in CMS FSWRITE (xx is the FSWRITE code).

502 Invalid output buffer address.

503 Permanent I/O error occurred.

508 Attempt to write an incorrect length record.

512 Attempt to write on a read only disk.

513 The disk is full.

522 Virtual storage capacity exceeded.

525 Insufficient free storage.

MUNLKF - Set whether the keyboard is unlocked at next output

```
MUNKLF        (number,retcode,flag)
```

MUNKLF sets or resets a flag that determines whether the keyboard is unlocked at the next terminal output. If the flag is zero, the keyboard is unlocked (the default). If the flag is non-zero the cursor is not unlocked.

This routine can be used to write applications that display data where no user input is desired.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MUPCAS - Convert all user input on this menu to upper case

```
MUPCAS        (number,retcode,flag)
```

MUPCAS sets or resets a flag that determines whether user input is converted to upper case. If the flag is non-zero, user input is converted to upper case. If the flag is zero, user input remains unchanged. If this routine is never called, the default upcase flag is off. Each loaded menu has its own upcase flag.

Conversion to upper case is performed at the time the menu is displayed and read. Resetting the MUPCAS flag only affects input received following the invocation of this routine.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MUPSCM - Update output, set Modified Data Tag and Cursor Position

```
MUPSCM        (number,retcode)
```

MUPSCM updates the output menu with all of the user's previous input. In addition, any user modified fields have their modified data tags (MDT) set, and the cursor position is updated to where it was at the end of the last input.

This special form of update is used to simulate an "ignore" user input condition for cases where the user must correct some previously input data. For example, MENUEXEC uses this call when "REQUIRE" conditions fail.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Data, attributes and cursor position updated successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

MUPSCR - Update a menu with last input

```
MUPSCR        (number,retcode)
```

MUPSCR updates the output menu with all of the user's previous input. VM Systems Group Internal Tools keeps separate buffers for menu output and menu input so that you can move only parts of the user's input data to the menu for the next menu display. MUPSCR is a quick way to move all user input from the input buffers to the output menu.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Menu updated successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

9 No user input yet (on calls for input information).

MUSER - Get your userid

```
MUSER        (userid)
             (userid,retcode)
```

MUSER returns your system user identification (userid).

If a non-zero return code needs to be passed and the "retcode" parameter is not supplied, the return code is only returned to the application in general register 15.

Parameters

userid **(character length 8)**
Returns your userid.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

Return Codes

0 Subroutine completed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MVERS - Get VM Systems Group Internal Tools version information

```
MVERS        (number,retcode,product,prodcode,version)
```

MVERS returns the product name, code and version of the VM Systems Group Internal Tools subroutine library. This data can be used to insure that your application has been linked with the appropriate version and level of the VM Systems Group Internal Tools subroutine package.

Parameters

number **(fullword binary)**
This parameter is ignored.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

product **(character length 8)**
Returns the character name of the product. This will contain "VM Systems Group Internal Tools."

prodcode **(character length 12)**
Returns the product number in character format. This will contain "5664-167-121."

version **(character length 8)**
Returns the version, release and maintenance level of VM Systems Group Internal Tools. This is returned as "VV.RR.MM" where "VV" is the version, "RR" is the release and "MM" is the maintenance level of VM Systems Group Internal Tools.

Return Codes

0 Subroutine completed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MV2JUL - Convert a Gregorian date to a Julian date

```
MV2JUL      (julian,retcode,date)
```

MV2JUL converts a Gregorian date (MM/DD/YY) into a Julian date (YY.DDD).

Parameters

julian **(character length 6)**
Output date in "YY.DDD" format.

retcode **(fullword binary)**
The return code for the conversion (see below).

date **(character length 8)**
Input date in "MM/DD/YY" format.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

32 Invalid MM/DD/YY passed to MV2TOD or MV2JUL, invalid YY.DDD to MJUL2V.

MV2TOD - Convert date and time into a TOD clock value

```
MV2TOD        (todval,retcode,date,time)
```

MV2TOD converts "MM/DD/YY" and "HH:MM:SS" character values to a time-of-day clock value. The conversion performed is assumed to have a time zone offset of zero.

Parameters

todval **(doubleword binary)**
Output time-of-day clock value.

retcode **(fullword binary)**
Return code for the conversion.

date **(character length 8)**
Input date in "MM/DD/YY" format.

time **(character length 8)**
Input time in "HH:MM:SS" format.

Return Codes

0 Conversion performed.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

32 Invalid MM/DD/YY passed to MV2TOD or MV2JUL, invalid YY.DDD to MJUL2V.

33 Invalid HH:MM:SS passed to MV2TOD.

MWAIT - Wait for an interrupt from a Multiple Terminal Facility Terminal

```
MWAIT          (number,retcode,status)
```

MWAIT is used by a VM Systems Group Internal Tools Multiple Terminal Facility application to wait for the next available terminal (user) interrupt.

When an application is processing a number of terminals at once, it becomes necessary to wait at times for the next terminal request. This routine returns the first pending terminal requiring service.

The status parameter is used to return the terminal's status at the time of the interrupt. The bits in this status byte match those in the device status byte of a Channel Status Word or Subchannel Status Word.

Parameters

number **(two binary fullwords)**
The first fullword is ignored. The second fullword returns the logical terminal number for the Multiple Terminal Facility logical terminal requiring service. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

status **(byte)**
Returns the device status for the interrupting device. This is most usually X'80' or attention.

Return Codes

0 Interrupting terminal information is present.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

96 Subroutine only supported under XMENU 2 MTF.

MWCCF - Control Write Control Characters

```
MWCCF        (number,retcode,flag)
```

MWCCF sets or resets a flag that determines whether Write Control Character (WCC) bits from lower priority windows are ored in with the highest priority window's WCC.

The most significant WCC bits control sounding the alarm and unlocking the keyboard.

If you want only the highest priority menu to sound the alarm, call this routine with a non-zero flag. If you want the alarm sounded if any window has its alarm bit set, call this routine with a zero flag. If this routine is never called, WCC bits are or-ed together.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

flag **(fullword binary)**
Set to zero to clear this option flag, non-zero to set this option.

Return Codes

0 Flag set successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

MWINRB - Perform an unsolicited read buffer from a window

```
MWINRB      (number,retcode,wname,buffer,length,rlen)
```

MWINRB is used to do an unsolicited read buffer operation from the terminal and a window.

This subroutine is intended for special purposes only.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

buffer **(character length "length")**
The buffer to receive the read buffer input from window wname.

length **(fullword binary)**
Contains the length of buffer. If more data is read than can fit into buffer it is truncated.

rlen **(fullword binary)**
Contains the actual length of data read into buffer.

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
99 The referenced device is disconnected.
100 The referenced device is not attached/offline.
103 Zero/non-positive buffer length passed to a subroutine.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.
644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MWINRD - Perform unsolicited read modified from a window

```
MWINRD      (number,retcode,wname,buffer,length,rlen)
```

MWINRD is used to perform an unsolicited read modified from the terminal and a window. A keyboard unlock write is performed followed by a hardware read modified without waiting for user input.

If an asynchronous interrupt was previously detected by the use of "MSETAI" the keyboard unlock is not preformed.

This subroutine is intended for special purposes only.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

buffer **(character length "length")**
The buffer to receive the read modified user input for window wname.

length **(fullword binary)**
Contains the length of buffer. If more data is read than can fit into buffer it is truncated.

rlen **(fullword binary)**
Contains the actual length of data read into buffer.

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
99 The referenced device is disconnected.
100 The referenced device is not attached/offline.
103 Zero/non-positive buffer length passed to a subroutine.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MWINRM - Wait for input, then do read modified from a window

```
MWINRM        (number,retcode,wname,buffer,length,rlen)
```

MWINRM is used to wait for user input, without any initial output, then perform a read modified from the terminal and a window.

This subroutine is intended for special purposes only.

Parameters

number **(fullword binary)**
Immaterial. If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

buffer **(character length "length")**
The buffer to receive the read modified input from window wname.

length **(fullword binary)**
Contains the length of buffer. If more data is read than can fit into buffer it is truncated.

rlen **(fullword binary)**
Contains the actual length of data read into buffer.

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
99 The referenced device is disconnected.
100 The referenced device is not attached/offline.
103 Zero/non-positive buffer length passed to a subroutine.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MWINW - Custom full screen output to a window

```
MWINW        (number,retcode,wname,buffer,length)
```

MWINW is used to output your own custom 3270 full screen data streams to an VM Systems Group Internal Tools window. It does not provide any additional VM Systems Group Internal Tools services other than transferring the data as is to the window, and displaying the current virtual screen on the terminal.

MWINW causes an implicit "MRSHOW" - the next menu displayed outside the window environment will entirely refresh the screen (rather than send changed data only).

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'00'	Issue Write command.
X'80'	Issue Write/Erase command.
X'C0'	Issue Write/Erase alternate.
X'10'	Trap PA1 if pressed at next input.
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

buffer **(character length "length")**
The buffer to be output to the window. This buffer should contain all appropriate data and 3270 orders.

length **(fullword binary)**
Contains the length of buffer.

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
99 The referenced device is disconnected.
100 The referenced device is not attached/offline.
103 Zero/non-positive buffer length passed to a subroutine.
6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MWINWR - Custom full screen I/O to a window

```
MWINWR      (number,retcode,wname,buffer,length,inbuff,inlen,inres)
```

MWINWR is used to output your own custom 3270 full screen data streams to an VM Systems Group Internal Tools window and wait for user input. It does not provide any additional VM Systems Group Internal Tools services other than transferring the data as is to the window and retrieving the window's input data.

MWINWR causes an implicit menu system "MRSHOW" - the next menu displayed outside of the window environment will entirely refresh the screen (rather than send changed data only).

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'00'	Issue Write command.
X'80'	Issue Write/Erase command.
X'C0'	Issue Write/Erase alternate.
X'10'	Trap PA1 if pressed at next input.
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

buffer **(character length "length")**
The buffer to be output to the window. This buffer should contain all appropriate data and 3270 orders.

length **(fullword binary)**
Contains the length of buffer.

inbuff **(character length "inlen")**
The buffer to receive the user's full screen input into the window.

inlen **(fullword binary)**
Contains the length of inbuff. If more data is received than will fit into inbuff, the data is truncated.

inres **(fullword binary)**
Returns the number of bytes of data moved into inbuff.

Return Codes

0 Data displayed successfully.

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

99 The referenced device is disconnected.

100 The referenced device is not attached/offline.

103 Zero/non-positive buffer length passed to a subroutine.

6xx Full screen I/O routine error.

604 The terminal does not support full screen I/O.

608 Attention received during write.

612 A permanent I/O error occurred.

616 Program attempted to do full screen read before the first full screen write.

620 The virtual console does not exist.

628 Insufficient storage to fulfill full screen request

632 Full screen function is not supported.

636 Screen busy (not in full screen mode) - total screen rewrite needed.

640 Terminal was disconnected.

644 Unit check occurred - sense data returned.

648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

MWRITE - Write a menu to a DASD file

```
MWRITE      (number,retcode,newname)
            (number,retcode,newname,flag)
```

MWRITE is used to create or update a menu in a DASD file from an active menu in virtual storage. If "flag" is not specified and the menu already exists, it is replaced.

Parameters

number **(fullword binary)**
Pass the value returned to the "number" parameter by "MLOAD," "MLOADX" or "MPGINT" denoting the menu you wish to manipulate. If using the XMENU subroutine package in a read-only mode, number should point to the buffer initialized by the "MRENT" subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

newname **(character length 8)**
Specifies the name given the menu when it is written to a DASD file. This becomes both the menu's internal name and its FILENAME. The output FILETYPE is "MENU." The output FILEMODE is "A2."

flag **(byte)**
Specifies control information for MWRITE. If the high order bit is set (X'80') and the menu already exists, MWRITE replaces it. If the high order bit is not set and the menu already exists, MWRITE exits without replacing the menu and returns to the caller with a return code of 82.

Return Codes

0 Subroutine completed successfully.

4 The menu number supplied in "number" is not loaded (or was purged by XMENU due to system errors).

7 Insufficient parameters were passed to the called subroutine.

8 Insufficient storage available to fulfill the request.

82 MWRITE - a menu with the same name already exists as a DASD file.

5xx Error in CMS FSWRITE (xx is the FSWRITE code).

502 Invalid output buffer address.

503 Permanent I/O error occurred.

508 Attempt to write an incorrect length record.

512 Attempt to write on a read only disk.

513 The disk is full.

522 Virtual storage capacity exceeded.

525 Insufficient free storage.

MW2ARR - Move Window Data to Application Arrays

```
MW2ARR      (number,retcode,wname,cpos,dbuff,abuff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
                exa2buff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
                exa2buff,exc2buff)
            (number,retcode,wname,cpos,dbuff,abuff,exabuff,excbuff,
                exa2buff,exc2buff,dbcbuff)
```

MW2ARR allows you to retrieve data from a window into application arrays. The window is not displayed, the data is simply moved to your program.

This subroutine can be used to display "free form" data, such as engineering designs, or small arrays, such as lists of correctly spelled words in pop-up windows. It is particularly useful for creating and displaying windows that are dynamically created at run time.

There are four subroutines associated with this facility:

MARRW displays the array data on the terminal, then returns without waiting for user input.

MARRWR displays the array data on the terminal, waits for user input, then returns this input into the passed arrays.

MARR2W moves array data to a window without any display. This is used to preload a group of windows before they are displayed.

MW2ARR moves window data to your arrays based on user input during the last display. This routine allows you to fetch input made to multiple windows made in one user input operation.

You must use the MDEFW subroutine to first define your window before using these routines. The size of your arrays is implicitly defined as the size of the window (number of rows multiplied by the number of columns in a row).

The cursor position and the initial Write Control Character are passed in the cpos parameter. This parameter also returns the key pressed in two forms.

Parameters

number **(fullword binary)**
Control flags used to determine the type of I/O desired. The flags are contained in the rightmost (low order) byte of the fullword. The possible flags are:

Code	Command
X'00'	Issue Write command.
X'80'	Issue Write/Erase command.
X'C0'	Issue Write/Erase alternate.
X'10'	Trap PA1 if pressed at next input.
X'1xx'	Return if a screen busy condition occurred.
X'2xx'	Return without any error message if a unit check occurred.

If using the VM Systems Group Internal Tools subroutine package in a read-only mode, number should point to the buffer initialized by the MRENT subroutine.

retcode **(fullword binary)**
The called XMENU subroutine always returns a value into "retcode." This returned value is the return (or result) code for this request.

wname **(character length 16)**
The name of the window, left justified, padded with blanks.

cpos **(character length 24)**
Passes and returns data for the array operation. The following data is kept in cpos:

Bytes	Use
0-3	Passes the current cursor position, returns the cursor position if it has been moved. The upper left corner of the window is position zero.
4	Passes the Write Control Character (WCC) for window output.
5	Returns the Attention Identifier (AID) byte of the key pressed.
6	Passes flags - X'80' means don't position the cursor,
7	Reserved for future use
8-15	Maps the programmable symbol set bits into logical symbol set numbers. Each byte corresponds to the programmable symbol set numbers passed in the exabuff and excbuff arrays, and contains the symbol set number for this symbol set.
16-23	Returns the key pressed, i.e. the character equivalent of the AID byte.

dbuff **(character array)**
Passes and returns character data for the window. The array size must be equal to the window size in bytes. Each position must be either valid displayable data, the code X'1D' which identifies the position of a 3270 attribute, or the code X'08' which identifies a graphics escape character whose second byte is located at the same array offset in the dbcbuff array.

abuff **(character array)**
Passes and returns 3270 attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

In general, the bits in each byte are as follows:

Bits	Use
0-1	Encoded based on the contents of bits 2-7.
2-3	Define a field's protection - 00 is unprotected, 01 is numeric, 02 is protected no skip, 03 is protected, skip.
4-5	Define a field's intensity - 00 is dim, 01 is light pen detectable, 02 is bright, 03 is dark.

6	is unused.
7	is the Modified Data Tag (MDT) bit.

See the 3270 Datastream Programmer's Reference for the actual definitions of the bits in these attributes.

exabuff **(character array)**
Passes and returns 3270 extended attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

The bit definitions of each byte in this array are as follows:

Bits	Use
0-2	Correspond to the programmable symbol sets passed in the cpos parameter. Values 0 through 7 correspond to bytes 0 to 7 in cpos respectively.
3-4	Correspond to the extended highlighting value. 00 is no highlighting, 01 is blink, 02 is reverse video, and 03 is underscore.
5-7	Correspond to the extended color value. 00 through 07 correspond to blue, red, pink, green, turquoise, yellow, and white respectively.

If the terminal does not support extended attributes, the data in this array has no effect.

excbuff **(character array)**
Passes and returns character attribute data for the window. The array size must be equal to the window size in bytes. Each defined byte acts as a field attribute override for the character at that window position.

The bytes in excbuff are in the same format as the bytes in exabuff.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

If the terminal does not support extended attributes, the data in this array has no effect.

exa2buff **(character array)**
Passes and returns secondary 3270 extended attribute data for the window. The array size must be equal to the window size in bytes. Each byte at the array offset position of an X'1D' byte in the dbuff must be a valid 3270 attribute. Attribute values at positions other than field starts are ignored. VM Systems Group Internal Tools will propigate these bytes throughout the fields in this array, thus the contents of positions other than where fields start are unpredictable.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

The bit definitions of each byte in this array are as follows:

Bits	Use
0-3	Correspond to the field's outlining data.
4	Corresponds to the field's background transparency.
5-7	Correspond to the background color value. 00 through 07 correspond to blue, red, pink, green, turquoise, yellow, and white respectively.

If the terminal does not support these extended attributes, the data in this array has no effect.

exc2buff **(character array)**
Passes and returns secondary character attribute data for the window. The array size must be equal to the window size in bytes. Each defined byte acts as a field attribute override for the character at that window position.

The bytes in exc2buff are in the same format as the bytes in exa2buff.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no extended attributes.

If the terminal does not support extended attributes, the data in this array has no effect.

dbcbuff **(character array)**
Passes and returns compound (graphic escape) character data for the window. The array size must be equal to the window size in bytes. Each position must contain valid displayable data.

If this buffer isn't passed, or its address is zero, it is assumed that the window has no compound characters.

If the terminal does not support graphic escape, the data in this array has no effect.

Return Codes

0 Data displayed successfully.
7 Insufficient parameters were passed to the called subroutine.
8 Insufficient storage available to fulfill the request.
99 The referenced device is disconnected.
100 The referenced device is not attached/offline.
103 Zero/non-positive buffer length passed to a subroutine.
6xx Full screen I/O routine error.
604 The terminal does not support full screen I/O.
608 Attention received during write.
612 A permanent I/O error occurred.
616 Program attempted to do full screen read before the first full screen write.
620 The virtual console does not exist.
628 Insufficient storage to fulfill full screen request
632 Full screen function is not supported.
636 Screen busy (not in full screen mode) - total screen rewrite needed.
640 Terminal was disconnected.
644 Unit check occurred - sense data returned.
648 Full screen routine called before initialization.

652 Terminal type changed between or during full screen call.

704 A window with this name does not exist.

705 The virtual screen does not exist.

716 Attempt to read from a window that was never displayed.

717 Attempt to read from a window when terminal is not in window mode.

725 Insufficient storage to allocate terminal input buffer.

751 An error was detected in an output datastream directed to a window.

753 Insufficient storage to display the menu in a window.

754 There are no active (visible) windows to display.

758 Insufficient storage to allocate a window control block (CVTA).

759 Insufficient storage to allocate a window control block (CVTI).

760 An error was detected in an input datastream directed to a window.

762 Insufficient storage to return a datastream (CVTJ).

766 Insufficient storage to return a datastream (CVTB).

768 Insufficient storage to allocate the character array (CVTA).

769 Insufficient storage to allocate the character array (CVTI).

770 Insufficient storage to allocate a window input buffer (WINF).

771 Insufficient storage to allocate terminal character array (WINF).

772 Insufficient storage to allocate terminal attribute array (WINF).

778 Insufficient storage to allocate the attribute array (CVTA).

779 Insufficient storage to allocate the attribute array (CVTI).

788 Insufficient storage to allocate the extnd. attribute array (CVTA).

789 Insufficient storage to allocate the extnd. attribute array (CVTI).

798 Insufficient storage to allocate 2nd extnd. attribute array (CVTA).

799 Insufficient storage to allocate 2nd extnd. attribute array (CVTI).

Chapter 3. XMENU Assembler Macros

This chapter describes VM Systems Group Internal Tools assembler macros.

MENUATT - Define page mode attribute

[label]	**MENUATT**	[COLOR=[DEFAULT|BLUE|RED|PINK|GREEN| TURQUOISE|YELLOW|WHITE],] [HI=[DEFAULT|BLINK|UNDERSCORE|REVERSE],] [PS=[00|40...EF],] [PROT=[YES|NO|NUMERIC],] [INTENS=[DIM|BRIGHT|DARK|LGHTPEN],] [SKIP=[NO|YES],] [MDT=[NO|YES]

MENUATT is an assembler macro used to create the ten byte attribute used by the "MPGINT" and "MPGD2S" subroutines. The default values are COLOR=**DEFAULT**, HI=**DEFAULT**, PS=**00**, PROT=**YES**, INTENS=**DIM**, SKIP=**NO** and MDT=**NO** (the first values in the above list).

Parameters

COLOR **(color value)**
Specifies the attribute extended color.

HI **(highlight value)**
Specifies the attribute extended highlighting.

PS **(programmed symbol set value)**
Specifies the attribute programmed symbol set.

PROT **(protection value)**
Specifies the attribute field protection.

INTENS **(intensity value)**
Specifies the attribute field intensity.

SKIP **(SKIP value)**
Specifies the attribute SKIP bit. If "SKIP=YES" is specified, "PROT=YES" must also be specified.

MDT **(MDT value)**
Specifies the attribute MDT (Modified Data Tag).

Return Codes

8 Invalid COLOR value specified.

8 Invalid HI value specified.

8 PROT=YES must be specified with SKIP=YES.

8 Invalid INTENS value specified.

MENUFLGS - Define MFLAGS flags

[label]	**MENUFLGS**	[MENUALRM,] [MENUSKIP,] [MENUMDT,] [MENUNULL,] [MENUMAP,] [MENUPA1,] [MENUCLER,] [MENUUPCS] [] [DEFINE]

MENUFLGS is an assembler macro used to create the flag passed to the "MFLAGS" subroutine in its "flagbyte" parameter.

Each parameter represents a bit within the flag definition. To set the appropriate flag, code the appropriate parameter. The parameters can be specified in any order. If no parameters are coded, the flag is initialized to all zeros. If "DEFINE" is coded, only bit equates are defined, no flag is defined.

Parameters

DEFINE

If coded, specifies that the macro generates the assembler equates for the bits contained in the flag. If "DEFINE" is specified, it must be the first and only parameter passed to the macro. "DEFINE" does not generate any storage - a second call to the macro with the other parameters must be made to generate the flag data.

MENUALRM

If coded, specifies that the alarm should be sounded the next time the menu is displayed.

MENUSKIP

If coded, specifies that all protected fields should have their SKIP bits set at the next output.

MENUMDT

If coded, specifies that all unprotected fields should have their MDT bits set at the next output.

MENUNULL

If coded, specifies that all unprotected fields should have all trailing blanks converted to nulls (binary zeros) at the next output.

MENUMAP

If coded, specifies that PF13-PF24 will be returned by XMENU/E as PF01-PF12.

MENUPA1

If coded, specifies that the PA1 key will be trapped by XMENU/E.

MENUCLER

If coded, specifies that the screen is cleared each time this menu is displayed. This should normally never be used.

MENUUPCS

If coded, specifies that all input should be converted to upper case.

Return Codes

8 Invalid parameter passed to the MENUFLGS macro.

MPAGFLGS - Define MPGD2S or MPGS2D flags

[label]	**MPAGFLGS**	[MPAGDFLG,] [MPAGDRFL,] [MPAGADSP,] [MPAGCRSP,] [MPAGWER,] [MPAGWERA,] [MPAGDTYP,] [MPAGAUTO,] [MPAGNTFG] [] [DEFINE]

MPAGFLGS is an assembler macro used to create the flag passed to the "MPGD2S" or "MPGS2D" subroutines in their "flags" parameter. This flag should be located in read/write storage.

Each parameter represents a bit within the flag definition. To set the appropriate flag, code the appropriate parameter. The parameters can be specified in any order. If no parameters are coded, the flag is initialized to all zeros. If "DEFINE" is coded, only bit equates are defined, no flag is defined.

Parameters

DEFINE

If coded, specifies that the macro generates the assembler equates for the bits contained in the flag. If "DEFINE" is specified, it must be the first and only parameter passed to the macro. "DEFINE" does not generate any storage - a second call to the macro with the other parameters must be made to generate the flag data.

MPAGDFLG

Specifies that the menu should be displayed now.

MPAGDRFL

Specifies that the command line parameters ("cmdbuf," "cmdbufl" and "cmdimpl") are passed and that if the menu is displayed, wait for user input.

MPAGADSP

Normally only set by XMENU/E. Specifies that XMENU/E displayed the menu.

MPAGCRSP

If coded, specifies that information has been passed in "linedefs" to direct this output to ("MPGD2S") or input from ("MPGS2D") a specific field by row/column.

MPAGWER

If coded, clear all columnar data fields to nulls (binary zeros) before moving any output to a column. This has no effect under "MPGS2D."

MPAGWERA

If coded, reset all columnar data field attributes to their values when the menu was created by "MPGINT." This has no effect under "MPGS2D."

MPAGDTYP

If coded, the data passed by "MPGD2S" contains data type information. This has no effect under "MPGS2D."

MPAGAUTO

If coded, specifies that XMENU/E should display the menu if data is being moved to the last row of columnar fields. This has no effect under "MPGS2D."

MPAGNTFG

If coded, and if this is a "line mode" menu, the data is moved to the menu and the line is saved but no physical output takes place.

Return Codes

8 Invalid parameter passed to the MPAGFLGS macro.

MPAGINIT - Define MPGINT flags

[label]	**MPAGINIT**	[MPAGNOCM,] [MPAGCTOP,] [MPAGNPRE,] [MPAGRPRE,] [MPAGCCOL,] [MPAGSPTL,] [MPAGNOCL,] [MPAGSPCN,] [MPAGTTAT,] [MPAGDEFS,] [MPAGWIND,] [MPAGIGNW,] [] [DEFINE]

MPAGINIT is an assembler macro used to create the flag passed to the "MPGINT" subroutine in its "flags" parameter.

Each parameter represents a bit within the flag definition. To set the appropriate flag, code the appropriate parameter. The parameters can be specified in any order. If no parameters are coded, the flag is initialized to all zeros. If "DEFINE" is coded, only bit equates are defined, no flag is defined.

Parameters

DEFINE

If coded, specifies that the macro generates the assembler equates for the bits contained in the flag. If "DEFINE" is specified, it must be the first and only parameter passed to the macro. "DEFINE" does not generate any storage - a second call to the macro with the other parameters must be made to generate the flag data.

MPAGNOCM

Specifies that a command line should not be created.

MPAGCTOP

Specifies that the command line should be placed before the columnar data. If not coded, the command line (if any) follows the columnar data.

MPAGNPRE

Specifies that a command line prefix/suffix area should not be created.

MPAGRPRE

Specifies that the command line prefix/suffix area should follow the command input area (i.e. be a suffix area). If not coded, the prefix/suffix area (if any) precedes the command input area (i.e. is a prefix area).

MPAGCCOL

If columnar data is shorter than the screen column size, center the data on the screen. If not specified, the columns are left justified on the screen.

MPAGSPTL

If coded, the last top title line follows the top command line. This allows the last top title line to be used as a column heading line.

MPAGNOCL

Create a "line mode" menu (no fields within columnar area).

MPAGSPCN

If coded, specifies that specific row/column starting field names have been passed as part of the MPGINT "coldefs" parameter.

MPAGTATT

If coded, the top and bottom title lines contain user specified fields.

MPAGWIND

If coded, a 16 character window name follows these flags.

MPAGIGNW

If coded, don't use window support, even if MPAGWIND is coded.

MPAGDEFS

If coded, the menu is created equal to the terminal's 3277 compatibility size. If not coded, the terminal's alternate screen size is used. The terminal sizes for the running terminal can be obtained from the "MSCRSZ" subroutine.

Return Codes

8 Invalid parameter passed to the MPAGINIT macro.

MPRTFLGS - Define MPRINT/MPRNOH/MPRLNE flags

[label]	**MPRTFLGS**	[PRTFILE,] [PRTSCRPT,] [PRTNOH,] [PRTDARK,] [PRTFNAMP,] [PRTNOINP] [] [DEFINE]

MPRTFLGS is an assembler macro used to create the flag passed to the "MPRINT," "MPRNOH" and "MPRLNE" subroutines in their "pflags" parameter.

Each parameter represents a bit within the flag definition. To set the appropriate flag, code the appropriate parameter. The parameters can be specified in any order. If no parameters are coded, the flag is initialized to all zeros. If "DEFINE" is coded, only bit equates are defined, no flag is defined.

Parameters

DEFINE

If coded, specifies that the macro generates the assembler equates for the bits contained in the flag. If "DEFINE" is specified, it must be the first and only parameter passed to the macro. "DEFINE" does not generate any storage - a second call to the macro with the other parameters must be made to generate the flag data.

PRTFILE

If coded, specifies that the generated output should go into a DASD file. If not coded, output goes to the virtual printer.

PRTSCRPT

If coded, specifies that the output is generated in IBM Document Composition Facility (SCRIPT) format.

PRTNOH

If coded, specifies that the menu is to be output without a leading header nor inside a numbered box. This is assumed under "MPRNOH." This has no effect under "MPRLNE."

PRTDARK

If coded, specifies that the contents of DARK fields are printed with the menu. If not specified, the DARK fields are printed as blanks. This has no effect under "MPRLNE."

PRTFNAMP

If coded, specifies that a specific file name has been passed in "pfname." If not specified, the file name used is the menu name.

PRTNOINP

If coded, specifies that the last user input is not output with the menu. If not specified, the last user input is output with the menu.

Return Codes

8 Invalid parameter passed to the MPRTFLGS macro.

UBUFFLGS - Define MPGD2S data type flags

[label]	**UBUFFLGS**	[UBUFHEX,] [UBUFDEC,] [UBUFIGNA,] [UBUFRJ,] [UBUFZS] [] [DEFINE]

UBUFFLGS is an assembler macro used to create the flag passed to the "MPGD2S" subroutine in its "BUF1L...BUFnL" parameters.

Each parameter represents a bit within the flag definition. To set the appropriate flag, code the appropriate parameter. The parameters can be specified in any order. If no parameters are coded, the flag is initialized to all zeros. If "DEFINE" is coded, only bit equates are defined, no flag is defined.

Parameters

DEFINE

If coded, specifies that the macro generates the assembler equates for the bits contained in the flag. If "DEFINE" is specified, it must be the first and only parameter passed to the macro. "DEFINE" does not generate any storage - a second call to the macro with the other parameters must be made to generate the flag data.

UBUFHEX

If coded, specifies that the passed data should be converted from binary to printable hexadecimal.

UBUFDEC

If coded, specifies that the passed data should be converted from binary to printable decimal. No sign is generated.

UBUFIGNA

If coded, specifies that the passed data does not contain a leading attribute definition. This flag is used when the binary data to be converted may look like a valid attribute.

UBUFRJ

If coded, specifies that the converted data should be right justified in its menu field. If not coded, data is left justified.

UBUFZS

If coded, specifies that the converted data should be leftmost zero suppressed.

Return Codes

8 Invalid parameter passed to the UBUFFLGS macro.

Attribute Definitions

Attributes - Bit definitions of 327x attribute

The following table defines the 327x attributes used in the XMENU/E subroutines "MPGINT" and "MPGD2S." Attributes can also be defined by using the "MENUATT" macro described earlier.

Byte	Contents
0	SFE attribute byte (X'29')
1	Count of following pairs (X'04')
2	Color type (X'42')
3	Color value (X'F1' BLUE, X'F2' RED, X'F3' PINK, X'F4' GREEN, X'F5' TURQUOISE, X'F6' YELLOW, X'F7' WHITE, X'00' Default)
4	Highlighting type (X'41')
5	Highlighting value (X'00' None, X'F1' BLINK, X'F2' REVERSE, X'F3' UNDERSCORE)
6	Logical symbol set type (X'43')
7	Logical symbol set value (X'00' default, X'40-EF' logical symbol set)
8	Standard attribute type (X'C0')
9	Standard attribute value (a bit definition - see below).

The definition for the standard 327x attribute is as follows:

FLAG	VALUE	RESULT IF SET
BIT0	80	See the following table.
BIT1	40	This bit must always be on.
UNPROT	00	The field is unprotected.
NUMERIC	10	The field allows numeric input only.
PROT	20	The field is protected.
SKIP	30	The field is protected with auto SKIP.
BRIGHT	08	The field is bright.
DIM	00	The field is dim.
DARK	0C	The field is dark (invisible).
LGHTPEN	04	The field is dim and light pen detectable.
MDT	01	Modified data tag set.

The values above are in hexadecimal. The value of "BIT0" can be determined from the following table:

V	A	V	A	V	A	V	A
40	00	50	10	60	20	F0	30
C1	01	D1	11	61	21	F1	31
C2	02	D2	12	E2	22	F2	32
C3	03	D3	13	E3	23	F3	33
C4	04	D4	14	E4	24	F4	34
C5	05	D5	15	E5	25	F5	35
C6	06	D6	16	E6	26	F6	36
C7	07	D7	17	E7	27	F7	37
C8	08	D8	18	E8	28	F8	38
C9	09	D9	19	E9	29	F9	39
4A	0A	5A	1A	6A	2A	7A	3A
4B	0B	5B	1B	6B	2B	7B	3B
4C	0C	5C	1C	6C	2C	7C	3C
4D	0D	5D	1D	6D	2D	7D	3D
4E	0E	5E	1E	6E	2E	7E	3E
4F	0F	5F	1F	6F	2F	7F	3F

All values are given in hexadecimal. Once you have combined the bits to get an attribute ("A") value, look it up in the table above to get the 327x ("V") value. This is the value you would pass as a standard attribute value.

If a terminal does not have the numeric input feature, "numeric" fields will allow any input.

Chapter 4. Sample programs

What the sample programs do

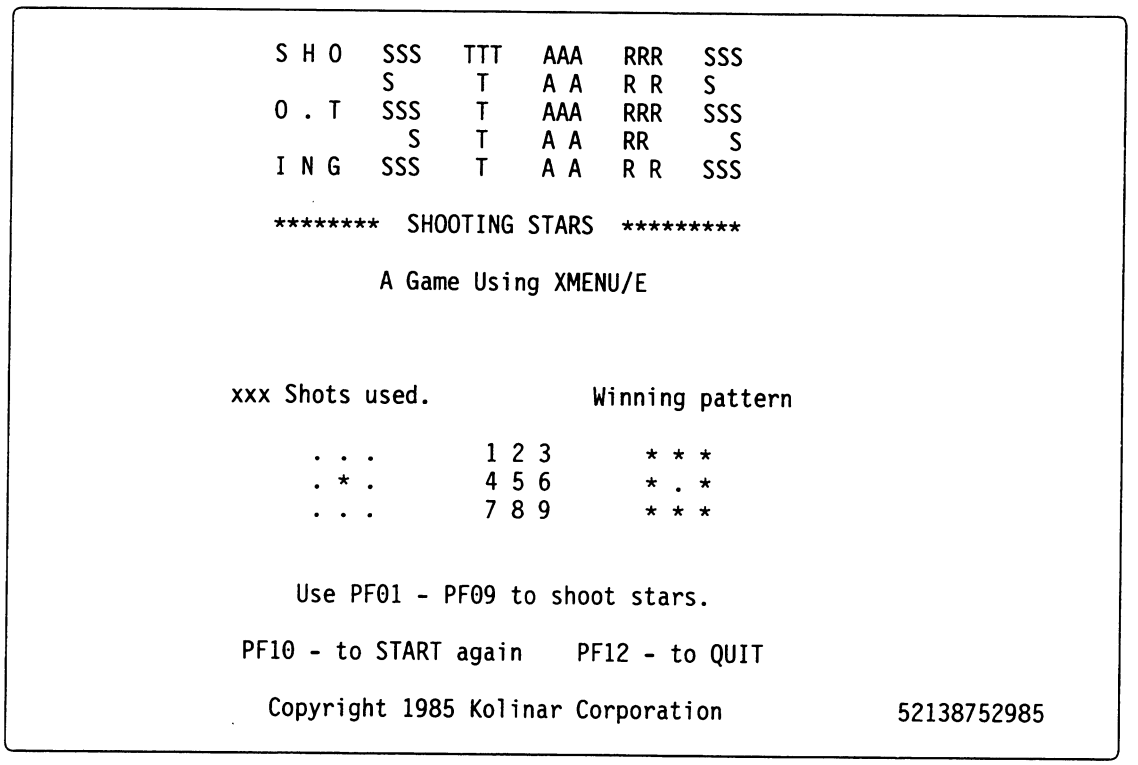
The following pages show an example VM Systems Group Internal Tools application developed in Assembler, COBOL/VS, EXEC 2, REXX, FORTRAN/VS, PASCAL/VS and PL/I. Output from the XMENU program is also given to show you what the menu and XMENU printed output look like.

All of the programs perform the same function. Each is an implementation of a computer game called "shooting stars."

The game is played by pressing Program Function keys to change the pattern of the stars ("*") and black holes ("."). You "win" if you create the pattern shown on the menu. You can only select Program Function keys that map to stars (rather than black holes). When a star is selected, it changes to a black hole, and all of its neighbors reverse their current state. If everything turns to black holes, you "lose." You can exit the program by pressing PF12. You can restart the game by pressing PF10. It is possible to "win" in eleven moves.

Each of these examples show menu loading, moving of data to and from the menu, cursor positioning, etc. While the programs are simple, they show that it is simple and straightforward to use VM Systems Group Internal Tools under the various programming languages.

The menu displayed by the sample programs



The menu's field descriptions

```
Field        starts on line  1 column   1 , is   79 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field        starts on line  2 column   1 , is   79 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field        starts on line  3 column   1 , is  239 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field        starts on line  6 column   1 , is   79 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field        starts on line  7 column   1 , is   79 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field        starts on line  8 column   1 , is  255 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field MSG    starts on line 11 column  17 , is   47 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field        starts on line 11 column  65 , is  102 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field MOVES  starts on line 13 column   8 , is   14 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color WHITE     , Highlight BLINK    , Logical symbol set 00
Field        starts on line 13 column  23 , is   22 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
             Color BLUE      , Highlight DEFAULT  , Logical symbol set 00
Field        starts on line 13 column  46 , is   15 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
             Color DEFAULT   , Highlight UNDERSCORE, Logical symbol set 00
Field        starts on line 13 column  62 , is  122 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field DU1    starts on line 15 column  25 , is    1 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color YELLOW    , Highlight BLINK    , Logical symbol set 00
Field DU2    starts on line 15 column  27 , is    1 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color PINK      , Highlight BLINK    , Logical symbol set 00
Field DU3    starts on line 15 column  29 , is    1 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color TURQUOISE , Highlight BLINK    , Logical symbol set 00
Field        starts on line 15 column  31 , is    6 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field        starts on line 15 column  38 , is    5 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
             Color DEFAULT   , Highlight REVERSE  , Logical symbol set 00
Field        starts on line 15 column  44 , is    5 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
             Color DEFAULT   , Highlight DEFAULT  , Logical symbol set 00
Field        starts on line 15 column  50 , is   54 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color GREEN     , Highlight BLINK    , Logical symbol set 00
Field DU4    starts on line 16 column  25 , is    1 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color RED       , Highlight BLINK    , Logical symbol set 00
Field DU5    starts on line 16 column  27 , is    1 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color WHITE     , Highlight BLINK    , Logical symbol set 00
Field DU6    starts on line 16 column  29 , is    1 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
             Color BLUE      , Highlight BLINK    , Logical symbol set 00
```

```
Field       starts on line 16 column  31 , is   6 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 16 column  38 , is   5 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight REVERSE , Logical symbol set 00
Field       starts on line 16 column  44 , is   5 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 16 column  50 , is  54 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color GREEN    , Highlight BLINK   , Logical symbol set 00
Field DU7   starts on line 17 column  25 , is   1 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color GREEN    , Highlight BLINK   , Logical symbol set 00
Field DU8   starts on line 17 column  27 , is   1 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color YELLOW   , Highlight BLINK   , Logical symbol set 00
Field DU9   starts on line 17 column  29 , is   1 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color RED      , Highlight BLINK   , Logical symbol set 00
Field       starts on line 17 column  31 , is   6 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 17 column  38 , is   5 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight REVERSE , Logical symbol set 00
Field       starts on line 17 column  44 , is   5 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 17 column  50 , is   5 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color GREEN    , Highlight BLINK   , Logical symbol set 00
Field       starts on line 17 column  56 , is 211 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 20 column  28 , is  11 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 20 column  40 , is 139 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 22 column  20 , is   4 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 22 column  25 , is   4 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 22 column  30 , is   5 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 22 column  36 , is   8 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 22 column  45 , is   4 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 22 column  50 , is   4 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 22 column  55 , is   4 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 22 column  60 , is 121 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 24 column  22 , is  14 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 24 column  37 , is   7 chars long; field attributes are: PROT   , BRIGHT , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
Field       starts on line 24 column  45 , is  35 chars long; field attributes are: PROT   , DIM    , NOSKIP, NOMDT
            Color DEFAULT  , Highlight DEFAULT , Logical symbol set 00
```

Assembler Sample Program

```
SAM1ASM  SAMPLE PROGRAM FOR XMENU                                                    PAGE    2

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                    ASM H V 02 11.27 12/28/84

                                     1 *.                                                          00001000
                                     2 *$ MACLIB DMSSP CMSLIB OSMACRO                              00002000
                                     3 *$ TXTLIB XMENUSUB                                          00003000
                                     4 *.                                                          00004000
                                     5 *****                                                       00005000
                                     6 *    COPYRIGHT 1984 - VM SYSTEMS GROUP.                     00006000
                                     7 *    THIS PROGRAM IS LICENSED MATERIAL AND IS PROPERTY OF   00007000
                                     8 *    VMSG - ALL RIGHTS RESERVED                             00008000
                                     9 *****                                                       00009000
                                    10 *                                                           00010000
                                    11 *        XMENU ASSEMBLER SAMPLE PROGRAM                     00011000
                                    12 *        THE ASSEMBLER IMPLEMENTATION OF THE "STARS" GAME.  00012000
                                    13 *        SEE THE XMENU SUBROUTINE REFERENCE FOR GAME RULES. 00013000
                                    14 *.                                                          00014000
000000                              15 SAM1ASM  CSECT ,                                            00015000
                                    17          PRINT ON,NOGEN       ONLY STANDARD SYSTEM MACROS CALLED 00017000
                                    19          EXTRN MLOAD,MPFMAP,MALARM,MDSPRD,MCVTBS,MEXIT,MLD2S,MPCUR 00019000
                         00000      21          USING START,R15                                   00021000
                                    22          REGEQU ,                                          00022000

000000 90EC D00C         0000C      75 START    STM   R14,R12,12(R13)     SAVE REGISTERS       00024000
000004 47F0 F020         00020      76          B     SKIP                                        00025000
000008 E2C1D4F1C1E2D440             78          DC    CL9'SAM1ASM'   EYECATCHERS                00027000
                                    79          DC    CL9'&SYSDATE'                             00028000
                                    80          DC    CL5'&SYSTIME'                             00029000
00001F 00
000020 18CF                         82 SKIP     LR    R12,R15        SET UP BASE REGISTER       00031000
                                    84          DROP  R15                                       00033000
                         00000      85          USING START,R12                                 00034000
000022 4110 C3DC         003DC      87          LA    R1,SAVEAREA    GET AN R13 SAVEAREA        00036000
000026 50D1 0004         00004      89          ST    R13,4(R1)      FORWARD AND                00038000
00002A 501D 0008         00008      90          ST    R1,8(R13)      BACKWARD POINTERS          00039000
00002E 9801 D014         00014      92          LM    R0,R1,20(R13)  RESTORE OUR CALLERS REGISTERS 00041000
000032 58DD 0008         00008      93          L     R13,8(R13)     GET NEW SAVE AREA          00042000

                                    95          CALL  MCLSCR,(NUMBER,RC),VL         CLEAR THE SCREEN 00044000
000052 4770 C1FC         001FC     108          BNE   NOT3270        EXIT IF NOT A 3270 TERMINAL 00045000

                                   110          CALL  MLOAD,(NUMBER,RC,MENU),VL     LOAD THE MENU 00047000
000076 4770 C254         00254     124          BNE   NOMENU         COMPLAIN IF MENU NOT AROUND 00048000

                                   126          CALL  MPFMAP,(NUMBER,RC,ON),VL                    00050000
                                   140 *                              TRANSLATE PF13-24 TO PF01-12 00051000
```

Assembler Sample

```
SAM1ASM  SAMPLE PROGRAM FOR XMENU                                                    PAGE    3

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                        ASM H V 02 11.27 12/28/84

00009A 1B33                       142 RESTART  SR    R3,R3          NO SHOTS                                  00053000
00009C D208 C496 C4B4 00496 004B4 143          MVC   UNVRS(9),=C'....*....'    START PATTERN               00054000

                                  145 DISPLAY  CALL  MALARM,(NUMBER,RC,OFF),VL    CLEAR ALARM FLAG           00056000

                                  160 ERRDISP  CALL  MLD2S,(NUMBER,RC,COUNT,FIELDS,BUFFERS,LENS,RLENS,RCS),VL 00058000
                                  179 *                          MOVE DATA FROM BUFFERS TO MENU FIELDS      00059000

                                  181          CALL  MDSPRD,(NUMBER,RC,KEY),VL    OUTPUT/WAIT FOR RESPONSE   00061000

000116 D507 C424 C4A0 00424 004A0 196          CLC   KEY(8),=CL8'PF12'  QUIT?                                00063000
00011C 4780 C1D0            001D0 197          BE    ALLDONE          YES                                    00064000
000120 D507 C424 C4A8 00424 004A8 198          CLC   KEY(8),=CL8'PF10'  RESTART?                             00065000
000126 4780 C09A            0009A 199          BE    RESTART          YES                                    00066000
00012A D502 C424 C4BD 00424 004BD 200          CLC   KEY(3),=C'PF0' MOVE KEY?                                00067000
000130 4770 C0A4            000A4 201          BNE   DISPLAY          NO                                     00068000

                                  203          CALL  MALARM,(NUMBER,RC,ON),VL SET ALARM FLAG                 00070000

000152 4320 C427            00427 218          IC    R2,KEY+3         GET PF NUMBER                          00072000
000156 5420 C4B0            004B0 219          N     R2,=XL4'F'       REDUCE TO NUMBER                       00073000
00015A 4142 C495            00495 220          LA    R4,UNVRS-1(R2) POINT TO SELECTED POSITION              00074000
00015E 8920 0003            00003 221          SLL   R2,3             TIMES 8                                00075000
000162 4152 C2A0            002A0 222          LA    R5,FIELDS(R2)                                           00076000

                                  224          CALL  MPCUR,(NUMBER,RC,(R5)),VL    POSITION OUTPUT CURSOR     00078000

00018E 954B 4000      00000       240          CLI   0(R4),C'.'       OK TO SHOOT?                           00080000
000192 4780 C0C4            000C4 241          BE    ERRDISP          COMPLAIN                               00081000
000196 8920 0001            00001 243          SLL   R2,1           TIMES 16                                 00083000
00019A 4122 C2E0            002E0 244          LA    R2,GALAXY-16(R2)   SELECT CORRECT GALAXY                00084000
00019E D708 C496 2000 00496 00000 245          XC    UNVRS(9),0(R2)   FLIP AS NEEDED                        00085000
0001A4 4130 3001            00001 246          LA    R3,1(,R3)        COUNT SHOTS                           00086000
0001A8 5030 C42C            0042C 247          ST    R3,NMOVES        SAVE FOR CALL                         00087000

                                  249          CALL  MCVTBS,(NMOVES,RC,MOVES+6),VL CONVERT BINARY TO         00089000
                                  263 *                ZERO SUPPRESSED DECIMAL                              00090000
0001CA 47F0 C0A4            000A4 265          B     DISPLAY                                                 00092000
```

```
SAM1ASM  SAMPLE PROGRAM FOR XMENU                                                    PAGE    4

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                        ASM H V 02 11.27 12/28/84

                                    267 *
                                    268 *        PROGRAM EXIT CODE...                                          00094000
                                    269 *                                                                      00095000
                                    270 ALLDONE  CALL  MEXIT,(NUMBER,RC),VL         PURGE THE MENU             00096000
                                                                                                               00097000
0001EA 1BFF                         284 EXIT     SR    R15,R15          CLEAR THE RETURN CODE                  00099000

0001EC 58DD 0004          00004     286 ERREXIT  L     R13,4(R13)       RESTORE OLD SAVE AREA                  00101000
0001F0 58ED 000C          0000C     288          L     R14,12(R13)      RESTORE OTHER REGISTERS                00103000
0001F4 980C D014          00014     289          LM    R0,R12,20(R13)   ...                                    00104000
0001F8 12FF                         291          LTR   R15,R15          SET CC FROM RETURN CODE                00106000
0001FA 07FE                         292          BR    R14              EXIT                                   00107000
                                    294 *                                                                      00109000
                                    295 *        ERROR MESSAGES...                                             00110000
                                    296 *                                                                      00111000
                                    297 NOT3270  WRTERM 'YOU NEED TO BE ON A 327X TERMINAL TO RUN THIS PROGRAM' 00112000
00024C 41F0 0001          00001     308          LA    R15,1            ERROR CODE                             00113000
000250 47F0 C1EC          001EC     309          B     ERREXIT          EXIT                                   00114000

                                    311 NOMENU   WRTERM 'SAMPLE1A MENU CANNOT BE LOADED'                       00116000
00028C 41F0 0002          00002     322          LA    R15,2            ERROR CODE                             00117000
000290 47F0 C1EC          001EC     323          B     ERREXIT          EXIT                                   00118000
```

```
SAM1ASM  SAMPLE PROGRAM FOR XMENU                                                                    PAGE    5

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  ASM H V 02 11.27 12/28/84
                                    325 *                                                                    00120000
                                    326 *        READ/ONLY DATA AREAS...                                     00121000
                                    327 *                                                                    00122000
000298                              328          DS    0D                                                    00123000
000298 E2C1D4D7D3C5F1C1             329 MENU     DC    CL8'SAMPLE1A'  THE MENU NAME                          00124000
0002A0 D4D6E5C5E2404040             331 FIELDS   DC    CL8'MOVES',CL8'DU1',CL8'DU2',CL8'DU3',CL8'DU4',CL8'DU5' 00126000
0002D0 C4E4F64040404040             332          DC    CL8'DU6',CL8'DU7',CL8'DU8',CL8'DU9'                   00127000
                                    333 *                          MLD2S,MLS2D FIELD NAMES                   00128000
0002F0 1717001717000000             335 GALAXY   DC    AL1(0,0,X,0,0,X,X,X,X),7AL1(X)                        00130000
000300 1717170000000000             336          DC    AL1(0,0,0,X,X,X,X,X,X),7AL1(X)                        00131000
000310 0017170017170000             337          DC    AL1(X,0,0,X,0,0,X,X,X),7AL1(X)                        00132000
000320 1700001700001700             338          DC    AL1(0,X,X,0,X,X,0,X,X),7AL1(X)                        00133000
000330 0017001717170017             339          DC    AL1(X,0,X,0,0,0,X,0,X),7AL1(X)                        00134000
000340 0000170000170000             340          DC    AL1(X,X,0,X,X,0,X,X,0),7AL1(X)                        00135000
000350 0000001717001717             341          DC    AL1(X,X,X,0,0,X,0,0,X),7AL1(X)                        00136000
000360 0000000000001717             342          DC    AL1(X,X,X,X,X,X,0,0,0),7AL1(X)                        00137000
000370 0000000017170017             343          DC    AL1(X,X,X,X,0,0,X,0,0),7AL1(X)                        00138000

000380 00000000                     345 OFF      DC    F'0'           USED FOR FLAGS                         00140000
000384 00000001                     346 ON       DC    F'1'           USED FOR FLAGS                         00141000
000388 0000000A                     348 COUNT    DC    F'10'          MLS2D,MLD2S NUMBER OF FIELDS           00143000
00038C 0000048800000496             349 BUFFERS  DC    A(MOVES,UNVRS+0,UNVRS+1,UNVRS+2,UNVRS+3)              00144000
0003A0 0000049A0000049B             350          DC    A(UNVRS+4,UNVRS+5,UNVRS+6,UNVRS+7,UNVRS+8)            00145000
                                    351 *                          MLD2S,MLS2D BUFFER ADDRESSES              00146000
0003B4 0000000E00000001             352 LENS     DC    F'14',9F'1'    MLD2S,MLS2D BUFFER LENGTHS             00147000
                                    354 *                                                                    00149000
                                    355 *        READ/WRITE DATA AREAS...                                    00150000
                                    356 *                                                                    00151000
0003DC                              357 SAVEAREA DS    18F            R13 SAVEAREA                           00152000

000424                              359 KEY      DS    CL8            THE KEY PRESSED BY THE USER            00154000
00042C                              360 NMOVES   DS    F              NUMBER OF MOVES                        00155000
000430                              361 NUMBER   DS    F              MENU NUMBER ASSIGNED BY XMENU          00156000
000434                              362 RC       DS    F              RETURN CODE FROM XMENU CALLS           00157000
000438                              363 RLENS    DS    10F            MLD2S,MLS2D RETURNED LENGTHS           00158000
000460                              364 RCS      DS    10F            MLD2S,MLS2D RETURNED RETURN CODES      00159000

000488 4040404040404040             366 MOVES    DC    CL14'            0'                                   00161000
000496 4B4B4B4B5C4B4B4B             367 UNVRS    DC    CL9'....*....'                                        00162000

                         00017      369 O        EQU   X'17'                                                 00164000
                         00000      370 X        EQU   X'00'                                                 00165000

0004A0                              372          LTORG ,                                                     00167000
0004A0 D7C6F1F240404040             373                =CL8'PF12'
0004A8 D7C6F1F040404040             374                =CL8'PF10'
0004B0 0000000F                     375                =XL4'F'
0004B4 4B4B4B4B5C4B4B4B             376                =C'....*....'
0004BD D7C6F0                       377                =C'PF0'

000000                              379          END   START                                                 00169000
```

C/370 Sample Program

```
5688040 R01 M01 IBM C/370                    SAM1C    C        A1              11/01/89 10:12:35     Page     1

LINE  STMT                              * * * * *  S O U R C E  * * * * *                                SEQNO INCLUDE
       *---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9-------*
   1       | /*                                                                                 */ |      1
   2       | /*       Copyright 1989 VM Systems Group Corporation                               */ |      2
   3       | /*       All Rights Reserved                                                       */ |      3
   4       | /*       C implementation of "Stars" game.                                         */ |      4
   5       | /*       See the XMENU subroutine reference for game rules.                        */ |      5
   6       | /*                                                                                 */ |      6
   7       |                                                                                       |      7
   8       |#pragma linkage (MLOAD,OS)                                                             |      8
   9       |void MLOAD(int*, int*, char*);                                                         |      9
  10       |#pragma linkage (MDSPRD,OS)                                                            |     10
  11       |void MDSPRD(int*, int*, char*);                                                        |     11
  12       |#pragma linkage (MEXIT,OS)                                                             |     12
  13       |void MEXIT(int*, int*);                                                                |     13
  14       |#pragma linkage (MALARM,OS)                                                            |     14
  15       |void MALARM(int*, int*, int);                                                          |     15
  16       |#pragma linkage (MD2SCR,OS)                                                            |     16
  17       |void MD2SCR(/* int*, int*, char*, char*, int, int* */);                                |     17
  18       |                                                                                       |     18
  19       |#include [stdarg.h]                                                                    |     19
  20       |#include [stdio.h]                                                                     |     20
  21       |#include [string.h]                                                                    |     21
  22       |                                                                                       |     22
  23       | /*                                                                                 */ |     23
  24       | /* Declare data types                                                              */ |     24
  25       | /*                                                                                 */ |     25
  26       |typedef struct {int  moves;                                                            |     26
  27       |                 char galaxy[9];                                                       |     27
  28       |                 short int  fos_galaxy;                                                |     28
  29       |                } GAMEINFO;                                                            |     29
  30       |                                                                                       |     30
  31       |union PFKEY   {char c[8];                                                              |     31
  32       |               int i;};                                                                |     32
  33       | /*                                                                                 */ |     33
  34       | /* Declare Constants                                                               */ |     34
  35       | /*                                                                                 */ |     35
  36       |const int ON  = 1;                                                                     |     36
  37       |const int OFF = 0;                                                                     |     37
  38       |const int F1  = 1;                                                                     |     38
  39       |const int F14 = 14;                                                                    |     39
  40       |                                                                                       |     40
  41       | /*                                                                                 */ |     41
  42       | /* Declare Static Storage                                                          */ |     42
  43       | /*                                                                                 */ |     43
  44       |static int   NUMBER, RETCODE;                                                          |     44
  45       |static int   RLEN;                                                                     |     45
  46       |static short int FOS_play[10] = {0x0800,                                               |     46
  47       |                                 0xD800, 0XE000, 0x6C00,                               |     47
  48       |                                 0x9200, 0x5D00, 0x2480,                               |     48
  49       |                                 0x1B00, 0x0380, 0x0D80};                              |     49
  50       |static union PFKEY PF10 = "PF10";                                                      |     50
  51       |static union PFKEY PF12 = "PF12";                                                      |     51
  52       |static char  MENUNAME[9] = "SAMPLE1A";                                                 |     52
  53       |static char *DFIELDS[9]= {"DU1     ","DU2     ","DU3     ",                           |     53
  54       |                          "DU4     ","DU5     ","DU6     ",                           |     54
  55       |                          "DU7     ","DU8     ","DU9     "};                          |     55
```

```
5688040 R01 M01 IBM C/370                    SAM1C    C        A1                  11/01/89 10:12:35    Page     2

LINE  STMT                              * * * * *  S O U R C E  * * * * *                                SEQNO INCLUDE
      *---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9-------*
  56        |static union PFKEY KEYP;                                                                |     56
  57        |static char  SHOTS[15];                                                                 |     57
  58        |                                                                                        |     58
  59        |static GAMEINFO game_info;                                                              |     59
  60        |                                                                                        |     60
  61        |int main(ac, av)                                                                        |     61
  62        |    int  ac;                                                                            |     62
  63        |    char *av[];                                                                         |     63
  64        |{                                                                                       |     64
  65        |  void newgame( void );                                                                 |     65
  66        |  void display( void );                                                                 |     66
  67        |  void shootit( void );                                                                 |     67
  68        |                                                                                        |     68
  69        |  int starnum;                                                                          |     69
  70        |                                                                                        |     70
  71     1  |  for (MCLSCR(&NUMBER, &RETCODE),                                                       |     71
  72        |       MLOAD(&NUMBER, &RETCODE, MENUNAME),                                              |     72
  73        |       MPFMAP(&NUMBER, &RETCODE, &ON),                                                  |     73
  74        |       newgame();                                                                       |     74
  75        |                                                                                        |     75
  76        |       display(), KEYP.i != PF12.i;                                                     |     76
  77        |                                                                                        |     77
  78     2  |      ) if (KEYP.i == PF10.i)                                                           |     78
  79     3  |          newgame();                                                                    |     79
  80        |        else                                                                            |     80
  81     4  |          shootit();                                                                    |     81
  82        |                                                                                        |     82
  83     5  |  MEXIT(&NUMBER, &RETCODE);                                                             |     83
  84        |}                                                                                       |     84
  85        |                                                                                        |     85
  86        |void display( void )                                                                    |     86
  87        |{                                                                                       |     87
  88        |  int i;                                                                                |     88
  89        |                                                                                        |     89
  90     6  |  sprintf(SHOTS, "%14d", game_info.moves);                                              |     90
  91     7  |  MD2SCR(&NUMBER, &RETCODE, "MOVES    ", SHOTS, F14, &RLEN);                            |     91
  92        |                                                                                        |     92
  93     8  |  for (i= 0; i[9; i++)                                                                  |     93
  94     9  |    MD2SCR(&NUMBER, &RETCODE, DFIELDS[i], &game_info.galaxy[i], F1,                     |     94
  95        |           &RLEN);                                                                      |     95
  96        |                                                                                        |     96
  97    10  |  MDSPRD(&NUMBER, &RETCODE, KEYP.c);                                                    |     97
  98        |}                                                                                       |     98
  99        |                                                                                        |     99
 100        |void newgame( void )                                                                    |    100
 101        |{                                                                                       |    101
 102        |  int i;                                                                                |    102
 103        |                                                                                        |    103
 104    11  |  MALARM(&NUMBER, &RETCODE, OFF);                                                       |    104
 105    12  |  game_info.moves= 0;                                                                   |    105
 106    13  |  game_info.fos_galaxy= FOS_play[0];                                                    |    106
 107        |  strcpy(game_info.galaxy, "....*....");                                                |    107
 108        |}                                                                                       |    108
 109        |                                                                                        |    109
 110        |void shootit( void )                                                                    |    110
```

```
5688040 R01 M01 IBM C/370                    SAM1C    C        A1                  11/01/89 10:12:35    Page      3

LINE  STMT                              * * * * *  S O U R C E  * * * * *                                  SEQNO INCLUDE
          *---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9-------*
 111        |{                                                                                            |  111
 112        |  int i;                                                                                     |  112
 113        |  short int j;                                                                               |  113
 114        |                                                                                             |  114
 115    15  |  i= KEYP.i & 0x0000000F;                                                                    |  115
 116    16  |  if ((KEYP.c[1]=='F') && (i[10) && (game_info.galaxy[i-1]=='*'))                            |  116
 117        |    {                                                                                        |  117
 118    17  |    MALARM(&NUMBER, &RETCODE, OFF);                                                          |  118
 119    18  |    game_info.moves= ++game_info.moves;                                                      |  119
 120    19  |    game_info.fos_galaxy= game_info.fos_galaxy ¬ FOS_play[i];                                |  120
 121        |                                                                                             |  121
 122    20  |    j= game_info.fos_galaxy;                                                                 |  122
 123    21  |    for (i=0; i[10; i++)                                                                     |  123
 124        |      {                                                                                      |  124
 125    22  |      if (j[0)                                                                               |  125
 126    23  |        game_info.galaxy[i]= '*';                                                            |  126
 127        |      else                                                                                   |  127
 128    24  |        game_info.galaxy[i]= '.';                                                            |  128
 129    25  |      j= j[[1;                                                                               |  129
 130        |      }                                                                                      |  130
 131        |    }                                                                                        |  131
 132        |  else                                                                                       |  132
 133        |    {                                                                                        |  133
 134    26  |    MALARM(&NUMBER, &RETCODE, ON);                                                           |  134
 135        |    }                                                                                        |  135
 136        |}                                                                                            |  136
```

COBOL/VS Sample Program

```
PP 5740-CB1 RELEASE 2.4                          IBM OS/VS COBOL  JULY  1, 1982

   1                     11.47.32        DEC 28,1984

00001          IDENTIFICATION DIVISION.
00002          PROGRAM-ID.   SAM1COB.
00003          AUTHOR.       NUMBER SIX.
00004          REMARKS.
00005              COPYRIGHT 1985 - VM SYSTEMS GROUP
00006              ALL RIGHTS RESERVED
00007              SAMPLE PROGRAM TO DEMOSRATE THE USE OF THE
00008              XMENU SUBROUTINE PACKAGE.
00009          INSTALLATION.   VM SYSTEMS GROUP.
00010          DATE-WRITTEN.   OCT 1982.
00011          ENVIRONMENT DIVISION.
00012          CONFIGURATION SECTION.
00013          SOURCE-COMPUTER.  IBM-4381-2.
00014          OBJECT-COMPUTER.  IBM-4381-2.
00015
00016          DATA DIVISION.
00017          WORKING-STORAGE SECTION.
00018          77  MENU-NUMBER         PICTURE 9(7) USAGE IS COMPUTATIONAL.
00019          77  MENU-RETURN-CODE    PICTURE 9(7) USAGE IS COMPUTATIONAL.
00020          77  RLEN                PICTURE 9(7) USAGE IS COMPUTATIONAL.
00021          77  I                   PICTURE 9(7) USAGE IS COMPUTATIONAL.
00022          77  J                   PICTURE 9(7) USAGE IS COMPUTATIONAL.
00023          77  K                   PICTURE 9(7) USAGE IS COMPUTATIONAL.
00024          77  TURN-ON             PICTURE 9(7) USAGE IS COMP VALUE IS 1.
00025          77  TURN-OFF            PICTURE 9(7) USAGE IS COMP VALUE IS 0.
00026          77  MENU-NAME           PICTURE X(8) VALUE "SAMPLE1A".
00027          77  DMOVES              PICTURE Z(13)9.
00028          77  MOVES               PICTURE 9(7) USAGE IS COMP VALUE IS 0.
00029          77  QMOVES              PICTURE X(8) VALUE "MOVES".
00030          77  LMOVES              PICTURE 9(7) USAGE IS COMP VALUE IS 14.
00031          77  LDU                 PICTURE 9(7) USAGE IS COMP VALUE IS 1.
00032          77  DU-NAME             PICTURE X(8).
00033          77  DU-VALUE            PICTURE X(1).
00034
00035          01  MENU-KEY.
00036              02 KEY1-3           PICTURE X(3).
00037              02 KEY-4            PICTURE X.
00038              02 KEY-REST         PICTURE X(4).
00039
00040          01  ALL-QDU.
00041              02  QDU1            PICTURE X(8) VALUE "DU1".
00042              02  QDU2            PICTURE X(8) VALUE "DU2".
00043              02  QDU3            PICTURE X(8) VALUE "DU3".
00044              02  QDU4            PICTURE X(8) VALUE "DU4".
00045              02  QDU5            PICTURE X(8) VALUE "DU5".
00046              02  QDU6            PICTURE X(8) VALUE "DU6".
00047              02  QDU7            PICTURE X(8) VALUE "DU7".
00048              02  QDU8            PICTURE X(8) VALUE "DU8".
00049              02  QDU9            PICTURE X(8) VALUE "DU9".
00050          01  ALL-FLDNAME REDEFINES ALL-QDU.
00051              02  FLDNAME OCCURS 9 TIMES PIC X(8).
00052
00053          01  ALL-DU.
00054              02 DU1              PICTURE X VALUE IS ".".
```

```
 2        SAM1COB         11.47.32        DEC 28,1984

00055             02 DU2              PICTURE X VALUE IS ".".
00056             02 DU3              PICTURE X VALUE IS ".".
00057             02 DU4              PICTURE X VALUE IS ".".
00058             02 DU5              PICTURE X VALUE IS "*".
00059             02 DU6              PICTURE X VALUE IS ".".
00060             02 DU7              PICTURE X VALUE IS ".".
00061             02 DU8              PICTURE X VALUE IS ".".
00062             02 DU9              PICTURE X VALUE IS ".".
00063          01  TABLE-DU REDEFINES ALL-DU.
00064             02 DU OCCURS 9 TIMES PICTURE X.
00065
00066          01  ALL-GALAXIES.
00067             02 COBOL-FUN OCCURS 9 TIMES.
00068                03 GALAXY OCCURS 6 TIMES PIC 9(7) USAGE IS COMP.
00069
00070          LINKAGE SECTION.
00071
00072          PROCEDURE DIVISION.
00073              MOVE ZEROS TO ALL-GALAXIES.
00074              MOVE 1 TO GALAXY (1, 1), GALAXY (2, 1), GALAXY (4, 1).
00075              MOVE 2 TO GALAXY (1, 2), GALAXY (2, 2), GALAXY (3, 1)
00076                        GALAXY (5, 1).
00077              MOVE 3 TO GALAXY (2, 3), GALAXY (3, 2), GALAXY (6, 1).
00078              MOVE 4 TO GALAXY (1, 3), GALAXY (4, 2), GALAXY (5, 2)
00079                        GALAXY (7, 1).
00080              MOVE 5 TO GALAXY (1, 4), GALAXY (3, 3), GALAXY (5, 3)
00081                        GALAXY (7, 2), GALAXY (9, 1).
00082              MOVE 6 TO GALAXY (3, 4), GALAXY (5, 4), GALAXY (6, 2)
00083                        GALAXY (9, 2).
00084              MOVE 7 TO GALAXY (4, 3), GALAXY (7, 3), GALAXY (8, 1).
00085              MOVE 8 TO GALAXY (5, 5), GALAXY (7, 4), GALAXY (8, 2)
00086                        GALAXY (9, 3).
00087              MOVE 9 TO GALAXY (6, 3), GALAXY (8, 3), GALAXY (9, 4).
00088              CALL "MCLSCR" USING MENU-NUMBER MENU-RETURN-CODE.
00089              CALL "MLOAD"  USING MENU-NUMBER MENU-RETURN-CODE MENU-NAME.
00090              CALL "MPFMAP" USING MENU-NUMBER MENU-RETURN-CODE TURN-ON.
00091          ACTION-RESTART.
00092              MOVE 0 TO MOVES.
00093              MOVE "....*...." TO ALL-DU.
00094          ACTION-DISPLAY.
00095              CALL "MALARM" USING MENU-NUMBER MENU-RETURN-CODE TURN-OFF.
00096          RAP-HIM.
00097              MOVE MOVES TO DMOVES.
00098              CALL "MD2SCR" USING MENU-NUMBER MENU-RETURN-CODE QMOVES
00099                                  DMOVES      LMOVES           RLEN.
00100              PERFORM DATA-TO-SCREEN , VARYING I FROM 1 BY 1,
00101                                       UNTIL I IS GREATER THEN 10.
00102              CALL "MDSPRD" USING MENU-NUMBER MENU-RETURN-CODE MENU-KEY.
00103              IF MENU-KEY IS EQUAL TO "PF10" THEN GO TO ACTION-RESTART.
00104              IF MENU-KEY IS EQUAL TO "PF12" THEN GO TO END-ACTION.
00105              IF KEY1-3 IS NOT EQUAL TO "PF0" THEN GO TO ACTION-DISPLAY.
00106              CALL "MALARM" USING MENU-NUMBER MENU-RETURN-CODE TURN-ON.
00107              IF MENU-KEY IS EQUAL TO "PF01" MOVE 1 TO I.
00108              IF MENU-KEY IS EQUAL TO "PF02" MOVE 2 TO I.
00109              IF MENU-KEY IS EQUAL TO "PF03" MOVE 3 TO I.
00110              IF MENU-KEY IS EQUAL TO "PF04" MOVE 4 TO I.
00111              IF MENU-KEY IS EQUAL TO "PF05" MOVE 5 TO I.
```

```
  3       SAM1COB         11.47.32        DEC 28,1984

00112            IF MENU-KEY IS EQUAL TO "PF06" MOVE 6 TO I.
00113            IF MENU-KEY IS EQUAL TO "PF07" MOVE 7 TO I.
00114            IF MENU-KEY IS EQUAL TO "PF08" MOVE 8 TO I.
00115            IF MENU-KEY IS EQUAL TO "PF09" MOVE 9 TO I.
00116            CALL "MPCUR" USING MENU-NUMBER MENU-RETURN-CODE DU (I).
00117            IF DU (I) IS NOT EQUAL TO "*" THEN GO TO RAP-HIM.
00118            PERFORM SHOT-STAR, VARYING J FROM 1 BY 1,
00119                                UNTIL J IS GREATER 6.
00120            ADD 1 TO MOVES.
00121            GO TO ACTION-DISPLAY.
00122          DATA-TO-SCREEN.
00123            MOVE FLDNAME (I) TO DU-NAME.
00124            MOVE DU (I) TO DU-VALUE.
00125            CALL "MD2SCR" USING MENU-NUMBER MENU-RETURN-CODE DU-NAME
00126                                DU-VALUE    LDU              RLEN.
00127          SHOT-STAR.
00128            MOVE GALAXY (I, J) TO K.
00129            IF K IS NOT EQUAL TO 0 THEN
00130              IF DU (K) IS EQUAL TO "*"
00131                 THEN MOVE "." TO DU (K)
00132                 ELSE MOVE "*" TO DU (K).
00133          END-ACTION.
00134            CALL "MEXIT" USING MENU-NUMBER MENU-RETURN-CODE.
00135            STOP RUN.
```

EXEC 2 Sample EXEC

```
&TRACE OFF
*
*     COPYRIGHT 1985 VM SYSTEMS GROUP
*     ALL RIGHTS RESERVED
*     SAMPLE EXEC TO DEMONSTRATE USE OF XMENU
*
&MENU = SAMPLE1A
CONTYPE MODEL
&IF &RETCODE = 0 &TYPE You need a 327x terminal to run this EXEC.
&IF &RETCODE = 0 &EXIT
&BEGSTACK 10 * LIFO
5 6 8 9 0 0
7 9 8 0 0 0
4 5 8 7 0 0
3 9 6 0 0 0
2 4 6 8 5 0
1 7 4 0 0 0
2 5 6 3 0 0
1 3 2 0 0 0
2 4 5 1 0 0
. . . . * . . . .
&READ VARS &DU1 &DU2 &DU3 &DU4 &DU5 &DU6 &DU7 &DU8 &DU9
&READ VARS &M11 &M12 &M13 &M14 &M15 &M16
&READ VARS &M21 &M22 &M23 &M24 &M25 &M26
&READ VARS &M31 &M32 &M33 &M34 &M35 &M36
&READ VARS &M41 &M42 &M43 &M44 &M45 &M46
&READ VARS &M51 &M52 &M53 &M54 &M55 &M56
&READ VARS &M61 &M62 &M63 &M64 &M65 &M66
&READ VARS &M71 &M72 &M73 &M74 &M75 &M76
&READ VARS &M81 &M82 &M83 &M84 &M85 &M86
&READ VARS &M91 &M92 &M93 &M94 &M95 &M96
&CLEAR  = CLEAR
&CURSOR = DU5
-INITIAL &MOVES = 0
 &STACK LIFO . . . . * . . . .
 &READ VARS &DU1 &DU2 &DU3 &DU4 &DU5 &DU6 &DU7 &DU8 &DU9
 &CURSOR = DU5
-DISPLAY &ALARM =
 &MSG = &BLANK
 &TEST = &CONCAT OF &DU1 &DU2 &DU3 &DU4 &DU5 &DU6 &DU7 &DU8 &DU9
 &IF &TEST = ****.**** &MSG = &STRING OF You Win!
 &IF &TEST = ......... &MSG = &STRING OF You Loose!
-ERRDISP &STACK MENUCMDS
         &STACK RJUST MOVES
         &IF .&MSG NE . &STACK CHANGE MSG REVERSE
         MENUEXEC &MENU (&ALARM &CLEAR CURSOR &CURSOR MAP EMSG SAVE
         &IF &RETCODE NE 0 &EXIT &RETCODE
         &CLEAR =
&IF &PFK  EQ PF12 &EXIT
&IF &PFK  EQ PF10 &GOTO -INITIAL
&A = &SUBSTR OF &PFK 1 3
&IF &A    NE PF0 &GOTO -DISPLAY
&A = &SUBSTR OF &PFK 4 1
&ALARM = ALARM
&MSG = &STRING OF That wasn't a legal move
&IF &DU&A EQ . &GOTO -ERRDISP
&MSG = &BLANK
&CURSOR = DU&A
&MOVES = &MOVES + 1
&I = 1
-LOOP &X = &CONCAT OF &A &I
      &Y = &M&X
      &IF &Y EQ 0   &GOTO -DISPLAY
      &T = .
      &IF &DU&Y EQ .  &T = *
      &DU&Y = &T
      &I = &I + 1
      &GOTO -LOOP
```

REXX Sample EXEC

```
/*********************************************************************/
/***    5664-167-121   COPYRIGHT - VM SYSTEMS GROUP - 1985         ***/
/***    LICENSED MATERIAL - PROGRAM PROPERTY OF VMSG               ***/
/***    Sample Program                                             ***/
/***    Purpose:       Example of how to use XMENU with REXX       ***/
/***    Description:   See game rules in XMENU Subroutine          ***/
/***                   Reference manual                            ***/
/***    Prerequisites: XMENU, 327x terminal                        ***/
/***    Output:        Full screen display                         ***/
/*********************************************************************/
trace o

        /*  1 2 3 4 5 6 7 8 9 */
flipper.1 = '171700171700000000'X
flipper.2 = '171717000000000000'X
flipper.3 = '001717001717000000'X
flipper.4 = '170000170000170000'X
flipper.5 = '001700171717001700'X
flipper.6 = '000017000017000017'X
flipper.7 = '000000171700171700'X
flipper.8 = '000000000000171717'X
flipper.9 = '000000001717001717'X
        /*  1 2 3 4 5 6 7 8 9 */
menu = 'SAMPLE1A'
j=0
DO WHILE PFK&notsign.=PF12
  du = '....*....'
  PARSE VAR du DU1 2 DU2 3 DU3 4 DU4 5 DU5 6 DU6 7 DU7 8 DU8 9 DU9
  Moves = 0
  "MENUEXEC" menu "( CLEAR CURSOR DU5 MAP SAVE"
  DO WHILE PFK&notsign.=PF10 &PFK&notsign.=PF12
    msg=' '
    Alarm=''
    a= 99
    IF substr(PFK,1,2)= 'PF'
       THEN IF substr(PFK,3,1)='0'
            THEN a= substr(PFK,4,1)
            ELSE IF PFK=PF11
                 THEN IF SUBSTR(CURFNAM,1,2)='DU'
                       THEN a= SUBSTR(CURFNAM,3,1)

    IF a [ 10 THEN DO
      IF VALUE('DU'||a)='.' THEN DO
        msg='Shoot stars with same PF key number'
        Alarm='ALARM'
      END
      ELSE DO
        Moves=Moves+1
        du= BITXOR(du,flipper.a)
        PARSE VAR du DU1 2 DU2 3 DU3 4 DU4 5 DU5 6 DU6 7 DU7 8 DU8 9 DU9
      END
    END
    IF du='.........'
       THEN DO
         msg= 'You LOOSE!!'
         Alarm='ALARM'
       END
    IF du='****.****'
       THEN DO
         msg= 'You WON!!!  The best possible score is 11 shots.'
         Alarm='ALARM'
       END
    queue "MENUCMDS"
    queue "RJUST Moves"
    "MENUEXEC" menu "(" Alarm " CURSOR" Cursor" MAP SAVE"
  END /* do while pfk&notsign.=10 & 12 */
END   /* do while pfk&notsign.=12      */
EXIT
```

FORTRAN/VS Sample Program

```
LEVEL 1.2.0 (SEPT  82)      VS FORTRAN       DATE: DEC 28, 1984    TIME: 12:24:31

REQUESTED OPTIONS (EXECUTE): LANGLVL(77)

REQUESTED OPTIONS (PROCESS):  SC(MLOAD,MEXIT,MDSPRD,MD2SCR,MCLSCR)
                              SC(MPCUR,MALARM,MPFMAP,MCVTBS)

OPTIONS IN EFFECT:  NOLIST NOMAP   XREF NOGOSTMT NODECK   SOURCE   TERM
                    OBJECT FIXED  NOTEST
                    OPTIMIZE(3)  LANGLVL(77) NOFIPS   FLAG(I)
                    NAME(MAIN  )  LINECOUNT(60)

          *....*...1.........2.........3.........4.........5.........6.........7.*.......8

          C     COPYRIGHT 1985 - VM SYSTEMS GROUP
          C     ALL RIGHTS RESERVED
          C     VS/FORTRAN IMPLEMENTATION OF "STARS" GAME
          C     SEE XMENU SUBROUTINE REFERENCE FOR GAME RULES
          C
ISN   1         INTEGER      NUM,RC,GALAXY(6,9),MOVES,RLEN
ISN   2         CHARACTER*14 DMOVE
ISN   3         CHARACTER*9  DUNVRS,EUNVRS
ISN   4         CHARACTER*8  KEY,KEYS(9),TCHAR8,FLDN(9)
ISN   5         CHARACTER*1  UNVRS(9),TCHAR1,EMOVE(14)
ISN   6         EQUIVALENCE  (DMOVE,EMOVE(1)),(EMOVE(7),TCHAR8),(UNVRS(1),DUNVRS)
ISN   7         DATA KEYS/'PF01','PF02','PF03','PF04','PF05','PF06','PF07','PF08',
                *          'PF09'/,DMOVE/'              '/,EUNVRS/'....*....'/
ISN   8         DATA GALAXY/1,2,4,5,0,0,  1,2,3,0,0,0,  2,3,5,6,0,0,
                *           1,4,7,0,0,0,  2,4,5,6,8,0,  3,6,9,0,0,0,
                *           4,5,7,8,0,0,  7,8,9,0,0,0,  5,6,8,9,0,0     /
ISN   9         DATA FLDN/'DU1','DU2','DU3','DU4','DU5','DU6','DU7','DU8','DU9'/
          C
ISN  10         CALL MCLSCR(NUM,RC)
ISN  11         CALL MLOAD(NUM,RC,'SAMPLE1A')
ISN  12         CALL MPFMAP(NUM,RC,1)
ISN  13         CALL MCLSCR(NUM,RC)
          C
ISN  14    10   MOVES=  0
ISN  15         DUNVRS= EUNVRS
ISN  16    100  CALL MALARM(NUM,RC,0)
ISN  17    110  CALL MCVTBS(MOVES,RC,TCHAR8)
ISN  18         CALL MD2SCR(NUM,RC,'MOVES   ',DMOVE,14,RLEN)
ISN  19         DO 120 I=1,9
ISN  20    120  CALL MD2SCR(NUM,RC,FLDN(I),UNVRS(I),1,RLEN)
ISN  21         CALL MDSPRD(NUM,RC,KEY)
ISN  22         IF (KEY .EQ. 'PF12') GOTO 900
ISN  23         IF (KEY .EQ. 'PF10') GOTO 10
ISN  24         DO 140 I=1,9
ISN  25         IF (KEY .NE. KEYS(I)) GOTO 140
ISN  26         CALL MPCUR(NUM,RC,FLDN(I))
ISN  27         CALL MALARM(NUM,RC,1)
ISN  28         IF (UNVRS(I) .EQ. '.') GOTO 110
ISN  29         MOVES= MOVES+1
ISN  30         DO 130 J=1,6
ISN  31         IF (GALAXY(J,I) .EQ. 0) GOTO 100
ISN  32         TCHAR1= '.'
ISN  33         IF (UNVRS(GALAXY(J,I)) .EQ. '.') TCHAR1= '*'
ISN  35    130  UNVRS(GALAXY(J,I))= TCHAR1
ISN  36    140  CONTINUE
ISN  37         GOTO 100
ISN  38    900  CALL MEXIT(NUM,RC)
ISN  39         STOP
ISN  40         END
```

PASCAL/VS Sample Program

```
PASCAL/VS RELEASE 2.1     SAM1PAS        :                    12/28/84  16:25:38                    PAGE   1

 B P C I  STMT #                              S O U R C E   P R O G R A M                      PAGE XREF

                 V---+----1----+----2----+----3----+----4----+----5----+----6----+----7-V SEQ NO
                 |program sam1pas;                                                      00000001 R *
                 |/*                                                                 */ 00000002
                 |/*       Copyright 1985 - VM Systems Group                         */ 00000003
                 |/*       All rights reserved                                       */ 00000004
                 |/*       Pascal/VS Implementation of "stars" game.                 */ 00000005
                 |/*       See the XMENU Subroutine reference for game rules.        */ 00000006
                 |/*                                                                 */ 00000007
                 | type                                                               00000008 R
                 |   pfkey      = (pf01, pf02, pf03, pf04, pf05, pf06, pf07, pf08, pf09, 00000009 * * * * * * * * * * *
                 |                 pf10, pf11, pf12, enter, clear);                   00000010 * * * * *
                 |   field   = packed array(.1..80.) of char;                         00000011 * R R R P
                 |   char8   = packed array(.1..8.) of char;                          00000012 * R R R P
                 |   char4   = packed array(.1..4.) of char;                          00000013 * R R R P
                 |   char2   = packed array(.1..2.) of char;                          00000014 * R R R P
                 |   coordinate = 1..9;                                              00000015 *
  1              | procedure mload (const number  : integer;                          00000016 R * R * P
  1              |                  var   retcode : integer;                          00000017 R * P
                 |                  const menuname: char8); fortran;                  00000018 R * 1 *
  1              | procedure mexit (const number  : integer;                          00000019 R * R * P
                 |                  var   retcode : integer); fortran;                00000020 R * P *
  1              | procedure mdsprd(const number  : integer;                          00000021 R * R * P
  1              |                  var   retcode : integer;                          00000022 R * P
                 |                  var   key     : char8); fortran;                  00000023 R * 1 *
  1              | procedure md2scr(const number  : integer;                          00000024 R * R * P
  1              |                  var   retcode : integer;                          00000025 R * P
  1              |                  const fldname : char8;                            00000026 R * 1
  1              |                  const buffer  : field;                            00000027 R * 1
  1              |                  const length  : integer;                          00000028 R * P
                 |                  var   rlen    : integer); fortran;                00000029 R * P *
  1              | procedure malarm(const number  : integer;                          00000030 R * R * P
  1              |                  var   retcode : integer;                          00000031 R * P
                 |                  const flag    : integer); fortran;                00000032 R * P *
  1              | procedure mpfmap(const number  : integer;                          00000033 R * R * P
  1              |                  var   retcode : integer;                          00000034 R * P
                 |                  const flag    : integer); fortran;                00000035 R * P *
  1              | procedure mclscr(var   number  : integer;                          00000036 R * R * P
                 |                  var   retcode : integer); fortran;                00000037 R * P *
  1              | procedure mpcur (const number  : integer;                          00000038 R * R * P
  1              |                  var   retcode : integer;                          00000039 R * P
                 |                  const fldname : char8); fortran;                  00000040 R * 1 *
  1              | procedure mcvtbs(const number  : integer;                          00000041 R * R * P
  1              |                  var   retcode : integer;                          00000042 R * P
                 |                  var   string  : char8); fortran;                  00000043 R * 1 *
```

```
PASCAL/VS RELEASE 2.1     SAM1PAS        :                    12/28/84  16:25:38                    PAGE   2

 B P C I  STMT #                              S O U R C E   P R O G R A M                          PAGE XREF

                V---+----1----+----2----+----3----+----4----+----5----+----6----+----7-V SEQ NO
                |                                                                      00000044
                | var                                                                  00000045 R
                |   num,rc,rlen,flag,moves,i       : integer;                          00000046 * * * * * * P
                |   display_moves                  : field;                            00000047 * 1
                |   universe                       : array(. coordinate .) of boolean; 00000048 * R 1 R P
                |   menuname,key,fieldname,tstring : char8;                            00000049 * * * * 1
                |   which_key                      : pfkey;                            00000050 * 1
                | static                                                               00000051 R
                |   keys          : array(. 1..14 .) of char8;                         00000052 * R R 1
                |   fldname       : array(. 1..9 .) of char8;                          00000053 * R R 1
                |   galaxy        : array(. pfkey .) of set of coordinate;             00000054 * R 1 R R R 1
                |   shoot_key     : set of pfkey;                                     00000055 * R R 1
                |   star          : array(. boolean .) of char;                       00000056 * R P R P
                | value                                                                00000057 R
                |   keys(.1.):=  'PF01';    keys(.2.):=  'PF02';    keys(.3.):='PF03'; 00000058 2 2 2
                |   keys(.4.):=  'PF04';    keys(.5.):=  'PF05';    keys(.6.):='PF06'; 00000059 2 2 2
                |   keys(.7.):=  'PF07';    keys(.8.):=  'PF08';    keys(.9.):='PF09'; 00000060 2 2 2
                |   keys(.10.):= 'PF10';    keys(.11.):= 'PF11';    keys(.12.):='PF12'; 00000061 2 2 2
                |   keys(.13.):= 'ENTER';   keys(.14.):= 'CLEAR';                     00000062 2 2
                |   fldname(.1.):='DU1';  fldname(.2.):='DU2';  fldname(.3.):='DU3';  00000063 2 2 2
                |   fldname(.4.):='DU4';  fldname(.5.):='DU5';  fldname(.6.):='DU6';  00000064 2 2 2
                |   fldname(.7.):='DU7';  fldname(.8.):='DU8';  fldname(.9.):='DU9';  00000065 2 2 2
                |   galaxy(.pf01.):=(.1,2,4,5.);      galaxy(.pf03.):=(.2,3,5,6.);    00000066 2 1 2 1
                |   galaxy(.pf07.):=(.4,5,7,8.);      galaxy(.pf09.):=(.5,6,8,9.);    00000067 2 1 2 1
                |   galaxy(.pf02.):=(.1,2,3.);        galaxy(.pf04.):=(.1,4,7.);      00000068 2 1 2 1
                |   galaxy(.pf06.):=(.3,6,9.);        galaxy(.pf08.):=(.7,8,9.);      00000069 2 1 2 1
                |   galaxy(.pf05.):=(.2,4,5,6,8.);                                    00000070 2 1
                |   shoot_key:= &notsign.(.pf10, pf11, pf12, enter, clear.);                  00000071 2 1 1 1 1 1
                |   star(.true.):= '*';               star(.false.):= '.';            00000072 2 P 2 P
```

```
PASCAL/VS RELEASE 2.1     SAM1PAS        :                    12/28/84  16:25:38                    PAGE   3

 B P C I  STMT #                              S O U R C E   P R O G R A M                          PAGE XREF

                  V---+----1----+----2----+----3----+----4----+----5----+----6----+----7-V SEQ NO
                      |                                                                  00000073
   1                  | procedure action_restart;                                       00000074 R *
   1                  | var                                                             00000075 R
   1                  |  i: integer;                                                    00000076 * P
   1                  | begin                                                           00000077 R
   1           1      |   moves:= 0;                                                    00000078 2
   1           2      |   for i:= 1 to 9 do universe(.i.):= false;                      00000079 R 3 R R 2 3 P
                      |   universe(.5.):= true  end;                                   00000080 2 P R
                      |                                                                 00000081
   1                  | procedure action_display;                                       00000082 R *
   1                  | var                                                             00000083 R
   1                  |  i: integer;                                                    00000084 * P
   1                  | begin                                                           00000085 R
   1           1      |   mcvtbs(moves,rc,tstring);                                     00000086 1 2 2 2
   1           2      |   for i:= 1 to 6  do display_moves(.i.):= ' ';                  00000087 R 3 R R 2 3
   1           4      |   for i:= 7 to 14 do display_moves(.i.):= tstring(.i-6.);       00000088 R 3 R R 2 3 2 3
   1           6      |   md2scr(num,rc,'MOVES',display_moves,14,rlen);                 00000089 1 2 2 2 2
   1   1       7      |   for i:= 1 to 9 do begin                                       00000090 R 3 R R R
 1 1   1       8      |     display_moves(.1.):= star(.universe(.i.).);                 00000091 2 2 2 3
   1           9      |     md2scr(num,rc,fldname(.i.),display_moves,1,rlen)  end;      00000092 1 2 2 2 3 2 2 R
   1          10      |   mdsprd(num,rc,key);                                           00000093 1 2 2 2
   1          11      |   malarm(num,rc,0);                                             00000094 1 2 2
   1          12      |   which_key:= clear;                                            00000095 2 1
                      |   for i:= 1 to 14 do if key=keys(.i.) then which_key:=pfkey(i-1)  end; 00000096 R 3 R R R 2 2 3 R 2 1
                      |                                                                 00000097
   1                  | procedure shoot_star;                                           00000098 R *
   1                  | var                                                             00000099 R
   1                  |  i: integer;                                                    00000100 * P
   1                  | begin                                                           00000101 R
   1           1      |   moves:= moves+1;                                              00000102 2 2
   1           2      |   mpcur(num,rc,fldname(.ord(which_key)+1.));                    00000103 1 2 2 2 P 2
   1 1         3      |   if universe(.ord(which_key)+1.)                               00000104 R 2 P 2
   1 1 1       4      |     then for i:= 1 to 9 do                                      00000105 R R 3 R R
   1 2 1       5      |       if coordinate(i) in galaxy(.which_key.)                   00000106 R 1 3 R 2 2
   1 2 1       6      |         then universe(.i.):= &notsign.universe(.i.)                    00000107 R 2 3 2 3
   1 2 1              |         else                                                    00000108 R
                      |     else malarm(num,rc,1)  end;                                 00000109 R 1 2 2 R
```

```
PASCAL/VS RELEASE 2.1     SAM1PAS        :                    12/28/84  16:25:38                    PAGE    4

 B P C I  STMT #                              S O U R C E   P R O G R A M                    PAGE XREF

                    V---+----1----+----2----+----3----+----4----+----5----+----6----+----7-V SEQ NO
                        |                                                                  00000110
                        | begin                                                            00000111 R
                      1 |   menuname := 'SAMPLE1A';                                        00000112 2
                      2 |   mclscr(num,rc);                                                00000113 1 2 2
                      3 |   mload(num,rc,menuname);                                        00000114 1 2 2 2
                      4 |   mpfmap(num,rc,1);                                              00000115 1 2 2
                      5 |   action_restart;                                                00000116 3
                      6 |   action_display;                                                00000117 3
          1           7 |   while which_key &notsign.= pf12 do begin                              00000118 R 2 1 R R
  1   1 1             8 |     if which_key in shoot_key                                    00000119 R 2 R 2
  1   1 1               |       then shoot_star                                            00000120 R 3
  1   2 1             9 |       else if which_key = pf10                                   00000121 R R 2 1
  1     1            11 |         then action_restart;                                     00000122 R 3
                     12 |     action_display  end;                                         00000123 3 R
                        |   mexit(num,rc)  end.                                            00000124 1 2 2 R
```

PL/I Sample Program

```
PL/I OPTIMIZING COMPILER         SAM1PLI: PROC OPTIONS(MAIN);

                  SOURCE LISTING

  NUMBER LEV NT

      10       0  |SAM1PLI: PROC OPTIONS(MAIN);                                          |
                  |/*                                                                    */|
                  |/*       COPYRIGHT 1985 VM SYSTEMS GROUP                             */|
                  |/*       ALL RIGHTS RESERVED                                         */|
                  |/*       PL/I IMPLEMENTATION OF "STARS" GAME.                        */|
                  |/*       SEE THE XMENU SUBROUTINE REFERENCE FOR GAME RULES.          */|
                  |/*                                                                    */|
      80   1  0  |DECLARE                                                               |
                  |  ((MLOAD, MDSPRD, MPCUR)                                             |
                  |           ENTRY (BINARY FIXED(31), BINARY FIXED(31), CHAR(8)),       |
                  |    MD2SCR ENTRY (BINARY FIXED(31), BINARY FIXED(31), CHAR(8),        |
                  |                  CHAR(*), BINARY FIXED(31), BINARY(31)),             |
                  |   (MPFMAP, MALARM)                                                   |
                  |           ENTRY (BINARY FIXED(31), BINARY FIXED(31),                 |
                  |                  BINARY FIXED(31)),                                  |
                  |   (MCLSCR, MEXIT)                                                    |
                  |           ENTRY (BINARY FIXED(31), BINARY FIXED(31)))                |
                  |      EXTERNAL OPTIONS(ASM INTER),                                    |
                  |  (GALAXIES(9)  INIT('001001111'B, '000111111'B, '100100111'B,       |
                  |                     '011011011'B, '101000101'B, '110110110'B,       |
                  |                     '111001001'B, '111111000'B, '111100100'B),      |
                  |   UNIVERSE     INIT('000010000'B)) BIT(9) ALIGNED,                   |
                  |  (DISPLAY_UNIVERSE(9), BLACK_HOLE_OR_STAR(0:1) INIT('.','*')) CHAR(1),|
                  |  (FIELDNAME(9) INIT('DU1',  'DU2',  'DU3',  'DU4',  'DU5',  'DU6',  |
                  |                     'DU7',  'DU8',  'DU9'),                         |
                  |   KEY_TABLE(9) INIT('PF01', 'PF02', 'PF03', 'PF04', 'PF05', 'PF06', |
                  |                     'PF07', 'PF08', 'PF09'),                        |
                  |   KEY)         CHAR(8),                                              |
                  |  (ACTION_MENU, RC, I, IKEY, RLEN, MOVES INIT(0))  FIXED BINARY(31);  |
                  |                                                                      |
     310  1  0  |DISPLAY_ACTION: PROC RETURNS(BIT(1));                                  |
     320  2  0  |  DO I= 1 TO 9;                                                       |
     330  2  1  |    DISPLAY_UNIVERSE(I)= BLACK_HOLE_OR_STAR(SUBSTR(UNIVERSE,I,1));    |
     340  2  1  |    CALL MD2SCR(ACTION_MENU, RC, FIELDNAME(I), DISPLAY_UNIVERSE(I),   |
     350  2  1  |                1, RLEN);  END;                                        |
     360  2  0  |  CALL MD2SCR(ACTION_MENU, RC, 'MOVES', MOVES, 14, RLEN);             |
     370  2  0  |  CALL MDSPRD(ACTION_MENU, RC, KEY);                                  |
     380  2  0  |  CALL MALARM(ACTION_MENU, RC, 0);                                    |
     390  2  0  |  RETURN(KEY &notsign.= 'PF12');  END; /* DISPLAY_ACTION */           |
                  |                                                                      |
     410  1  0  |CALL MCLSCR(ACTION_MENU,RC);                                          |
     420  1  0  |CALL MLOAD(ACTION_MENU,RC,'SAMPLE1A');                                |
     430  1  0  |CALL MPFMAP(ACTION_MENU,RC,1);                                        |
     440  1  0  |DO WHILE(DISPLAY_ACTION);                                             |
     450  1  1  |  IF SUBSTR(KEY,1,3) = 'PF0'                                          |
                  |    THEN DO;                                                          |
```

```
PL/I OPTIMIZING COMPILER          SAM1PLI: PROC OPTIONS(MAIN);

  NUMBER LEV NT

     470   1  2 |       IKEY= SUBSTR(KEY,4,1);                                          |
     480   1  2 |       CALL MPCUR(ACTION_MENU, RC, FIELDNAME(IKEY));                   |
     490   1  2 |       IF SUBSTR(UNIVERSE, IKEY, 1)                                    |
               |         THEN DO;                                                      |
     510   1  3 |           UNIVERSE= BOOL(UNIVERSE, GALAXIES(IKEY), '1001'B);          |
     520   1  3 |           MOVES= MOVES+1;  END;                                       |
     530   1  2 |         ELSE CALL MALARM(ACTION_MENU, RC, 1);                         |
     540   1  2 |       END;                                                            |
     550   1  1 |     ELSE IF KEY = 'PF10'                                              |
               |       THEN DO;                                                        |
     570   1  2 |         UNIVERSE= '000010000'B;                                       |
     580   1  2 |         MOVES= 0;  END;  END;                                         |
     590   1  0 |CALL MEXIT(ACTION_MENU,RC);  END; /* SAM1PLI */                      |
```

Index

Printed in U.S.A.