HONEYWELL

CP-6 HOST MONITOR SERVICES REFERENCE MANUAL VOLUME 1 (DESCRIPTIONS)

SOFTWARE

CP-6

HOST MONITOR SERVICES REFERENCE MANUAL VOLUME 1 (DESCRIPTIONS)

SUBJECT

Description of Monitor Services Available for the Host Processor

SOFTWARE SUPPORTED

Operating System C00.

ORDER NUMBER

CE74-00

March 1985

Honeywell

Preface

This manual explains the monitor services available for use on the host in Honeywell's Control Program Six (CP-6) system. This manual consists of Volume 1 (CE74) which provides service descriptions and Volume 2 (CE75) which provides service structures and related information. Each volume must be ordered individually.

The host monitor services are used both by application level languages (e.g., COBOL, FORTRAN) and by system level languages (e.g., PL-6, BMAP). However, they are invoked in a different manner in the two cases. For an application level language, the compiler provides an interface to each monitor service needed; the user of the language does not call the service directly. For the system level languages, the programs contain explicit calls to each monitor service needed. Thus, this manual is essential for system programmers and serves as a general reference for users who need to understand CP-6 operating system capabilities on this level of detail.

The Los Angeles Development Center (L.A.D.C.) of Honeywell Information Systems Inc. has developed Computer Aided Publications (CAP). CAP is an advanced document processing system providing automatic table of contents, automatic indexing, format control, integrated text and graphics, and other features. This manual is a product of CP-6 CAP.

Readers of this document may report errors or suggest changes through a STAR on the CP-6 STARLOG system. Prompt response is made to any STAR against a CP-6 manual, and changes will be incorporated into subsequent releases and/or revisions of the manuals.

The information in this publication is believed to be accurate in all respects. Honeywell Information Systems cannot assume responsibility for any consequences resulting from unauthorized use thereof. The information contained herein is subject to change. New editions of this publication may be issued to incorporate such changes.

Table of Contents

Tables:

Figures:

About This Manual

This manual consists of Volume 1 (CE74) and Volume 2 (CE75). Volume 1 describes the functions, parameters, and results of all host monitor services that users may find helpful. Volume 1 may be sufficient for programmers debugging by means of symbolic representations only. In most cases, Volume 2 consisting of the appendixes is also essential; Volume 2 includes monitor services structures, error codes and messages, and tabular information. Volume 1 (CE74) and Volume 2 (CE75) must be ordered individually.

To prepare to use the monitor services, the user should first study the following manuals:

CP-6 Concepts and Facilities Manual (CE26)

CP-6 Programmer Reference Manual (CE40)

CP-6 PL-6 Reference Manual (CE44)

Also of interest are the following:

CP-6 System Programmer Guide (CE62)

CP-6 FEP Monitor Services Reference (CE66)

The host monitor services are presented in functional groupings. The most commonly used monitor services are presented in the initial sections. The nine sections in Volume 1 of this manual provide the following information:

Section 1 describes the use of macro library for monitor services and describes the syntax of the monitor service calls.

Section 2 shows a sample program illustrating the use of monitor services.

Sections 3-4 contain the most commonly used monitor services that perform file management, device management, and program management.

Sections 5-7 contain less commonly used monitor services including terminal services, exceptional condition services, and resource control services.

Sections 8-9 include monitor services that require special privilege and in-depth knowledge of the operating system. Some of these services are included for completeness but are reserved for use by Honeywell system programmers.

Section 1

Using Monitor Services

Introduction

This section explains how to call host monitor services and how control is returned to the user. This section also explains how to gain access to the system macro library and other data provided to facilitate use of the monitor services.

The user calls a monitor service by coding the PL-6 CALL statement naming a monitor service and no more than one argument. The argument is the name of a Function Parameter Table (FPT) which contains parameters specific to each monitor service. Thus the syntax for monitor service calls is standard, but the FPT supplies a unique set of parameters to each monitor service.

Note: Typically users of higher-level languages do not call monitor services directly; when this is necessary, refer to the appropriate language guide for information on the interface with PL-6 procedures. Users of BMAP assembly language may refer to the System Programmer Guide (CE62) for information on Inter-language Calling.

Before calling monitor services for execution, the programmer gains access to the macro library and generates FPTs and other structures. In other words, the programmer is required to code:

1. PL-6 %INCLUDE directives to gain access to the CP-6 system macro library and related data, including monitor service entry declarations. (See "INCLUDE Files" in this section.)

2. PL-6 % directives to invoke macros that generate function parameter tables (FPTs) and other structures. (See "System Macro Library" in this section.)

3. PL-6 CALL statements to invoke monitor service subroutines, as discussed later.

What are Monitor Services?

The monitor services consist of subroutines which the user program or user processor may call. PL-6 users must rely on monitor services for functions which the PL-6 compiler does not provide.

Monitor services consist of code that occupies memory outside the user working space. To transfer control to a monitor service at a CALL statement, the PL-6 compiler generates a hardware instruction: the privileged master mode entry (PMME) form of the CLIMB. This form of the CLIMB instruction causes a change of domain, giving control to the monitor.

The domain change, a feature of the system hardware architecture, protects the monitor from all users; it protects each user from all other users; and it protects the current user from the monitor. The monitor returns control to the user following completion of the monitor service. The transfer of control and domain change at a monitor service call is depicted in a simplistic way in Figure 1-1.

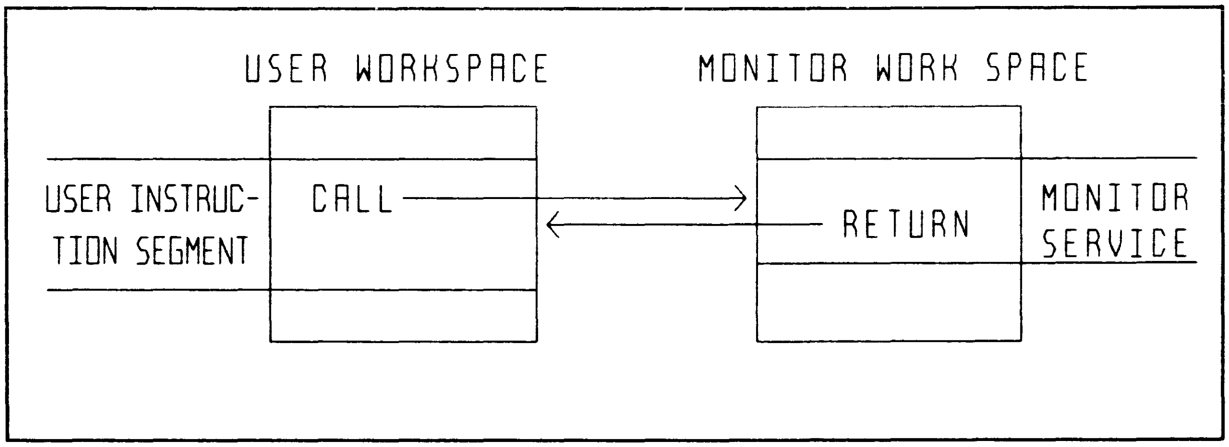

Figure 1-1. Transfer of Control to a Monitor Service

Monitor Service HELP Facility

The Monitor Service HELP facility provides on-line documentation of the monitor services and structures. The HOSTMON HELP facility can be invoked via IBEX and most other system-supplied processors. For each service or structure, the following information is available: service or structure description (and CALL form, for services), parameter or field descriptions. The following examples demonstrate use of the HELP facility.

For the description of the M$OPEN monitor service, enter:
    HELP (HOSTMON) M$OPEN
For the list of parameters for M$OPEN, enter:
    HELP (HOSTMON) M$OPEN TOPICS
For information on a specific parameter of M$OPEN (e.g.,ORG), enter:
    HELP (HOSTMON) M$OPEN ORG
For all available topics pertaining to monitor services, enter:
    HELP (HOSTMON) TOPICS

Within a HELP message, type ? for the next level of information, or ?? for all succeeding levels of information. Additional information on the HELP command is provided below.

Format:

```
HELP [(fid)] [TOPICS] [keyword1 [-] [keyword2]
```

Note: The following elements can be specified in any order:

```
(fid)
TOPICS
[keyword1] [-] [keyword2]
```

For example, HELP (fid) keyword1 - keyword2 TOPICS is acceptable.

Parameters:

(fid) specifies the processor name (for example, HOSTMON). If (fid) is omitted, the current processor is assumed.

TOPICS requests a list of topic or subtopic names, rather than an information message.

Form	Result
HELP (fid) TOPICS	Lists all topics
HELP (fid) TOPICS keyword1 - keyword2	Lists all topics in the range specified by keyword1 - keyword 2
HELP (fid) TOPICS keyword1?	Lists all topics beginning with the prefix specified by keyword1
HELP (fid) TOPICS keyword1	Lists all subtopics for the topic specified by keyword1

keyword1 [- [keyword2]] specifies a topic, a range of topics, or a topic and subtopic to identify what HELP information is requested.

Form	Result
HELP (fid) keyword1	Displays the first level information message for the topic keyword1
HELP (fid) keyword1 keyword2	Displays the information message for keyword1, but only the level identified by the subtopic keyword2
HELP (fid) TOPICS keyword1 [-] [keyword2]	Lists topic or subtopic names. See TOPICS parameter.

keyword1 may include the wildcard (?) character as the rightmost character, if TOPICS is specified.

Description:

HELP displays information.

HELP messages have levels. Once the initial level has been displayed, entering a question mark displays the next level, usually containing greater detail. Entering two question marks, displays the entire message.

Service Calls

The CALL statement naming a monitor service — with the Function Parameter Table (FPT) as the single argument — provides a simple interface between the monitor and the user. The syntax is:

CALL M$name [(fptname)] [ALTRET (label)];

Parameters:

M$name is the reference of the entry to a monitor service subroutine. All monitor service names begin with M$ to distinguish them from other subroutine names.

fptname is the name of the Function Parameter Table (FPT) for the monitor service. The default name of an FPT consists of FPT_ followed by the last portion of the monitor service name. The user typically specifies a different unique name for the FPT as explained later.

ALTRET (label)   specifies the alternate return to be taken if the monitor service cannot be completed successfully. As discussed more fully later in this section, the label must refer to a statement contained in the same procedure as the call to the monitor service.

Example:

```
CALL M$OPEN (FPT_OPEN) ALTRET (L1);
```

In a few cases the monitor service request does not include an FPT name or never results in an alternate return. In those cases the required CALL syntax is explained in the description of the specific monitor service.

System Macro Library

The system macro library is a file called CP_6 in the :LIBRARY account. It contains macros to generate FPTs, Data Control Block (DCBs), and other structures. In addition, the file contains declarations for monitor service entry points.

Invoking Function Parameter Table Macros

The FPT supplies parameters to the monitor service; in many cases the monitor service returns results values to the user via the FPT. The PL-6 macro (%) invocation generates an FPT. The syntax is:

```
%FPT_name [(refid=string[,refid=string]...)];
```

Parameters:

refid=string   can be the following:

FPTN=fptname   specifies a user-defined name for the FPT. The default is FPT_ followed by the last portion of the monitor service name.

STCLASS=storage class   specifies the storage class attribute. For an FPT structure containing data, the storage class may be STATIC, CONSTANT, CONSTANT SYMDEF, STATIC SYMDEF, EXT, or EXTROOT. The storage class may be BASED, AUTO, or SYMREF for a non-data generating FPT structure. The default is STATIC.

keyword=xxx[,keyword=xxx]...   supply initialization parameters. Each keyword represents a field in the FPT. (Initialization parameters can be specified on any FPT macro invocation, but cause values to be stored only for the data-generating form of the macro.)

Invoking Data Control Block (DCB) Macros

The DCB is an essential parameter in the FPT for many monitor services. For this reason a macro is provided to generate DCBs. However, it should be used rarely, as the linker provides DCBs as needed. For further description of DCB content, see Section 3.

The system macro library contains a data-generating macro and a non-data generating macro for DCBs. The invocation of those macros is shown below.

Invocation of %M$DCB generates a DCB which occupies memory in the user work space. The format is:

```
%M$DCB (refid=string[,refid=string]...);
```

Parameters:

refid=string   can be the following:

DCBN=dcbname   specifies the name of the DCB and must be specified. There is no default. See Section 3 for a discussion of DCB naming conventions.

keyword=xxx[,keyword=xxx]...   are initialization parameters which specify initial values for the DCB.

Invocation of %F$DCB generates a DCB structure. The format is:

%F$DCB (refid=string[,refid=string]...);

where

refid=string   can be the following:

DCBN=dcbname   the default is F$DCB.

STCLASS=storageclass   specifies the storage class attribute which may be BASED or "BASED(ptr)". The default is BASED.

Invoking Macros for Vector-Located Structures

Certain parameters passed to monitor services (and also results data returned by monitor services) are stored in areas framed by vectors STORED in the Function Parameter Table. As a convenience, the system macro library provides macros to generate these vector-located structures, including vector-located parameters (VLPs), vector-located results (VLRs), and vector-located arrays (VLAs).

These macros generally take the name of the corresponding keyword. For example, the macro to generate an area named by the ACCT=xxx parameter is VLP_ACCT. The user codes ACCT=xxx, where xxx is VLP_ACCT or the user-defined name for the structure.

The macro invocation of a vector-located structure macro is similar to that for an FPT macro. The syntax is:

%macroname [(refid=string[,refid=string]...)];

Parameters:

refid=string   can be the following:

FPTN=name   specifies a user-defined name for the vector-located structure. The default name is the macro name (i.e., VLP_ or VLR_ or VLA_ followed by a name which is usually a keyword for an FPT macro).

STCLASS=storageclass   specifies the storage class attribute. For a structure containing data, the storage class may be STATIC, CONSTANT, CONSTANT SYMDEF, STATIC SYMDEF, EXT, or EXTROOT. The storage class may be BASED, AUTO, or SYMREF for a non-data generating structure. The default is STATIC.

keyword=xxx[,keyword=xxx]...   supply initialization parameters. Each keyword represents a field in the structure. (Initialization parameters can be specified on a VLP macro invocation, but cause values to be stored only for the data-generating form of the macro.) Keywords are unnecessary on the invocation of VLR or VLA macro.

Later sections of this manual discuss vector-located structures with the monitor services to which they apply. Fields within a vector-located structure are described using the same conventions as for FPT parameter descriptions.

Parameter Conventions

Monitor service parameters are shown in the format keyword=xxx throughout this manual. For each monitor service, keywords are listed in alphabetical order, unless there is a special reason to present them in a functional sequence.

NOTE: FPTN and STCLASS are not repeated for each monitor service, but these keywords may be used.

The user can supply monitor service parameters via initialization parameters (as already discussed) and by two additional methods. The following list describes, in order of precedence, the source of parameters passed to a monitor service in the FPT:

1. Default parameters from FPT or VLP macro definitions.

2. Initialization parameters supplied by the user via FPT or VLP macro invocation.

3. Run-time modification performed between monitor service calls.

Figure 1-2 illustrates cases 1 and 2; case 3 is described in a later subsection.

Initialization Parameters

Initialization parameters specified in the format keyword=xxx at macro invocation are represented throughout this manual as shown in Table 1-1.

Default Parameters

When the user invokes an FPT macro but omits a parameter, the default is used. The default parameters passed to monitor services are listed along with the structure summaries in Volume 2, CE75. Defaults are explained briefly in Table 1-1.

```
FPT macro:  The FPT_DELREC macro used in conjunction with the
M$DELREC monitor service provides these keywords:

    FPTN
    STCLASS
    KEY
    LKEY
    DCB

Sample FPT macro invocation:

  %FPT_DELREC (DCB=M$MASTER,KEY=KEY_BUF);

  As a result the structure called FPT_DELREC (by default) generated
  in STATIC storage (by default) contains:

  o   The DCB name M$MASTER to identify the file from which the
      record deletion is to occur.

  o   A vector framing KEY_BUF where the user supplies the key of
      the record to be deleted.

  o   A NIL vector for LKEY (by default) which means that only the
      record identified via KEY (and not a range of records) is to
      be deleted.

FPT format in abbreviated form:

  DCL 1 FPTN STCLASS,
        2 V_ ...,
        2 KEY_ ...,
        2 LKEY_ ...,
        2 V ...,
         3 DCB# ...,

  NOTE:  The vector V_ frames the values portion of the FPT
  which is called FPT_DELREC.V.
```

Figure 1-2. FPT Macro Invocation and FPT Format

Table 1-1. Parameter Conventions and Defaults

Keywords=	Meaning
VARIABLE	VARIABLE is always framed by a vector in the FPT. A variable can be created by invoking a macro in the system macro library or can be declared by the user.

Table 1-1. Parameter Conventions and Defaults (cont.)

Keywords=	Meaning
	VARIABLE may also specify the word NIL or ERASE. NIL implies that no value is being specified, nothing is to be done, and any previously specified value to remain unchanged. ERASE implies that any current or previous value is to be erased, and generally implies that the default is to be reinstated. The default for VARIABLE is NIL, unless otherwise stated.
ENTRY	ENTRY is a procedure name, usually designated as an ASYNC procedure, to which control may be transferred. ENTRY is always defined as a PL-6 entry point. The default for ENTRY is NIL, unless otherwise stated.
DCBNAME	DCBNAME is the name of a PL-6 variable with the storage class attribute DCB. It can be created by invoking the M$DCB macro or by declaring the DCB storage class attribute. When the DCB parameter is listed for a service, DCBNAME is a required parameter unless otherwise stated.
OPTION	OPTION represents one of several possible or parameter assignments. (In some cases the choices {xxx|xxx} are listed in braces separated by the | symbol.)
{YES|NO}	{YES|NO} specifies one of two conditions possible for a specific parameter. The default is usually NO. In some cases, the default causes the parameter to be ignored.
VALUE-DEC(range)	VALUE-DEC(range) assigns a decimal value within the specified range. Zero is often defined as the default.
VALUE-CHAR(n)	VALUE-CHAR(n) assigns a character value of the length n characters. Blank is often defined as the default.

Table 1-1. Parameter Conventions and Defaults (cont.)

Keywords=	Meaning
VALUE-BIT(n)	VALUE-BIT(n) assigns an octal value which is n bits in length. Zero is often defined as the default.

Run-Time Modification of Parameters

Modification of parameters between monitor service calls requires an understanding of the format of parameters as they appear at run-time, (i.e., parameters passed to the monitor in or via the FPT).

o Vectors: VARIABLEs are framed by vectors stored in the FPT.

o Values: Other parameters are stored as values in the FPT or VLP, including VALUEs, ENTRYs, OPTIONs, and parameters represented as {xxx|xxx}.

Rules for modifying FPT and VLP fields at run time are listed below:

1. Use the fully qualified name of an FPT or VLP field. Refer to the structure summaries in Volume 2, CE75 to determine the exact field name.

   FPTs are structured according to several conventions listed below that ensure conformity in FPT field names.

   o Names for vectors appear at the second level of the structure and field names end with the underscore character. Thus, the fully qualified name for a vector is fptname.keyword_ (for example, FPT_DELREC.KEY_).

   o Values generally appear at the third level of structure (within the structure called V) and each field name contains the # symbol as the last character. Thus, the fully qualified name of a value at the third level of the FPT structure is fptname.V.keyword# (for example, FPT_DELREC.V.DCB#).

     Figure 1-2 illustrates the format of the FPT mentioned in the preceding example.

     Vector-located structures conform to these conventions:

   o Parameters generally occupy the second level of the structure and the field names end in the # symbol. Thus, the fully qualified name for a value at the second level of a VLP structure is vlpname.keyword# (for example, VLP_TAB.MARGIN#).

2. Use subfields defined for vectors in the FPT. For write operations, for example, the vector BUF_ includes the fields BOUND and BUF$. BOUND contains the buffer size minus 1 (in bytes); BUF$ is a pointer to the start of the buffer. To change the size of a buffer, include the buffer field name and the subfield BOUND.

   Example:

   ```
   FPT_WRITE.BUF_.BOUND = BUFSIZE-1;
   ```

3. For parameters represented as OPTION or {xxx|xxx}, use a special form of the expression including the % symbol before and the # symbol after the expression. The file CP_6_SUBS supplies these expressions via the PL-6 %EQU facility.

   Example:

```
        FPT_OPEN.V.EXIST# = %OLDFILE#;
```

4. For services (such as M$OPEN and M$STRMCTL) that may cause modification to pre-existing values, the defaults often cause parameters to be ignored. EQUs for these defaults, available in CP_6_SUBS.:LIBRARY, are as follows:

```
        IGNORE_YES_NO#='01'B
        IGNORE_BINVAL#=0
        IGNORE_CHARVAL#=BINASC(0)
        IGNORE_PTRVAL#=CHARTEXT('ADDR(NIL)')
        IGNORE_VARIABLE#=CHARTEXT('VECTOR(NIL)')
        IGNORE_VLP_BYTE#=255
```

Include Files in :LIBRARY

The files related to the use of monitor services reside in the :LIBRARY account. The user codes the PL-6 %INCLUDE directives to gain access to the contents of those files. Most users need only the files shown in Table 1-2. For a brief description of all files in the :LIBRARY account refer to Volume 2, CE75.

Table 1-2. INCLUDE Files Related to Monitor Services

File Name	Description
CP_6	System macro library containing macros for: Data Control Blocks (DCBs) Function Parameter Tables (FPTs) Vector-Located Parameters (VLPs) Vector-Located Results (VLRs) Vector-Located Arrays (VLAs) Task Control Block structures (see Section 6) and declarations for monitor service entry points.
CP_6_SUBS	PL-6 %EQUs to define substitution strings used as options to initialize FPTs and DCBs.
B_ERRORS_C	PL-6 %EQUs to correlate monitor error codes and error names.
B$JIT	Macros to generate the Job Information Table (JIT) structure. NOTE: As a site option, the system manager may create a system file containing these and other files in a partially compiled state to speed compilations. The items in the system file are included at any PL-6 compilation (unless the NSYS option is specified); any INCLUDE statements for the files listed above are ignored.

Naming Conventions

Table 1-3 summarizes the naming conventions associated with monitor services.

Table 1-3. Naming Conventions

Category	Sample	Convention
Monitor service	M$OPEN	All monitor service names begin with M$ followed by a descriptive name.
FPT macros	FPT_OPEN	ALL FPT macro names begin with FPT_ followed by a name which is usually the last portion of the monitor service name. The FPT macro name is used as the default name for the FPT structure, if the user does not include FPTN= to provide another name.
FPT fields	FPT_OPEN.V.ORG FPT_OPEN.NAME_	The fully qualified name of an FPT field consists of the FPT name, V which is the name associated with all parameters stored within the FPT itself (if applicable), and the keyword with # or _ as the last character. (# denotes a parameter actually stored within the FPT; _ denotes a vector-located parameter or results area.) For certain FPTs an additional level of structure is included. (See Volume 2, CE75 for a summary of structure and field names.)
VLP macros	VLP_NAME	All vector-located parameter names begin with VLP_ followed by the associated keyword. The VLP macro name is used as the default name for the VLP structure; alternatively the user can override the default name by use of the FPTN keyword.
VLR macros	VLR_DISPLAY	Most VLRs conform to the conventions stated for VLPs.
VLA macros	VLA_JOBSTATS_	VLA names begin with VLA_CHECK# followed by the name associated VLP name.
DCB	M$LM M$LO F$101 MYDCB	Reserved names for Data Control Blocks (DCBs) are listed in Section 3. These names begin with M$ or F$, however, the user may define a DCB name which does not start with M$ or F$.

Normal Return on Monitor Services

Following a monitor service, control typically returns to the statement following the CALL statement that invokes the service. The FPT contains resultant values if a results parameter is defined for the service.

Error Return on Monitor Services

The monitor can detect errors specific to a monitor service request. For example the monitor may detect an error in a user-supplied parameter; or the monitor may encounter another condition that prevents completion of the service. The monitor service takes an error return if any such error occurs.

If the user makes no provision for the error return from a monitor service, the monitor action depends on the severity of the error. If the severity code for the error is greater than 0, the monitor causes the program to be aborted. If the severity code is 0, the monitor returns control to the point immediately following the original monitor service call statement.

The error return transfers control to a user-specified location if the user requests one of these options:

- o The ALTRET option on the call to the monitor service. This option permits the user to specify the label of a routine to receive control in the case of an error on a specific monitor service request.

- o An option of the M$TRAP monitor service. This option specifies a common routine to process errors on monitor service call statements that do not include the ALTRET option.

- o The M$XCON service allows specification of a routine to be entered when a program is about to be aborted. This includes an error on a monitor service call when neither ALTRET nor the M$TRAP option is specified.

The full error description is stored in the user's Task Control Block (TCB) when the user's program receives control to process the error. Section 6 describes the TCB, provides detailed information on the various options for processing monitor service errors, and discusses the based structures provided in the file CP_6 to access the full error description.

The standard format for monitor service error information is illustrated below:

Content	Field Name	Format
Functional Code Group	xx.ERR.FCG	VALUE-BIT(12)
Module Identifier	xx.ERR.MID	VALUE-BIT(6)
Monitor Flag	xx.ERR.MON	VALUE-BIT(1)
Error Number	xx.ERR.ERR#	VALUE-DEC(0-16383)
Severity Level	xx.ERR.SEV	VALUE-DEC(0-7)

The error code can be accessed via several based structures which can be invoked from the CP_6 file.

The last three fields (MON, ERR#, and SEV) are of primary importance to the user. A user routine may first analyze the SEV field to determine if it can process the error. The ERR# field gives precise identification of the error condition. These codes are listed in Volume 2, CE75 with an explanation of the condition detected. The field MON, set at any monitor service error, can be tested if the user processes errors on monitor services as well as other errors which use the same format for error description. (For example, an alternate shared library could use this format for error reporting to the user.)

The file called :ERRMSG in the system account contains an error message for each monitor error condition. The M$ERRMSG service is available to report errors using this file (or any other file).

Section 2

Sample Program

The sample program illustrated in this section is written in the PL-6 language. Users wishing to call monitor services from another language may refer to the discussion of the interface to PL-6 procedures in the appropriate language guide, or for BMAP users to the System Programmer Guide (CE62) Inter-language Calling.

The sample program performs a straightforward function which is described in the commentary. The commentary also refers to pertinent sections of this manual.

```
        /*M*     LINKFID   ACCEPT A TEXTFID AND M$LINK TO THE
            RU SPEC'D*/
                           /*The sample program LINKFID calls
                             these monitor services:  M$FID,
                             M$WRITE, M$YC, M$LINK, and M$ERRMSG.
                             The INCLUDE statements (for the
                             system macro library and %EQUs and
                             invocation of macros for Function
                             Parameter Tables (FPTs) and
                             Vector-Located Parameter structures
                             precede the calls for these
                             services. */
        LINKFID: PROC(TEXTFID,TEXTCMD,SIZCMD);
                           /*LINKFID can be used to link (via
                             M$LINK) to a run unit or to call IBEX
                             (via M$YC) depending on what is
                             specified.  If the specified run unit
                             is IBEX.:SYS, M$YC is done, otherwise
                             M$LINK is done.
                             The first argument is a text string
                             specifying the fid of the desired run
                             unit.  The string must be less than
                             128 bytes and be terminated with a
                             blank.  If a command is to be passed,
                             it may be specified.  If not
                             specified, the file name of the
                             target run unit is passed as the
                             command.
                             The command string is the second
                             command.  The command size may also
                             be specified.  if not specified, it
                             is assumed to be 80 bytes.  If
                             specified, it is the third argument.*/

    1   DCL TEXTFID CHAR(FIDSIZ);
    1   DCL FIDSIZ UBIN;
    1   DCL TEXTCMD CHAR(CMDSIZ);
    1   DCL CMDSIZ UBIN;
    1   DCL SIZCMD UBIN;
                           /*These INCLUDE files are discussed in
                             Section 1. */
        %INCLUDE CP_6;
        %INCLUDE CP_6_SUBS;
```

```
1    DCL B$TCB$ PTR SYMREF;
                         /*Structures to get ERRCODE from TCB
                           as discussed in Section 6 */
     %B$TCB (STCLASS="BASED(B$TCB$)");
     %B$ALT ;
1    DCL M$DO DCB;

                         /*Invoking FPT macros is described in
                           Section 1.  The montior service
                           parameters illustrated here are
                           described in Sections 3 and 4. */
                         /*FPT's for FID, LINK, and YC.  Note
                           that M$FID  returns components of fid
                           to the variables which M$LINK
                           specifies for run unit location. */
     %FPT_FID (FPTN=FID,
             STCLASS=STATIC,
             NAME=RUNM,
             ACCT=RUAC,
             PASS=RUPS,
             SN=RUSN,
             WSN=RUWSN,
             ASN=RUASN,
             RES=RURES);
     %FPT_LINK (FPTN=LINK,
             STCLASS=CONSTANT,
             CMD=ERBUF,
             NAME=RUNM,
             ACCT=RUAC,
             PASS=RUPS,
             PSID=RUSN.SN#);
     %FPT_YC (FPTN=YC,
             STCLASS=CONSTANT,
             CMD=ERBUF);
                         /*Various FPT's to print errors, etc.*/
     %FPT_ERRMSG (FPTN=ERRMSG,
             CODE=ECODE,
             BUF=ERBUF,
             OUTDCB1=M$DO,
             STCLASS=CONSTANT);

     %FPT_WRITE (FPTN=WERRALT,
             STCLASS=CONSTANT,
             BUF=MESERRALT,
             DCB=M$DO);
     %FPT_WRITE (FPTN=WBADFID,
             STCLASS=CONSTANT,
             BUF=MESBADFID,
             DCB=M$DO);
     %FPT_WRITE (FPTN=WBADASN,
             STCLASS=CONSTANT,
             BUF=MESBADASN,
             DCB=M$DO);

1    DCL MESERRALT CHAR(0) STATIC
1        INIT('ALTRET FROM FID OR LINK, THEN FROM
1                    ERRMSG');
1    DCL MESBADFID CHAR(0) STATIC
1        INIT('FID CAN''T BE THAT LONG');
1    DCL MESBADASN CHAR(0) STATIC
1        INIT('YOU CAN ONLY M$LINK TO A FILE');
```

```
1    DCL ECODE BIT(36) STATIC;
1    DCL ERBUF CHAR(140) STATIC;
                         /*Variables specified in the FPTs
                           reference the following VLP
                           structures which are described in
                           Section 3.*/
     %VLP_NAME (FPTN=RUNM,
               LEN=31,
               STCLASS=STATIC);
1    DCL RUNMC REDEF RUNM CHAR(32);
1    DCL RUAC CHAR(8) STATIC;
1    DCL RUPS CHAR(8) STATIC;
     %VLP_SN (FPTN=RUSN,
               STCLASS=STATIC);
     %VLP_WSN (FPTN=RUWSN,
               STCLASS=STATIC);
1    DCL RUASN UBIN BYTE STATIC;
1    DCL RURES CHAR(4) STATIC;


     %EJECT;
1              FIDSIZ=128;
1              CALL INDEX1(FIDSIZ,' ',TEXTFID) ALTRET
1                   (BADFID);
                         /*Isolate the fid for M$FID.*/
1              IF FIDSIZ=0
1              THEN
1                 GOTO NOTFILE;
1              FID.TEXTFID_=VECTOR(TEXTFID);
                         /*The general format of monitor
                           service calls is discussed in
                           Section 1.  The form of the call for
                           each service in this sample program
                           is illustrated in Sections 3 and 4.*/
1              CALL M$FID(FID) ALTRET(PERRMSG);
                         /*Break fid down to its components. */
1              IF NOT RUASN=%FILE#
                         /*Run unit must be a file somewhere. */
2              THEN DO;
2    NOTFILE:     CALL M$WRITE(WBADASN);
2                 RETURN;
2                 END;
1              IF ADDR(SIZCMD)=ADDR(NIL)
                         /*Get command size - use 80 if none. */
1              THEN
1                 CMDSIZ=80;
1              ELSE
1                 CMDSIZ=SIZCMD;
1              IF (RUNM.NAME#='IBEX') AND (RUAC=':SYS')
                         /*If IBEX.:SYS was specified, do M$YC
                           passing text command, however
                           acquired. */
2              THEN DO;
2                 IF ADDR(TEXTCMD)=ADDR(NIL)
2                 THEN
2                    ERBUF=RUNM.NAME#;
2                 ELSE
2                    ERBUF=TEXTCMD;
2                 CALL M$YC(YC) ALTRET(PERRMSG);
2                 END;
2              ELSE DO;
                         /*Otherwise, do M$LINK passing textc
                           command, however acquired. */
2                 IF ADDR(TEXTCMD)=ADDR(NIL)
2                 THEN
2                    ERBUF=RUNMC;
```

```
2              ELSE
2                 CALL CONCAT(ERBUF,BINASC(CMDSIZ),TEXTCMD);
2              CALL M$LINK(LINK)ALTRET(PERRMSG);
2              END;
1           RETURN;
    %EJECT;
1   PERRMSG: ECODE=B$TCB.ALT$->B$ALT.ERR;
1           CALL M$ERRMSG(ERRMSG) ALTRET(CATCH22);
                        /*Output any error message resulting
                          from M$FID, M$LINK, or M$YC. */
1           RETURN;

1   CATCH22: CALL M$WRITE(WERRALT);
1           RETURN;


1   BADFID: CALL M$WRITE(WBADFID);
1           RETURN;

1   END LINKFID;
```

Section 3
File Management Services

INTRODUCTION

All I/O operations are performed by the monitor for the user. The user program never directly accesses an I/O device, but rather requests that the monitor do so. This section describes I/O and the I/O services provided by the monitor and briefly describes related file management concepts. However, for a comprehensive discussion of I/O management, refer to the CP-6 Programmer Reference Manual which presents file formats, file access, file storage and allocation, and pertinent IBEX commands.

To request I/O, the user calls an I/O service specifying a Function Parameter Table(FPT), which in turn refers to a Data Control Block(DCB). The combination of the I/O service call, the FPT, and the DCB provides the information that the monitor needs to perform the requested operation. Generally, the DCB contains the attributes of a file or device. For example, for output to a line printer, the width of a printline is one value in the DCB. The FPT contains information that is specific to the operation to be performed (e.g., the location of the buffer that is to be output to the printer in this specific operation).

Separation of information into the DCB and the FPT allows the user to create one DCB for a destination and reference that DCB throughout his program, whenever I/O for that destination is required.

In addition to serving as a source of information for the monitor in performing an I/O operation, the DCB also provides a place for the monitor to store information while it is performing an I/O operation. Some of the information stored in the DCB by the monitor may be of use to the user, and some has meaning only to the monitor.

The user is responsible for providing the name of a properly initialized DCB with every call to the monitor requesting an I/O operation. The user may obtain a DCB by

1. Explicitly creating his own DCB at program compilation time by invoking the M$DCB macro.

2. Specifying the DCB data type without a structure to cause the linker to create a DCB.

3. Explicitly requesting creation of a DCB at run time by calling the M$GETDCB monitor service.

Monitor calls are provided to permit the user to initialize or alter DCBs.

DCB Manipulation

If the user program is written in COBOL or FORTRAN, or most other higher level languages, the compiler includes in the object unit all necessary I/O service calls and supplies initialized DCBs or DCB references to be satisfied by the linker. However, if the user program is written in PL-6, the user must provide all I/O calls and generate DCBs explicitly:
by invocation of the M$DCB macro, by specifying the DCB storage class attribute, or by calling the M$GETDCB monitor service at run time.

The DCB contents can be set or modified in various ways. The sources of values for the DCB and the order in which they are entered into the DCB are as follows:

- o Default parameters supplied in the M$DCB macro or the default values supplied by the linker or the M$GETDCB service.
- o Parameters supplied at invocation of the M$DCB macro.
- o An assign/merge record is entered into a DCB at program invocation (as a result of a SET command) or at M$GETDCB (if MERGE is specified).
- o Parameters in the FPT for the M$OPEN monitor service.

The DCBs reside in the Read-Only segment which cannot be modified directly by the user program. The monitor services and the techniques mentioned above are the only means for a user to modify a DCB.

When a DCB is opened in the INPUT or UPDATE mode, the attributes of the disk or tape file or device are returned to the user in the DCB by the monitor. The fields in the DCB may be inspected by the user through a based structure of a DCB. The F$DCB macro in the file CP_6 provides such a structure.

The user refers to a DCB by a name for most monitor service calls. The monitor, however, assigns a number to the DCB for internal use. DCBs defined by the user are assigned numbers starting at DCB number 10. All DCBs for a job are included in a DCB pointer table created by the system. (DCBs 1 to 9 are reserved for the monitor.)

A number of DCB names are associated with common I/O functions: M$LO for listing output, M$SI for input, F$101, etc. The system supplies initial values for the DCB, if the user does not supply values. For DCBs with the reserved names listed in Table 3-1, appropriate values are supplied for the function(FUN) and device name(RES) parameters; but only when 1) the DCB is supplied by the linker or M$GETDCB or 2) the DCB is "scrubbed" via M$OPEN. The standard defaults for FUN and RES are used whenever the M$DCB macro is invoked.

There are additional parameters in the DCB which are always set by the monitor. For notational purposes only, they are sorted to follow parameters set by the user and they are flagged for your attention by an embedded period.

The following discussion describes the M$DCB macro and monitor services that permit DCB manipulation: getting and releasing DCBs dynamically (M$GETDCB, M$RELDCB), checking for correspondence of two DCBs to the same file or device (M$CORRES), and supplying a file management account which is the default if the user omits the account on opening the DCB (M$SETFMA).

M$DCB - Data Control Block

The DCB is the communication data block between the user and the monitor concerning the attributes of a disk file, labeled tape file, or device.

ACCT = VALUE-CHAR(8). The account in which the file is cataloged is 8 characters from the ASCII 7-bit set, excluding control characters.

If ACCT is not specified, it is taken from the user's File Management account which defaults to the logon account but can be modified by the M$SETFMA service or !DIRECTORY command.

ACS = OPTION. All ACS options apply to disk files; only the SEQUEN and JRNL options apply to tape files. The ACS parameter defines access characteristics for a file and restricts user access to the file only as specifically stated in the explanation of each option.

SEQUEN	When creating a keyed file the user must supply keys in ascending order. A file created with this option can be read or written with or without a key.
DIRECT	When creating or updating a keyed file, the user may supply keys in any order. A file created with this option can be read or written with or without a key.
JRNL	Specifies journal access mode for a consecutive, named disk or tape file. This mode is designed to permit one user (the journal owner) to control the file while any user may write to the file. A user may become the journal owner by performing the first open of the file if neither the file nor the DCB specifies GHSTACS=JRNLOPN. Otherwise, the first attempt to open the file waits for the system ghost to open it first, thereby becoming its owner. In order to keep this file name continually available for output from users, the journal owner can periodically close and rename the journal file. Thus, other users of the file can continue to send output to the file without being aware that a new physical file was substituted for the file just closed. The file opened in journal mode is subject to these special rules:

1. The first user opening the file must specify FUN=CREATE whether the file exists or not and whether GHSTACS=JRNLOPN or not (otherwise an alternate return results). EXIST=OLDFILE must be used to extend an old file. The owner keeps the journal available by keeping this DCB open (close with rename does not close a journal DCB). (Any attempt to update the file after the owner has closed it results in an alternate return.)

2. Other users who access the file in journal mode must open the file with FUN=UPDATE or CREATE; FUN=IN causes an alternate return. The only meaningful EXIST option is ERROR.

3. For the owner and any user of the file, the following rules apply at M$OPEN:

   - o DISP must be set to NAMED or must not be specified at all.
   - o CTG = YES is assumed.
   - o ORG must be set to CONSEC.
   - o SHARE is ignored.
   - o NXTA, NXTF, THISF, DELETE must not be specified.
   - o For a tape file opened in journal mode, a file of the same name must not concurrently exist on disk. (When a tape journal file is opened in journal mode, a dummy file of the same name is opened on disk.)

DS2 through DS8   Specifies data-segment access for a random file. This mode permits users to share the data of a random file directly in memory. The size of the file(IXTNSIZE) must be between 2 and 257 granules, which requires between 1 and 256 pages of memory to be available for the data-segment. The first granule of the file is used to fill in the holes caused by the 1-word granule headers. M$EXTEND may be used to enlarge the file and data segment simultaneously, or if SIZ=0, to enlarge the data segment to the current size of the file. M$TRUNC may be used to cause the current state of the data segment to be saved in the file. If an attempt is made to create a 1-granule file in this mode, EXIST=ERROR is enforced and the data segment size is obtained from XTNSIZE instead of IXTNSIZE-1. Also in this case, the file cannot be cataloged (although the CTG option is honored and permits the file to be shared, and any access control specifications function normally). In addition, M$EXTEND is still used to enlarge the data segment, but M$TRUNC does nothing.

BLOCK   Is for internal system use only.

The default is ACS=SEQUEN.

ASN = OPTION. FILE, TAPE, DEVICE, or COMGROUP indicates whether the DCB parameters describe a DISK file (FILE), a labeled tape (TAPE), a specific device (DEVICE), or a communication group (COMGROUP).

The default is DEVICE.

AU = {YES|NO}. is meaningful only for a COMGROUP and indicates that the user is to be the administrative user.

BLKL = VALUE-DEC(1-32764). Specifies the maximum physical tape record size (in bytes) that will be read from or written to a labeled tape. For CREATE opens BLKL will be rounded up to a multiple of 4 (ANS formats) or 8 (CP-6 ORGs). For IN or UPDATE opens BLKL is determined from the file opened except for managed tape files for which BLKL must be specified.

If no BLKL (0) is specified, a format dependent default will be supplied.

For complete information on options and defaults for ANS ASCII or IBM EBCDIC tapes, see Host Monitor Services Reference Manual (CE75), Appendix F. That appendix also specifies the maximum size for BLKL that is also format dependent. The default for CP-6 format tapes is 4096.

BLOCKED = {YES|NO}. YES specifies that logical records and record segments are to be packed into physical tape records. NO results in, at most, one record or record segment per tape record. BLOCKED applies only to FIXED and VARIABLE labeled and managed tape files. BLOCKED is determined for IN or UPDATE opens from the labeled tape file opened. Blocked and unblocked files may be freely intermixed on a volume. BLOCKED should be used in conjunction with SPANNED for optimum tape record utilization. (BLOCKED must be specified for INPUT or UPDATE managed tape files.) CP-6 format tape files are always blocked.

The default is YES which results in the most efficient usage of tape records.

BUPM = {YES|NO}. YES specifies that file is to be backed up if modified. The default causes this parameter to be ignored. The default is an attribute of the file's account.

CNVRT = {YES|NO}. YES specifies that for EBCDIC tape files (as determined by tape labels on input and DCB.EBCDIC on output) data is to be translated from EBCDIC to ASCII after reading or ASCII to EBCDIC before writing. Translation is done while data is being moved between user and monitor buffer for all file formats. CNVRT applies only to FIXED and VARIABLE formats. For non-binary writes to free tape if CNVRT is set and EBCDIC is not, the user's buffer is converted to assure that no high order bits are set (which will cause an I/O error). If both CNVRT and EBCDIC are set for device tapes, normal translation takes place.

The default is YES.

COMP = {YES|NO}. YES specifies that the file is to have the attribute "compressed". All records written into a file will be compressed. The records will be restored when read. Compression reduces file size to approximately 60 percent of the space it would occupy if uncompressed and increases user service time required to access the file by approximately two and one half times. COMP is applicable only to CREATE opens of consecutive and keyed disk files. Encryption cannot be used on compressed files.

The default is NO.

CTG = {YES|NO}. is meaningful only for CREATE opens of disk files. This parameter controls cataloging of the file at open time.

YES specifies that the file is to be cataloged in the file directory at open time. If EXIST=NEWFILE, any existing old file of the same name is deleted from the file directory (and is released when any current users close the old file). When CTG=YES and SHARE=ALL or SHARE=IN, the new file may be concurrently opened by other users, as if it were opened UPDATE instead of CREATE.

NO specifies that the file is not to be cataloged at open time. If CTG=NO and a new file is being created (EXIST=NEWFILE, EXIST=ERROR, or EXIST=OLDFILE when the old file does not exist), the newly created file is not available for access by other users until after it is closed for the first time with DISP=SAVE.

The default is CTG = NO.

CVOL = {YES|NO}. YES specifies that the user desires to be notified (via ALTRET on M$READ or M$WRITE) of end-of-volume conditions. For labeled tapes the user may then M$CVOL to cause the next volume to be mounted or perform another I/O request which automatically causes the next volume to be mounted. CVOL is primarily useful when user trailing labels are desired and may cause the ALTRETurning M$WRITE to fail in which case it must be retried.

CVOL should only be specified for SPANNED=NO tape files since spanned tape files may have records which cross volume boundaries. These records are difficult to process if the user takes CVOL control.

For free tapes, end-of-tape will be reported to permit the user to M$CVOL to the next output tape or M$WRITE which automatically mounts the next output tape. For read operations, two file marks are interpreted to indicate end-of-volume. A read encountering two file marks will 1) if CVOL was specified, return an end-of-file error and leave tape position after the first file mark or 2) if CVOL was not specified, cause the next volume to be automatically mounted and read.

The default is NO in which case volume changes are done automatically.

DCBN = DCBNAME. Specifies the DCB name. Any 1-31 character name that conforms to language requirements may be used. Certain DCB names result in defaults being assumed for FUN and RES at M$OPEN.

This parameter is required.

DENSITY = OPTION. Options are D556, D800, D1600, and D6250. The recording density at which a tape is to be written. DENSITY may be specified when creating the first file of a volume set or when the volume's density can not be determined. All following files are created at the same density as the first. For IN or UPDATE opens, DENSITY is determined from the tape volume containing the opened file unless the density could not be determined during AVR due to an I/O error.

The default is determined for the site by an option of the TIGR processor.

DISP = OPTION. Is meaningful for disk files for FUN=CREATE opens only. When a new file is created, it is either allowed to be cataloged as a named file or never allowed to be cataloged and is classified as a scratch file. The options are as follows:

NAMED - Specifies that when the new file is created it has been given a name that will allow it to be cataloged as a named file.

SCRATCH - Specifies that the new file is a scratch file. It will not have a name and can never be cataloged.

The default is DISP=NAMED.

DVFC = VALUE-CHAR(1). Specifies the default format control character for records which have no format control specified on the M$WRITE request. The default is not to use VFC.

EBCDIC = {YES|NO}. YES specifies that tape file labels (including user labels) are to be translated from EBCDIC to ASCII on input and from ASCII to EBCDIC on output. Data is subject to translation (see DCB.CNVRT). For IN or UPDATE, DCB.EBCDIC is determined from the current volume. For CREATE it must be the same as for the current volume unless the current volume is the first volume of the volume set and is positioned to the beginning of the volume. EBCDIC files cannot be created on ASCII volumes and vice versa. EBCDIC applies only to VARIABLE and FIXED files on labeled and managed tape and to free tape. EBCDIC must be specified on output and input for free and managed tape files.

The default is NO.

EVENT = VALUE-DEC(1-?). Specifies an event number to be returned to the user of this ASN=DEVICE DCB when B$COMIO events or FPRG-related events are returned to the user. Zero indicates not to return these events. The default causes this parameter to be ignored.

EXIST = OPTION. Is meaningful at CREATE opens for disk files only. Named disk files (DISP=NAMED) are conditionally created according to the EXIST options listed next, provided there are no sharing conflicts (see SHARE parameter). Scratch files are unconditionally created (thus the EXIST parameter is not meaningful if DISP=SCRATCH).

ERROR - Requests new file creation if a file of this name does not exist. If a file of the same name exists the alternate return is taken.

NEWFILE - Requests new file creation even if a file of the same name exists. The new file replaces any existing old file if CTG=YES on the create open or if the new file is closed with DISP=SAVE.
NOTE: When EXIST=NEWFILE and FUN=CREATE, EXIST operates as if it were OLDFILE if all the following conditions are true:

1. The M$OPEN does not specify NAME.
2. The previous open of the DCB of this name in this session (not necessarily in the job step) specified FUN = CREATE, EXIST = NEWFILE.
3. A SET or RESET command for this DCB has not been issued since the previous open.

See the CTG parameter for a discussion of the types of concurrent use of the new and old files by other users. If another user already has the old file open with FUN=UPDATE or FUN=CREATE, the alternate return is taken.

OLDFILE - Requests that the existing file be opened to end-of-file. This is similar to opening a file with FUN=UPDATE. If the user is not allowed to write records to the existing file, then the altreturn is taken. If a file of the same name does not exist, a new file is created.

The default is EXIST=NEWFILE.

EXPIRE = {ddd|NEVER}. Specifies the number of days to retain the file (ddd) or that the file is never to expire(NEVER). Files may be automatically purged from the public disk file system if they have expired whenever secondary storage space passes below an installation established threshold. The value specified may not exceed the maximum expiration period authorized for the user. If the maximum expiration period is exceeded or unspecified, the default expiration period authorized for that user will be used.

For labeled tape volumes in protected systems EXPIRE specifies the number of days to protect the volume against content changing operations (UPDATE or CREATE). In semi-protected systems, unexpired volumes require an OVER keyin to UPDATE or CREATE open. EXPIRE has no effect in unprotected systems. The expiration date of the entire volume set is that of the first file of the volume set.

If EXPIRE is not specified, the default value, as established in the authorization record for the user, determines the expiration date.

If the DCB is open to a deleted file (see M$OPEN, NXTF and SRCHCOND.DELF) the UTS of deletion time is returned here. (UTS, defined in Section 4 for M$TIME, is the standard CP-6 notation for time.)

FSN = VALUE-DEC(1-9999). File sequence number for labeled tape indicates the position of the tape file relative to the beginning of the set with first file of volume set numbered 1. For IN or UPDATE opens, FSN indicates the number of the desired file if no name is specified and the number of the file at which to begin searching for the named file if a name is specified. FSN is supplied for CREATE opens.

FUN = OPTION. Applies to comgroups, disk and tape files used to read, update or create a file. Unless otherwise specified, record positioning is to the first data record in the file. For devices, FUN specifies the I/O operations to be allowed.

IN – Used to read records from a file. Specifies the read only mode.

UPDATE – Used to update records in a file. Specifies the read and write mode. On opening an existing UR file for UPDATE, the file will be positioned at the end, not the beginning as for non-UR files.

CREATE – The type of file open that CREATE performs will depend on the parameters of the M$OPEN option EXIST. See EXIST for details. Specifies the write and read mode. If the file is cataloged at M$OPEN time (CTG=YES), the function effectively changes to UPDATE (FUN=UPDATE). This means that any of the SHARE options specified will apply.

For the default, see M$OPEN, the DCB parameter.

GHSTACS = VALUE-DEC(0-15). Specifies a function code to be passed to the system access ghost prior to opening the DCB to a disk file. The only currently implemented code is JRNLOPN, which asks the ghost to open a journal, either as the owner of this journal (see M$DCB.ACS) or to journal this comgroup (see FPT_CGCTL.JRNLSTA). The value NONE may be used to reset to zero a previous non-zero value (either in the DCB for M$OPEN, or in the file for M$CLOSE). The default is zero, which causes M$OPEN not to call the ghost.

IXTNSIZE = VALUE-DEC(1- ?). For FUN=CREATE disk files or comgroups, size in granules of the initial extent. Not applicable to RELATIVE files. The default is 2.

JRNLBYPASS = {YES|NO}. YES specifies that for opens of comgroups that have the journaling attribute, journaling is not to be done for this open. AU=YES must also be specified. The default is NO.

KEYL = VALUE-DEC(1-255). length of the key in the record. The default is 0.

KEYTYPE = {FLDID|COORD|BIN10|BINHLF|BIN521|STRING}. This field specifies the default KEYTYPE for reads and writes to a form device and the KEYTYPE for ORG=SE. The KEYTYPEs are described below:

FLDID refers to a field identifier (i.e., FPT_DCLFID.ID, a byte-aligned 2-byte value). See M$DCLFLD for details.

COORD refers to the coordinates of a field (i.e., a byte-aligned, 2-byte structure consists of the line and column of the field's location).

BIN10 specifies that KEYINCR is to be divided by 10 until it fits. If KEYINCR=1000, then the first of 1000, 100, 10, and 1 that fits is used.

BINHLF specifies that KEYINCR is to be halved. If KEYINCR=32, then the first of 32, 16, 8, 4, 2, and 1 that is small enough will be used.

BIN521 specifies that division by 5, 2.5, and 2 is to be repeated. If KEYINCR=1000, then the sequence 1000, 500, 200, 100, 50, 20, 10, 5, 2, and 1 is used.

STRING specifies that key incrementation is not possible.

KEYX = VALUE-DEC(0-32K). For ORG=INDEXED, position of key in the record (byte offset).

LINES = VALUE-DEC(1-32K). Specifies the number of printable lines per page. The default is the FORM definition value.

LOAD = {YES|NO} is meaningful only for files with alternate indices. If LOAD=NO, then all alternate indices, along with the primary index are updated every time a record is written.

If "LOAD=YES", then only the primary index, and any index that is specified as being "UNIQUE" is updated when a record is written. Unless the user specifies otherwise, the alternate indices are updated when the file is closed. The default is NO.

LRDL0 = VALUE-DEC(1-511). Specifies the number of contiguous blocks that can be added to the KEYED, INDEXED, or IREL disk file's level 0 index since the current higher-level index structure was created. If the specified number is exceeded, the higher-level index structure will be rebuilt when the file is closed. A value of 511 specifies that the upper level will never be rebuilt for this reason. The default is 3.

LSLIDE = VALUE-DEC(1-511). Specifies the number of blocks that can be added to the KEYED, INDEXED, or IREL disk file's level 0 index since the current higher-level index structure was created. If the specified value is exceeded, the higher-level index structure will be rebuilt when the file is closed. If a value of 511 is specified, the higher-level index structure will never be rebuilt for this reason. The default is 510.

LSTAOR = {YES|NO} Specifies, for comgroups only, when the VLP_STATION pointed to by DCB.LASTSTA$ is to be updated. If YES, it is updated only on reads. If NO, it is updated on both reads and writes. Default is NO.

MAXVOL = VALUE-DEC(1-511). Specifies the number of extra tape volumes which will be requested as scratch tapes after the volumes in the serial number list are used up. The default is 0.

NAME= VALUE-CHAR(1-31) Specifies the name of the file to which the DCB is to be assigned. The named file will be maintained on disk storage or labeled tape volumes. For FIXED or VARIABLE tape file names, lower-case alphabetic characters are not allowed and, to conform to ANSI standards, this rule must be observed: a name must be composed of at most 17 characters including upper case alphabetic characters, numeric characters, space, and the following special characters: ! " % & ' ( ) * + , . / ? ; : < > = -. For other tape files and NAMED (see DISP) disk files, the name may consist of up to 31 alphanumeric characters from the following character set: A-Z,a-z,0-9,:,$,_,-. The name is specified in TEXTC form. For "star" disk files, the first character is an asterisk and may be followed by up to 30 arbitrary bytes. For FEP real device handlers (see ORG option below), NAME locates the handler name.

NRECS = VALUE-DEC(1- ?). For FUN=CREATE of a relative disk file, specifies the number of records in the file. When FUN=IN or UPDATE, this field is set to the number of records in a file.

ORG = OPTION. File organization.

When ASN=FILE, COMGROUP or TAPE, ORG is meaningful for FUN=CREATE opens only.

CONSEC specifies that the records in the file are consecutively organized and each record will be processed sequentially.

DBGCG is legal for ASN=DEVICE issued from the debugger domain only and specifies that this DCB is to be opened as a pathway across which the debugger will communicate with its FEP counterpart in order to debug a FEP user. The FEP user to debug is specified by the NAME, ACCT, PSN and SETSTA parameters which are used to describe a STATION on a COMGROUP that is to be debugged.

DBGDCB is legal for ASN=DEVICE issued from the debugger domain only and specifies that this DCB is to be opened as a pathway across which the debugger will communicate with its FEP counterpart in order to debug a FEP user. The FEP user to debug is specified by the DBGDCBNO (in the M$OPEN FPT) which specifies the DCB number to which the FEP user is connected.

DBGSYSID is legal for ASN=DEVICE issued from the debugger domain only and specifies that this DCB is to be opened as a pathway across which the debugger will communicate with its FEP counterpart in order to debug a FEP user. The FEP user to debug is specified by the DBGSYSID (in the M$OPEN FPT) which gives the FEP user's sysid and by RES='FEnn' specifying that the user resides on node nn, or by RES='FE ' in conjunction with the NODENAME (at M$OPEN) which gives the name of the node on which the user resides.

FIXED specifies ANS or EBCDIC tape format F - fixed-length (RECL) records with no control information. (See EBCDIC, CNVRT, BLOCKED.)

FPRG is legal for ASN=DEVICE and specifies FEP program access. The user program reads and writes simple strings of data which are the outputs of and inputs to FEP programs which control the actual presentation of the data on the media. If VFC is specified on an M$WRITE, the first character of the buffer is discarded. Any character in the data (including TAB) is sent unmodified to the FEP program. Appropriate forms programs may be invoked via the M$SETFP system service or via the FPRG variable on this FPT_OPEN.

FREE specifies real free tape (i.e., not managed tape for which FIXED or VARIABLE must be specified).

HANDLER specifies a real device handler and is legal for ASN=DEVICE only. RES must be either 'FEnn' specifying the handler be started on node nn, or RES is 'FE ' specifying the NODENAME at M$OPEN to identify the node on which the handler will reside. FEP HANDLERS are described in the FEP Monitor Services Reference Manual. The FEP HANDLER will continue to run even after this DCB is closed. Starting of FEP HANDLERS requires the MFEP privilege. (NAME specifies the name of this handler.)

IDS is a special case of RANDOM which is formatted by the IDS processor.

INDEXED is a special case of KEYED in which the key value is contained within the record data, is of fixed length, and may be as large as 255 characters. More than one index may be specified via the ALTKEYS option at M$OPEN.

IREL is an enhanced form of indexed files in which each key may be made up of more than one part of the record, and each key fragment may have a data type associated with it. See the IRKEYS option of M$OPEN and VLP_IRKEYS (for the key definitions). NOTE: IREL files are not implemented in C00.

KEYED specifies that the location of each record in the file is determined by an explicit identifier (key) that may be used to access the record directly. A key may consist of up to 255 characters.

RANDOM specifies that the records in the file are a collection of 1024 word granules. The first word of each granule is used by file management to indicate whether or not the granule has been written.

RELATIVE specifies that the file space is preallocated for NRECS number of records, the records are of fixed length(RECL), and may be accessed sequentially or by specifying the relative record number within the file.

RESTRICT is for internal system use only.

SE specifies that a video display terminal is to be operated in a screen-editing mode. ASN=DEVICE and RES=UCnn must be specified, and the UCnn must be used only in this DCB. In this mode, records are uniquely identified by 1 to 4 byte(KEYL), fixed-length keys. Several records are displayed on the screen at a time, and a call to M$READ permits the user to position and edit them at will, using the normal terminal editing facilities, until one of them is changed, at which time the M$READ completes, returning the key and the new data for the changed record (KEYR must be set to obtain the key, of course). The M$READ will also complete if the user enters any activation character. After

M$READ, the DCB.ACTPOS field records the position of the cursor in the record as a byte index, starting at zero. If the user "joins" two adjacent records on the screen, the active M$READ completes, returning the new record, and the next M$READ altreturns with E$RECORD_DELETED and the key of the eliminated one. The FEP maintains a small cache of records whose size defaults to twice the length of the window, or the length of the screen, whichever is larger, unless NRECS specifies a non-zero value less than 50. This cache is replenished when needed by means of the E$RECORDS_NEEDED altreturn on a call to M$READ. For this altreturn, DCB.ARS contains a signed number, indicating by its value how many records are needed, and by its sign which ones. If DCB.ARS is positive, the records should be adjacent to, and greater than, the key that was returned. Otherwise, they should be less than the key, but still adjacent to it. The records are supplied with M$WRITE calls. If insufficient records are available, M$WEOF is used to abort the "records needed" phase. It is also possible for a call to M$READ to altreturn because the user attemped to insert a new record, and the FEP was unable to conjure up a key for it. In this case, the E$SPLIT_NO_ROOM error code accompanies the key, data, and cursor position of the attempted record-split. The program might then want to renumber some records by means of M$DELREC and M$WRITE. To eliminate the need for the user to reenter the split request, the program should also perform the split operation and reposition the cursor appropriately. M$SINPUT is used to position the cursor. M$ERASE removes all records from the screen and the cache. M$DEVICE causes the screen to be updated to reflect any changes made by program calls to M$WRITE and M$DELREC. The screen is otherwise only updated when M$READ begins. M$STRMCTL and M$GTRMCTL are also applicable in the ORG=SE mode, particularly with VLPTYPE=EDTCTL.

FORM specifies that an FEP-attached terminal or line printer is to be operated using a field-oriented form. The user program declares the position on the screen or page of each field and its attributes using the M$DCLFLD monitor service. The attributes may later be modified using M$MDFFLD. The data in the field may be erased with M$ERASE, or the entire field declaration released using M$RLSFLD. M$SLCFID is used to select which fields are to be modified, erased, released, or read from. An M$READ returns the data from an input field that is selected and has had data entered into it by the terminal operator. When all input data has been returned, an end-of-file ALTRET will occur. M$WRITE allows the program to display data in a field.

SYMB specifies a special type of UR file used by the operating system. It is not useful for users.

TERMINAL is legal only for ASN = DEVICE or COMGROUP and specifies terminal control type access. The page control formatting available in ORG = UR is not available, although TAB and VFC (if appropriate) still apply. Unlike ORG = UR, characters are sent untranslated to the media, and the TRANS option on M$WRITE controls whether the media handler is to manipulate the output (e.g., append CR,LF). The terminal control monitor services (e.g., M$STRMATTR) are available only via DCBs with TERMINAL organization.

UR specifies a Unit Record formatted file. All options available for formatting unit record output devices (such as HDR, TAB, FORM, etc.) are available in UR files. UR is the default organization for unit record devices. Whenever an M$WRITE is done through a DCB with ORG = UR, the written data is translated into the 96 ASCII graphics and TAB with all characters outside this set translated to space. This translation is inhibited by specifying TRANS = YES on the M$WRITE and this is the only effect of TRANS in this organization.

VARIABLE specifies ANS tape format D for unspanned files, ANS tape format S for spanned files, and format V for EBCDIC files - variable length records. (See EBCDIC, CNVRT, BLOCKED, SPANNED.)

The default for disk and tape files is CONSEC. The default for comgroups is TERMINAL. The default for free tape is FREE. The default for other devices is UR.

PASS= VALUE-CHAR(8) Is 8 characters with no restriction on the choice of characters. Passwords do not apply to FIXED or VARIABLE tape files, or to STAR disk files.

The default is blanks which means no password is required.

PSN = VALUE-CHAR(6) Specifies for a disk file, the pack set name if the account is not in public storage. For labeled or free tape, this parameter specifies the serial number of the volume currently mounted.

The default is blanks which means public disk storage or scratch tape.

QISS = {YES|NO} is meaningful only for CREATE opens of comgroups. YES specifies that the station tree is to be the queue tree. NO specifies that the message type tree is to be the queue tree. The default is NO.

RECL = VALUE-DEC(1-32K). For fixed record length formats such as RELATIVE and FIXED, RECL is the maximum data record length in bytes. For VARIABLE formats, RECL is the maximum record segment length in bytes. For unspanned files, at most one record segment is used per record, thus limiting each record to RECL bytes which includes 4 bytes of record information. For spanned files, RECL is the maximum length of the data record not including record information bytes. RECL may be zero for spanned files indicating that there is no maximum.

RECL is determined for IN or UPDATE opens from the file opened except for managed tape files for which RECL must be specified.

If no RECL (0) is specified for unspanned files, a format dependent default will be supplied. The maximum is also format dependent.

For complete information on options related to creating ANS ASCII or IBM ECBDIC tapes, see Host Monitor Services Reference Manual (CE75), Appendix F. That appendix also specifies the maximum size for RECL that is also format dependent.

RES = VALUE-CHAR(4) Specifies a device ('dvnn' or 'dv'). dv is the 2-character device mnemonic or special name (e.g., MT for tape, special name ME). nn is a 2-digit number identifying a device allocated via the RESOURCE command, a reserved logical device name (such as LP01 or CP01), or a logical device name defined via the LDEV command or M$LDEV service. FE[nn] specifies opening of either a real device handler or a FEP-user debugging path. nn, if specified, is the FEP number.

The reserved device types and special names are as follows:

7T[nn]	FT[nn]	LT[nn]
CR01	JE	ME
CG	JF	MT[nn]
CP[nn]	LP01	NO
CR[nn]	LO	CP01
DP[nn]	LP[nn]	UC[nn]
FE[nn]		

(See the Files, Devices, and Comgroups section in the CP-6 Programmer Reference Manual for detailed information.) When RES is a 2-character device mnemonic, see the WSN parameter which influences the disposition of output to the device.

SEQ = {YES|NO}. YES specifies that sequencing is to occur on each output record. The default is NO.

SEQCOL = VALUE-DEC(1-255). number in which sequencing is to be performed. The sequence ID is printed followed by the record sequence number. Sequencing is controlled by the SEQ = YES parameter. The default SEQCOL is 73.

SEQID = VALUE-CHAR(4). Specifies a 4 character sequence identification which is to be appended to each output record when record sequencing has been requested. The default is blank characters.

SHARE = OPTION. Is meaningful for disk files and comgroups only, and is applicable for disk files only if ORG is not CONSEC or UR. SHARE restricts concurrent usage of a file by multiple DCBs. The SHARE options are: NONE, IN, and ALL. File management allows a DCB to be opened to a file only after examining the FUN, EXIST, and SHARE parameters specified for this user and the FUN and SHARE options in effect for any other DCBs currently open to the file. Table 3-2 shows the conditions under which a DCB is permitted to open the file that is to be shared with other DCBs. If parameters on this open call are incompatible with current usage of the file, the alternate return is taken with a 'file busy' indication given. If this open succeeds, the FUN and SHARE parameters on this open may alter or set the current usage status of the file. (See Table 3-2 in the description of M$OPEN for complete information on this topic.)

The default is NONE for disk files and ALL for comgroups.

SPANNED = {YES|NO}. YES specifies whether logical records may be divided between physical tape records. Spanned records may exceed the record limit imposed by RECL since RECL applies to record segment size. SPANNED applies only to VARIABLE labeled and managed tape files. SPANNED is determined for IN or UPDATE opens from the tape file opened. Spanned and unspanned files may be freely intermixed on a volume. SPANNED should be used in conjunction with BLOCKED for optimum tape record utilization. (SPANNED must be specified for INPUT or UPDATE managed tape files.)

The default is YES which results in the most efficient usage of tape records besides permitting records of any length.

SPARE = VALUE-DEC(1-511). Specifies in words the amount of spare space to be left unused at the end of each index block while a KEYED, INDEXED, or IREL disk file is being created or updated with sequential access. The value specified may not exceed 511 words. If it does, it is treated modulo 512. This spare space is reserved so that additional keys can be inserted in a minimum time when updating the file with direct access (as in EDIT). If the file will never be updated with direct access, a spare value of 1 should be specified. The default is 1.

SYSID = VALUE(1-65535). This contains the SYSID for the FPRG.

TYPE = VALUE-CHAR(2). Is the processor specified file type supplied for convenience and is not used by file management. This 2-character field may be supplied by users or system processors when creating files. See M$OPEN, TYPE for the list of defined file types. The default is blanks.

UOPT0 - UOPT8 = {YES|NO}. Specify flags controllable by !SET, M$DCB, and M$OPEN for use by programs. These flags are ignored by the monitor and are available to pass information to a DCB. The default is NO.

VOL = VALUE-DEC(1-511). On open, VOL specifies which volume of the volume set (as specified by the serial number list) is to be initially mounted. A value of 0 or 1 indicates the first volume, 2 the second, etc. VOL is used in conjunction with XTEND to determine which volume of the volume set to extend. VOL indicates which volume is currently mounted if the DCB is open.

VOLACCESS = OPTION. Indicates labeled tape volume set access limitations.

NONE specifies that only the owner of the volume set (account of user creating first file on the volume set) will be allowed access to this volume set.

IN specifies that all may read the volume set but only the owner may write on it.

ALL specifies that all may read or write the volume set.

WSN= VALUE-CHAR(8). Specifies a workstation name or the '@' symbol to mean the user's workstation of origin. This parameter is meaningful in combination with the RES parameter when RES is set to a 2-character device mnemonic. When WSN is specified, output is sent when the associated DCB is closed or at job step (rather than at end-of-job, as happens if WSN is blank). Also, when WSN is specified, output through multiple DCBs open to the same RES is kept separate (instead of being merged, as occurs if WSN is blank).

NOTE: WSN must be blank if RES is to specify a special name, a logical device defined by LDEV or M$LDEV, a device allocated by the RESOURCE command, or the reserved logical device name LP01 or CP01.

The default is blanks.

XTEND = {YES|NO}. For CREATE opens of labeled tape files YES causes the volume set to be positioned after the last file if VOL = 0. If VOL is non-zero, the volume selected is positioned after its last file. XTEND = NO causes the next file to be created at current volume position. The default is NO.

XTNSIZE = VALUE-DEC(1-2**17). For FUN=CREATE disk files, size in granules of the secondary extents (used to add on extents). Not applicable to RELATIVE, RANDOM or IDS organizations. The default is 2.

A listing of additional fields in the DCB follows. These fields are not set via parameters for M$DCB but are available for access by users.

ACTPOS. Contains the editing position within the input record at which an ORG=SE M$READ operation caused information to be returned to user. This is a REDEF of the FLDID area.

ALTKEYX. Contains the offset into the read-only segment of a FITALTKEYS area describing the alternate key structure of an indexed file.

AMR. Set if !SET command information has been merged into the DCB.

ARS. Actual record size read in bytes. Number of records skipped by M$PRECORD. Number of records deleted by M$DELREC. Approximate maximum record size in bytes after M$OPEN, to the nearest page size.

ATTR.APL. Set if device prints APL characters.

ATTR.BIN. Set if BIN is legal on M$WRITE.

ATTR.GRLGL. If set, output through this DCB is treated as if it is for a graphics device. This bit is automatically set in all graphic form records and propagated down.

ATTR.LOWERCASE. Set if device prints lower-case.

ATTR.NATL. Set if translation should be done according to the NATIONAL translation table. This table is patchable by rumming the monitor variable NK_NATLTBL which is 512 characters long and is the translation table used in an internal CALL XLATE. See Table E-5 in Host Monitor Services Reference Manual, Volume 2 (CE75).

ATTR.NOXLATE. Set if there should not be any translation done on output to this device.

ATTR.TRANSLGL. If set, the user can write out through this DCB transparently. This will only affect symbiont files.

ATTR.TRUOVRPRT. Set if true overprinting is possible on this device. On devices which cannot inhibit upspace or CRTs with destructive overprint, this bit is reset signifying that overprinted graphics will be represented by '*'. Not currently implemented.

ATTR.UPPERCASE. Set if all output should get translated to uppercase before delivery to output file or device.

AUTHFFLG. is a collection of access control flags set by M$OPEN when it processes account security checks and opens the DCB. These flags are different from those at DCB.FFLG in that they depend only on the access control settings, not on any other items in the DCB (like DCB.FUN). This area is a REDEF of DCB.FLDID.

AUTHFFLG.AU. Meaningful for comgroups only. If set, this user is allowed to be the administrative user.

AUTHFFLG.AURD. Meaningful for comgroups only. If set, this user is allowed to issue monitor services normally reserved for the administrative user that examine but do not change the comgroup.

AUTHFFLG.CREATE. If set, the opening user is permitted to create new files in the DCB's account.

AUTHFFLG.DELF. If set, file may be deleted or its name, password, or access control may be changed (via these options on M$CLOSE: DISP=RELEASE, NAME, PASS, ACCESS, or ACSVEH).

AUTHFFLG.DELR. If set, existing records may be deleted.

AUTHFFLG.EXEC. If set, DCB was opened by a system process(or) named in the access vehicle list for this file. It indicates (to such processes) that restricted access has been invoked and requires that subsequent operations through the DCB be done by the same process that opened it.

AUTHFFLG.NOLIST. If set, file appears not to exist.

AUTHFFLG.READ. If set, records may be read.

AUTHFFLG.REATTR. If set, disk file attributes may be modified at M$CLOSE.

AUTHFFLG.TCTL. Meaningful for comgroups only. If set, this user is allowed to issue terminal control monitor services (such as M$PROMPT and M$STRMATTR). For ORG=FPRG comgroups, M$SETFP is allowed.

AUTHFFLG.UPD. If set, existing records may be updated (overwritten).

AUTHFFLG.WNEW. If set, new records may be written.

BUPF. Set if any backup is possible in this account.

CHANTIME. Contains the accumulated channel time in microseconds since this T & D DCB was opened.

CODE16$. Points to a CODE16 area elsewhere in the read-only segment that contains information about current position of the unit-record medium this DCB is writing on. CODE16$ is ADDR(NIL) if such a CODE16 is not available.

CONNCT. Number of connects accumulated on this T & D DCB since it was opened.

DCT$. Points to the Device Control Table (DCT) for the device to which this T & D DCB is open.

DDEV. Is a collection of flags set by M$OPEN. These flags are provided to help the user of the DCB to take appropriate action.

DDEV.CP. If set, this DCB has card punch characteristics; card sequencing may be meaningfully specified.

DDEV.GR. If set, output written through this DCB is treated as if it is going to a graphic device.

DDEV.IC. If set, this DCB has interactive terminal characteristics, in the sense that input is already displayed.

DDEV.LP. If set, this DCB has line printer characteristics; vertical format control applies, and the M$LINES question (where am I on the page?) may be meaningfully asked.

DDEV.UCOUT. If set, output written through this DCB is being written to the user's interactive terminal.

DDEV.XEQ. If set, the last record read through this DCB was a command created by the !XEQ process.

DESC.ACTIVE. If DESC.QS# and DESC.FETCH# are both true, the FETch request was made by a user with the FMSEC privilege active at JOB rather than as a function of the EFT processor privileges. This will allow a privileged user to override the account granule limits. If DESC.QS# is true, and DESC.FETCH# is false, and DESC.ACTIVE# is false, the file granules will be released as a function of the close when the file is closed after being ARChived.

DESC.ARCHIVE. Set if file is known in the ARCHIVE system.

DESC.BAD. Set if file inconsistency found.

DESC.BUF. Set if file to be backed up.

DESC.BUPM. Set if the file is to be backed up if modified.

DESC.DELF. Set if a deleted file was found on NXTF open with SRCHCOND.

DESC.FETCH. Meaningful only if QS is set. Set if file needs to be RETRIEVED to service a user FETCH request. Reset if file needs to be ARCHIVED to service a user STOW request.

DESC.NBUF. Set by EFT in two cases. 1) If the file's "last modified" date is less than its creation date, this is the FIT(only) for a file in an account with the ACUP attribute; if the packset is rebuilt, EFT can use this FIT to insure that it rebuilds the file with the correct "last accessed" date. 2) If the file's "last modified" date is greater than or equal to its creation date, this is a DUAL copy of a file that EFT was unable to open during its last DUAL operation, and the packsets have subsequently been switched.

DESC.NOLIST. Set if this file has NOLIST attribute.

DESC.QS. Set if EFT ARCHIVE or RETRIEVE pending. See DESC.FETCH and DESC.ACTIVE for further detail.

DESC.TERMINATE. Set if to be removed from ARCHIVE.

DESC.TPB. Set if last backup was to tape.

DEVADR.CHAN. Specifies the channel number.

DEVADR.DVN. Specifies the device number.

DEVADR.IOM. Specifies the IOM number.

DQH$. Points to the DQH for the current T & D operation.

DVBYTE. DVBYTE is returned to the DCB on reads and informs the user about the nature of the record read.

DVBYTE.BIN. Set if the record was read in binary mode. This means that data was read bit-by-bit from the medium into the buffer rather than one character per 9 bit byte.

DVBYTE.TOP. This bit has two different meanings.

If the record was read from a control command file (F$DCB.DDEV.XEQ set), this bit is set to indicate that the record is the last one in the file.

If the record was read from a unit-record file, this bit is set to indicate that the record is the first one to be printed on a page.

DVBYTE.TRANS. Set if the record was read in transparent mode. This means that no translation was performed by the system and the data is that which was read from the I/O medium. This bit is only set in cases where the default read of the record would cause a translation to be done, as with time-sharing terminals.

DVBYTE.VFC. Set if the first character of the record read should be interpreted as a VFC character.

DVTYPE contains the device type for T & D DCBs open for test and diagnostic services. The DV_ constants in NI_DATA_C define the possible values for DVTYPE.

EOMCHAR. When a read is done to a terminal EOMCHAR is filled in with the activation character. If there is no activating character or if the read is not to a terminal, EOMCHAR is set to EOM_EOR# ('501'O).

FCD. Set if the DCB is currently open.

FCI. Set if the DCB was successfully opened by the last attempt to do so.

FFLG. is a collection of access control flags set by M$OPEN when it processes account security checks and opens the DCB. These flags control the I/O that a user may do when the DCB is open.

FFLG.AU. Meaningful for comgroups only. If set, this user is allowed to be the administrative user.

FFLG.AURD. Meaningful for comgroups only. If set, this user is allowed to issue monitor services normally reserved for the administrative user that examine but do not change the comgroup.

FFLG.CREATE. If set, the opening user is permitted to create new files in the DCB's account.

FFLG.DELF. If set, file may be deleted or its name, password, or access control may be changed (via these options on M$CLOSE: DISP=RELEASE, NAME, PASS, ACCESS, or ACSVEH).

FFLG.DELR. If set, existing records may be deleted.

FFLG.EXEC. If set, DCB was opened by a system process(or) named in the access vehicle list for this file. It indicates (to such processes) that restricted access has been invoked and requires that subsequent operations through the DCB be done by the same process that opened it.

FFLG.NOLIST. If set, file appears not to exist.

FFLG.READ. If set, records may be read.

FFLG.REATTR. If set, disk file attributes, with the exception of ACCESS and ACSVEH, may be modified at M$CLOSE. For modification of ACCESS and ACSVEH, see FFLG.DELF.

FFLG.TCTL. Meaningful for comgroups only. If set, this user is allowed to issue terminal control monitor services (such as M$PROMPT and M$STRMATTR). For ORG=FPRG comgroups, M$SETFP is allowed.

FFLG.UPD. If set, existing records may be updated (overwritten).

FFLG.WNEW. If set, new records may be written.

FIELDX. Index into the read-only segment of the start of VLP_RECFIELD.

FLDID. Contains the field ID reported by the most recent operation for an ORG=FORM DCB. It is the same as that returned in KEY if KEYTYPE=FLDID, but is always returned here. This is particularly useful for operations like M$DCLFLD that have no provision for returning a key.

FLPW. Contains the final List Pointer Word (LPW) from the channel mailbox.

FORM$. Points to a VLP_FORM elsewhere in the read-only segment that names the form to be mounted on this unit record device. The form name is obtained from the !SET or !LDEV commands, or from M$OPEN or M$LDEV.

FPRGX. Index into the read-only segment of the VLP_FPRG associated with this DCB.

HEADER$. Points to a VLP_HDR elsewhere in the read-only segment that contains header information for a unit record device. The information is provided by the !SET or !LDEV commands, M$OPEN or M$LDEV.

KFIELDX. index into read-only segment of the start of FMG$KFIELD.

LASTSTA$. Points to a VLP_STATION elsewhere in the read-only segment that describes the last message read or written via this DCB through a COMGROUP. If the destination station of an M$WRITE is not supplied on the write or in SETSTA above, LASTSTA is the default.

MPC$. Points to the Device Control Table (DCT) for the Micro Programmed Controller (MPC) to which this T & D DCB is open.

PATH.CHAN. Channel number for current operation for T & D DCBs.

PATH.IOM. Logical IOM number for current operation for T & D DCBs.

PBITX. Index into read-only segment of a temporary area for storing presence bits.

RDL0. Number of granules read after reaching level 0 of the index of a KEYED or INDEXED disk file.

RETRYCNT. Accumulates the number of retries when physical IO errors occur. This field is set to zero when the DCB is opened and incremented by the retry count whenever a recoverable IO error occurs.

SETSTA$. Points to a VLP_SETSTA elsewhere in the read-only segment that supplies STATION and MSGTYP defaults for COMGROUP I/O through this DCB. SETSTA$ is initialized via the !SET command or M$OPEN FPT.

SLIDE. Number of granules on level 0 not in level 1 of the index of a KEYED or INDEXED disk file.

TAB$. Points to a VLP_TAB elsewhere in the read-only segment that describes TAB settings for a UR or TERMINAL DCB. The VLP is obtained from the !SET or !LDEV command, or from M$OPEN or M$LDEV.

TDFLG. Test and Diagnostic flags.

TDFLG.SDSK. If set, the T & D DCB is open to a disk that is part of a CP-6 packset. Write access is only allowed to the T & D cylinder.

TDSTAT. Receives the hardware status doubleword. In addition to the hardware status, TDSTAT may be used to report the following conditions:

Lost Interrupt - The major status code is 8 (decimal). Lost interrupts are detected by a poller whose period is 5 seconds.

System Fault - The major status code is 15 (decimal). In this case the second word of TDSTAT is the hardware fault word (reported via the Fault channel) with bit 0 set. T&D DCB only.

TYC. Type completion codes returned to the user upon completion of an I/O operation through the DCB.

TYC.BLNKTP. Set if blank tape on read.

TYC.BOF. Set for beginning of file.

TYC.BOT. Set for beginning of tape.

TYC.CDALRT. Set if a byte with high order bit set was detected during a TAPE write of a format which retains only the lower eight bits.

TYC.CGCRCW. Set on COMGROUP READ if a latched, continued write is awaiting its last segment to be written.

TYC.CGCWRV. Set on COMGROUP WRITE if continue is specified, and a wild-carded destination is supplied, or a different message type is supplied from other writes for this same message. Set on COMGROUP READ if CONT was specified but there is no record to continue reading.

TYC.CGFULL. Set on WRITE if the COMGROUP is full; set on READ if the COMGROUP is too full to allow fetching messages.

TYC.CGKEYV. Set for COMGROUP key violation; that is, read or write specifying illegal message type or station.

TYC.CGLWRV. Set for COMGROUP LATCHed write error; no input message to latch the write to, or latched input message spawn depth or spawn count exhausted, or wild-carded destination specified.

TYC.DACT Set when I/O is cancelled by deactivation.

TYC.DI. Set for file data inconsistency.

TYC.EGV. Set if TYC not yet returned to user.

TYC.EOD. Set if end-of-data encountered.

TYC.EOF. Set for end of file.

TYC.EOT. Set for end of tape.

TYC.FRAW. Set if read after forward write on tape.

TYC.IOERR. Set for I/O medium error.

TYC.LAST. Set if an error occurred on previous operation after the status was returned.

TYC.LD. Set for lost data.

TYC.LDISC Set upon line disconnect.

TYC.MTRAP. Set for memory trap during I/O.

TYC.OPER. Set if tape error occurred that lost position.

TYC.PROT. Set if tape is write-protected.

TYC.TIMO. Set when read times out.

TYC.XTRARD. Set if read with read pending.

VFCCHAR = VALUE-CHAR(1). The VFC character from the record. See the STRPVFC option in M$READ.

WIDTH. Set to the number of columns on a printable line or card. Zero if WIDTH is meaningless for this device; e.g., zero for files.

WSR. Set to the OPENer's working space register. If the open was performed for an access vehicle (DCB.FFLG.EXEC set), subsequent access thru this DCB can only be by the access vehicle (i.e., same WSR).

M$GETDCB - Build DCB

The M$GETDCB service allocates a DCB. The M$GETDCB service returns the DCB number for the DCB which is built. If a DCB of the name specified already exists, its DCB number is returned as if it had been built.

The M$GETDCB service call is of the form:

CALL M$GETDCB (FPT_GETDCB) [ALTRET (label)];

The parameters for this service are as follows:

DCB = VALUE-DEC(5-9). This parameter is used to request the allocation of a system DCB (5-9) by special shared processors only.

DCBNAME = VARIABLE Locates an area containing the name of the DCB to be obtained. This area may be generated by invoking the VLP_NAME macro which is described later in this section.

DCBNUM = VARIABLE Locates either a half-word or a 1-word area into which the M$GETDCB service returns the number of the DCB allocated for or corresponding to DCBNAME.

MERGE = {YES|NO} YES specifies that any assignment (!SET command) for this DCB is to be merged into it if it is created. The default is NO.

TEST = {YES|NO} YES specifies that no DCB is to built if the specified DCB does not already exist. Instead, M$GETDCB returns a zero DCB number. The default is NO.

M$RELDCB - Release Closed DCB

This service releases a DCB's Read-Only segment space and deletes the DCB from the DCB pointer table. Any user DCB may be released, regardless of how it was obtained. However, a DCB must be closed before M$RELDCB is called. Once released, the DCB and its DCB number are invalid for monitor service calls. The DCB number can be obtained by calling the M$GETDCB service.

The service call is of the form:

CALL M$RELDCB (FPT_RELDCB) [ALTRET (label)];

The parameter for the service is as follows:

DCB = VALUE-DEC(5-?). Specifies the number of the DCB to be released. System DCBs(1-4) may not be released. Only shared processors may release special DCBs (5-9). Others may be released as desired. A DCB need not have been defined via a GETDCB in order to be released. The DCB must be closed in order for it to be released.

M$CORRES - Check DCB Correspondence

The DCB correspondence-checking routine allows a user to determine whether or not two DCBs are assigned to the same file or device. The routine returns normally if the two DCBs are uniquely assigned and takes the alternate return if the DCBs are assigned to the same file or device.

The service can be used, for example, by compilers to determine whether the DCB for listing output (M$LO) and the DCB for diagnostic output (M$DO) correspond to the same device, for example. If not, the compiler may write listing output through M$LO only and may write diagnostic output through M$LO and M$DO.

The service call is of the form:

CALL M$CORRES (FPT_CORRES) [ALTRET (label)];

The parameters for this service are as follows:

DCB1 = DCBNAME Name of first DCB.

DCB2 = DCBNAME Name of second DCB.

M$SETFMA - Set FM ACCOUNT/PACKSET

This service sets the packset name and account that file management services use when no packset name or account are specifically named. These fields are initially set at logon time to "no packset name" and the user's logon account.

The form of the call for this service is as follows:

CALL M$SETFMA(FPT_SETFMA) [ALTRET (label)];

The parameters for this service are as follows:

ACCT = VALUE-CHAR(1-8) Locates an area containing the textual representation of the account to be used if one is not specified in an MDCB, MOPEN or !SET. If this parameter is NIL or ERASE, both the default account and packset name will be reset.

PSN = VALUE-CHAR(1-6) Locates an area containing the textual representation of the pack set name to be used in conjunction with the above account if neither an account nor packset name is specified in MDCB, MOPEN or !SET. If this parameter is NIL or ERASE, the default packset name will be reset.

File Manipulation

File manipulation services are provided to control disk files, tape files, and devices. The related macros for generating vector-located parameters are included in a group at the end of this subsection.

As described in the CP-6 Programmer Reference Manual, the system maintains account and file directories as well as file information tables. The monitor provides file manipulation services to allow the user to operate on these structures.

For disk, these services provide extensive support for user permanent and scratch files as well as for temporary "star" files. These services permit the user to:

o Create or locate files on public storage or on a specific pack set (M$OPEN).

o Control cataloging and disposition of permanent files (M$OPEN or M$CLOSE).

o Supply password protection and access controls for named files (M$OPEN or M$CLOSE).

o Obtain portions of the file information table (M$OPEN; also see Appendix C of CE75 for the FIT contents and sevices which allow the user to view portions of the FIT).

The system supports a variety of tape file formats, including several ANS standard formats. The monitor services permit the user to:

o Define and create tape volume sets which the system supervises, assuring that the recording density is uniform and that the format is consistent for labeled tapes (M$OPEN).

o Supply password protection and access controls for labeled tape CP-6 formats.

- o Obtain portions of the file information table (FIT) for labeled tape CP-6 formats.
- o Control tape volume changes (M$CVOL).

For devices, monitor services permit the user to:

- o Acquire access to a device through the name mentioned when it was reserved on a !RESOURCE command.
- o Access unit record devices by use of special names (such as LO, ME). Special names provide appropriate defaults for on-line and batch processing which direct input or output to the correct device for each of these processing environments.
- o Control logical devices when more flexibility is needed than special names provide. Input from or output to a device is processed symbiotically, that is spooled on disk before being directed to a final destination. Two logical devices automatically defined for each user are LP01, a line printer; and CP01, a card punch.

M$FID - Convert FID to DCB File ID

For all types of files and devices, the monitor provides services to convert the file identification (FID) from the format used with the IBEX SET command to the format in which this information is stored in the DCB. The M$FID service performs this conversion; the M$UNFID service converts the information in DCB format to FID format.

M$FID is used by processors to convert a FID into a format needed to initialize a DCB via the M$OPEN monitor service. The user supplies areas to receive the file name (NAME), account (ACCT), password (PASS), serial number list (SN), work station (WSN), assignment type (ASN), and resource type (RES). Modifications to these areas are reflected in a RESULTS area, also supplied by the user in the M$FID FPT.

In addition, the M$FID service permits the user to request scrubbing of any portion of the file identification parameter area. Scrubbing sets the area to a value meaning that the parameter is not used and if it already exists in the DCB, it is to be removed. The scrub options are useful when a user wishes not to have to check the RESULTS area following the call to M$FID. The user instead calls M$FID, and barring an alternate return condition, calls M$OPEN with no intervening steps.

The M$FID service performs no character set validation for any portion of the FID. The RESULTS returned by M$FID consists of seven bits corresponding to the seven parameters created from the FID. If a RESULTS bit is set, the parameter has been filled. If a RESULTS bit is reset, the parameter is unchanged if scrubbing was not requested; or the parameter contains a NIL value if scrubbing was requested. The RESULTS bit for ASN is always set for a valid call to M$FID.

The alternate return is taken if the TEXTFID parameter does not contain a valid FID, if TEXTFID is NIL, or if any area provided to receive the converted FID is too small to contain the pertinent information. An appropriate error code is returned in the ALTRET frame if an error occurs.

The service call is of the form:

CALL M$FID (FPT_FID) [ALTRET (label)];

The parameters for this service area as follows:

ACCT= VARIABLE locates a VLP_ACCT area which may be referenced within an FPT for M$OPEN.

ASN= VARIABLE locates an ASN area which may be within an FPT for M$OPEN. This parameter is mandatory.

CHECK. specifies a mask of bits that are used to select what types of checking are to be performed to detect malformed FIDs. The name CHECK provides a superstructure for the series of mask bit fields in the M$FID FPT. The user sets mask bits via the keywords listed below. For example, CCHARS=YES sets the CCHARS.SCRUB field in the M$FID FPT. As a result M$FID will check for certain illegal characters in the FID.

CHECK.AWODF= {YES|NO} YES allows a a workstation with a disk FID, an illegal combination. The default is NO.

CHECK.CCHARS= {YES|NO} YES requests checking for certain characters illegal in a FID. Currently, this only checks for '?' in a file name. Enclosing a field (the NAME for example) in single quotes indicates that the field is to be taken literally. In that case, the checking is skipped, regardless of the CCHARS option. The default is NO.

CHECK.PAOSF= {YES|NO} YES prohibits the use of an account with a star-file, an illegal combination. The default is NO.

CHECK.PPOSF= {YES|NO} YES prohibits the use of an packset with a star file, an illegal combination. The default is NO.

NAME= VARIABLE locates a VLP_NAME area which may be referenced within an FPT for M$OPEN.

PASS= VARIABLE locates a VLP_PASS area which may be referenced within an FPT for M$OPEN.

RES= VARIABLE locates a RES area which may be within an FPT for M$OPEN. See M$OPEN for more details.

RESULTS= VARIABLE locates an area which may be generated by the VLR_FID macro discussed next. The RESULTS parameter is optional.

SCRUB. specifies a mask of bits that are used to select what parameters are to be scrubbed. The name SCRUB. is not used as a keyword but provides a superstructure for the series of mask bit fields in the M$FID FPT. The user sets mask bits via the keywords listed below. For example, SNAME=YES sets the SCRUB.SNAME field in the M$FID FPT. As a result M$FID sets the vector-located parameter NAME to a NIL value.

SCRUB.SACCT= {YES|NO} YES requests scrubbing of the ACCT area. The default is YES.

SCRUB.SNAME= {YES|NO} YES requests scrubbing of the NAME area. The default is YES.

SCRUB.SPASS= {YES|NO} YES requests scrubbing of the PASS area. The default is YES.

SCRUB.SRES= {YES|NO} YES requests scrubbing of the RES area. The default is YES.

SCRUB.SSN= {YES|NO} YES requests scrubbing of the SN area. The default is YES.

SCRUB.SWSN= {YES|NO} YES requests scrubbing of the WSN area. The default is YES.

SN= VARIABLE locates a VLP_SN area which may be referenced within an FPT for M$OPEN.

TEXTFID = VARIABLE Locates a string to parse as a fid. It must not be null. This is a TEXT string. Its length is indicated by the VECTOR that frames it. Any leading blanks are ignored.

WSN = VARIABLE Locates a VLP_WSN area. See M$DCB for definition of WSN.

VLR_FID

This macro generates a VLR area definition for the RESULTS parameter of M$FID service.

ACCT = VALUE-BIT(1) Set if ACCT was found in TEXTFID.

ASN = VALUE-BIT(1) Set if ASN was found in TEXTFID. Always set on a successful return.

CLENGTH Indicates the length (a UBIN(9) value) of each portion of the FID. CLENGTH contains the following fields: CLENGTH.ACCT, CLENGTH.NAME, CLENGTH.PASS, CLENGTH.RES, CLENGTH.SN, and CLENGTH.WSN.

NAME = VALUE-BIT(1) Set if NAME was found in TEXTFID.

PASS = VALUE-BIT(1) Set if PASS was found in TEXTFID.

RES = VALUE-BIT(1) Set if RES was found in TEXTFID.

SINDEX Indicates the length (a UBIN(9) value) of each portion of the FID. SINDEX contains the following fields: SINDEX.ACCT, SINDEX.NAME, SINDEX.PASS, SINDEX.RES, SINDEX.SN, and SINDEX.WSN.

SN = VALUE-BIT(1) Set if SN was found in TEXTFID.

TYPE Describes the input format of each portion of the FID. UMF$SIMPLE_STRING# indicates a a simple string of characters. UMF$QUOTED_STRING# indicates a quoted string. UMF$OCTAL_STRING# indicates that the field was entered in OCTAL. TYPE contains the following fields: TYPE.ACCT, TYPE.NAME, TYPE.PASS, TYPE.RES, TYPE.SN, and TYPE.WSN.

WSN = VALUE-BIT(1) Set if WSN was found in TEXTFID.

M$UNFID - Convert DCB File ID to FID

The M$UNFID service converts file identification in DCB format to a character string in FID format. The user supplies the file identification in one of two ways: by naming a DCB or by specifying parameters that correspond to the FPT for M$OPEN (that is, name, account, password, assignment, resource type, serial number list, and work station). If a DCB is specified, the M$UNFID service obtains FID information only from the DCB and ignores other parameters.

M$UNFID requires a valid assignment type in the DCB or the ASN parameter. For other unspecified identification fields, the service supplies defaults. For resource type (RES), the following defaults are used: 'DP' if ASN=FILE, 'LT' if ASN=TAPE, and 'CG' if ASN=COMGROUP. For account (ACCT), the default is taken from the file management account field in the JIT (JIT.FACCN) in the case of DCB UNFIDs or if FORCE is requested.

If the user supplies a DCB, two anomalies should be noted. The FID will not contain a password, because the password is irreversibly encrypted and does not appear in the DCB. For a closed tape DCB, only the last active serial number is reported.

The alternate return is taken from the M$UNFID service if the assignment (ASN) is invalid or is not supplied in the DCB or the ASN parameter, if the TEXTFID area is too small to contain the FID generated or if the TEXTFID parameter is not supplied.

The form of the call for this service is as follows:

CALL M$UNFID (FPT_UNFID) [ALTRET (label)];

The parameters for this service are as follows:

ACCT = VARIABLE locates a VLP_ACCT area which may be referenced within an FPT for the M$OPEN service.

ASN = VARIABLE locates an ASN area which may be a field within an FPT for the M$OPEN service.

DCB = DCBNAME specifies that the FID is to be built from information in the DCB. If no DCB is specified, the FID is built from other parameters supplied in the FPT.

FORCE = {YES|NO} YES requests that default values for PSN and ACCT should be used if no values are passed or found in the DCB. The default is NO.

LEN = VARIABLE locates an optional UBIN WORD area in which to store the returned fids length.

NAME = VARIABLE locates a VLP_NAME area which may be referenced within an FPT for the M$OPEN service.

PASS = VARIABLE locates a VLP_PASS area which may be referenced within an FPT for the M$OPEN service.

RES = VARIABLE locates a RES area which may be a field within an FPT for the M$OPEN service.

SN = VARIABLE Locates a VLP_SN area which may be referenced within an FPT for the M$OPEN service.

TEXTFID = VARIABLE locates an area in which the textual representation of the FID is to be returned. This is a TEXT string. Its size is indicated by the VECTOR framing it. After return, the number of characters actually used in this buffer is returned in LEN.

WSN = VARIABLE locates a VLP_WSN area which may be referenced within an FPT for the M$OPEN service. ('@' is not valid as a WSN parameter.)

M$OPEN - Open DCB

The M$OPEN monitor service performs the functions necessary to give a user access to the I/O medium through a DCB. For scratch disk files and for devices, the open procedure requires few parameters.

In all cases, the open procedure initializes the DCB with the parameters supplied in the FPT. A number of M$OPEN parameters are meaningful at file creation (and have no meaning if specified when opening an existing file). File attributes are transferred from the open FPT to the DCB and to the file information table (FIT) at file creation. Whenever the file is opened subsequently for input or updating functions, the open procedure obtains the attributes from the FIT and places them in the DCB for use by the monitor.

The open procedure can be complex, especially for disk and labeled tape files. Some or all of the following functions may occur during the open procedure:

o Security checking - Access permission (ACCESS) and password correctness (PASS) are checked for disk and labeled tape files. Ownership of resource or authorization to use a workstation are checked for devices.

o Cataloging - On opening a disk file for input or update functions, the file directory is checked to determine that the file exists and, if so, where it is located.

  The user may replace an existing file by creating a new file (EXIST=NEWFILE). The user may catalog the new file at open time (CTG=YES) which deletes the old file at that time also. Or the user may preserve the old file while the new file is being created; the new file can be cataloged when it is closed by specifying SAVE disposition which deletes the old file.

- o Shared access checking - Each DCB opening a file can specify that other DCBs may concurrently use the file for input or updating, or that sharing is not permitted. Whether a file open is allowed depends on the open status of any DCBs currently open to the file and options specified for the DCB attempting to open the file. These options are compared and the results of this testing either allow the open, or disallow the open, giving a 'file busy' indication and causing an alternate return. Conditions required for file sharing are summarized in Table 3-2 (see M$OPEN). When the open is allowed, the apparent status of DCBs currently open to the file is changed in some cases, as noted in the table.

- o File extension - Disk space allocation is performed automatically for some disk files. The user may specify the size of the secondary extents for keyed, indexed, IREL, unit record, and consecutive disk files (XTNSIZE). For other file organizations, the user may request file extension(M$EXTEND).

In addition to the conventional open functions, the M$OPEN service permits the user to request related functions. Among the functions available through the M$OPEN service are:

- o Obtaining file attributes and attempting security checks without actually opening the file (TEST=YES).

- o Specifying indirectly the next disk or labeled tape file to be accessed. The user can request 1) the next file following the file most recently accessed by the DCB, 2) the first file on the tape volume set or the account's file directory, or 3) a file that meets certain conditions.

- o Deleting a disk file from the catalog, which results in the release of the granules containing the file (DELETE=YES).

- o Reusing a set of file attributes from an existing file to create a new file (for example, calling M$OPEN with FPARAM retrieves the attributes; then calling M$OPEN with IFPARAM framing the FPARAM attributes can create a new file with the same attributes).

The form of the call for this procedure is as follows:

CALL M$OPEN (FPT_OPEN) [ALTRET (label)];

The options specified in the OPEN FPT override those previously specified in the DCB or on the SET command. For each parameter not specified, either the current value or the default is used.

ACCESS = VARIABLE Locates an access control list which contains the access permissions to the file for specific accounts. The list can be created by invoking the VLP_ACCESS and VLP_ATTR macros which are described later in this section.

ACCT = VARIABLE. The account in which the file is cataloged is 8 characters from the ASCII 7-bit set, excluding control characters.

ACCT locates an area which contains the account name. This area may be generated by invoking the VLP_ACCT macro. The default causes this parameter to be ignored.

ACS = OPTION. All ACS options apply to disk files; only the SEQUEN and JRNL options apply to tape files. The ACS parameter defines access characteristics for a file and restricts user access to the file only as specifically stated in the explanation of each option.

SEQUEN	When creating a keyed file the user must supply keys in ascending order. A file created with this option can be read or written with or without a key.
DIRECT	When creating or updating a keyed file, the user may supply keys in any order. A file created with this option can be read or written with or without a key.

JRNL Specifies journal access mode for a consecutive, named disk or tape file. This mode is designed to permit one user (the journal owner) to control the file while any user may write to the file. A user may become the journal owner by performing the first open of the file if neither the file nor the DCB specifies GHSTACS=JRNLOPN. Otherwise, the first attempt to open the file waits for the system ghost to open it first, thereby becoming its owner. In order to keep this file name continually available for output from users, the journal owner can periodically close and rename the journal file. Thus, other users of the file can continue to send output to the file without being aware that a new physical file was substituted for the file just closed. The file opened in journal mode is subject to these special rules:

1. The first user opening the file must specify FUN=CREATE whether the file exists or not and whether GHSTACS=JRNLOPN or not (otherwise an alternate return results). EXIST=OLDFILE must be used to extend an old file. The owner keeps the journal available by keeping this DCB open (close with rename does not close a journal DCB). (Any attempt to update the file after the owner has closed it results in an alternate return.)

2. Other users who access the file in journal mode must open the file with FUN=UPDATE or CREATE; FUN=IN causes an alternate return. The only meaningful EXIST option is ERROR.

3. For the owner and any user of the file, the following rules apply at M$OPEN:

   o DISP must be set to NAMED or must not be specified at all.
   o CTG = YES is assumed.
   o ORG must be set to CONSEC.
   o SHARE is ignored.
   o NXTA, NXTF, THISF, DELETE must not be specified.
   o For a tape file opened in journal mode, a file of the same name must not concurrently exist on disk. (When a tape journal file is opened in journal mode, a dummy file of the same name is opened on disk.)

DS2 through DS8 Specifies data-segment access for a random file. This mode permits users to share the data of a random file directly in memory. The size of the file(IXTNSIZE) must be between 2 and 257 granules, which requires between 1 and 256 pages of memory to be available for the data-segment. The first granule of the file is used to fill in the holes caused by the 1-word granule headers. M$EXTEND may be used to enlarge the file and data segment simultaneously, or if SIZ=0, to enlarge the data segment to the current size of the file. M$TRUNC may be used to cause the current state of the data segment to be saved in the file. If an attempt is made to create a 1-granule file in this mode, EXIST=ERROR is enforced and the data segment size is obtained from XTNSIZE instead of IXTNSIZE-1. Also in this case, the file cannot be cataloged (although the CTG option is honored and permits the file to be shared, and any access control specifications function normally). In addition, M$EXTEND is still used to enlarge the data segment, but M$TRUNC does nothing.

BLOCK Is for internal system use only.

The default causes this parameter to be ignored.

ACSVEH = VARIABLE Locates an access vehicle control list which contains the access permissions to the file for specific processors. By defining an access vehicle control list, the file creator can permit a processor to access the file on behalf of a user although the user's account is not granted the same permissions. The list can be generated by invoking the VLP_ACSVEH and VLP_ATTR macros which are described later in this section.

This parameter is valid only in conjunction with an access control list which includes 'EXEC' permission (restriction).

ALTKEYS = VARIABLE. Locates an area containing a VLP_ALTKEYS which describes the alternate indices to be built. Meaningful only on a create open for a indexed file.

ASN = OPTION. FILE, TAPE, DEVICE, or COMGROUP indicates whether the DCB parameters describe a DISK file (FILE), a labeled tape (TAPE), a specific device (DEVICE), or a communication group (COMGROUP). The default causes this parameter to be ignored.

AU = {YES|NO}. is meaningful only for a COMGROUP and indicates that the user is to be the administrative user. The default causes this parameter to be ignored.

BLKL = VALUE-DEC(1-32764). Specifies the maximum physical tape record size (in bytes) that will be read from or written to a labeled tape. For CREATE opens BLKL will be rounded up to a multiple of 4 (ANS formats) or 8 (CP-6 ORGs). For IN or UPDATE opens BLKL is determined from the file opened except for managed tape files for which BLKL must be specified.

The default causes this parameter to be ignored.

BLOCKED = {YES|NO}. YES specifies that logical records and record segments are to be packed into physical tape records. NO results in, at most, one record or record segment per tape record. BLOCKED applies only to FIXED and VARIABLE labeled and managed tape files. BLOCKED is determined for IN or UPDATE opens from the labeled tape file opened. Blocked and unblocked files may be freely intermixed on a volume. BLOCKED should be used in conjunction with SPANNED for optimum tape record utilization. (BLOCKED must be specified for INPUT or UPDATE managed tape files.) CP-6 format tape files are always blocked.

The default causes this parameter to be ignored.

BUPM = {YES|NO}. YES specifies that file is to be backed up if modified. The default causes this parameter to be ignored.

CNVRT = {YES|NO}. YES specifies that for EBCDIC tape files (as determined by tape labels on input and DCB.EBCDIC on output) data is to be translated from EBCDIC to ASCII after reading or ASCII to EBCDIC before writing. Translation is done while data is being moved between user and monitor buffer for all file formats. CNVRT applies only to FIXED and VARIABLE formats. For non-binary writes to free tape if CNVRT is set and EBCDIC is not, the user's buffer is converted to assure that no high order bits are set (which will cause an I/O error). If both CNVRT and EBCDIC are set for device tapes, normal translation takes place.

The default causes this parameter to be ignored.

COMP = {YES|NO}. YES specifies that the file is to have the attribute "compressed". All records written into a file will be compressed. The records will be restored when read. Compression reduces file size to approximately 60 percent of the space it would occupy if uncompressed and increases user service time required to access the file by approximately two and one half times. COMP is applicable only to CREATE opens of consecutive and keyed disk files. Encryption cannot be used on compressed files.

The default causes this parameter to be ignored.

COPIES = VALUE-DEC(1-511). Specifies the number of copies of this file to be printed. This field is copied into the CODE16.COPIES field in the fit of the opened file.

The default is 1. Values of zero will be ignored. This feature is not yet implemented.

CTG = {YES|NO}. is meaningful only for CREATE opens of disk files. This parameter controls cataloging of the file at open time.

YES specifies that the file is to be cataloged in the file directory at open time. If EXIST=NEWFILE, any existing old file of the same name is deleted from the file directory (and is released when any current users close the old file). When CTG=YES and SHARE=ALL or SHARE=IN, the new file may be concurrently opened by other users, as if it were opened UPDATE instead of CREATE.

NO specifies that the file is not to be cataloged at open time. If CTG=NO and a new file is being created (EXIST=NEWFILE, EXIST=ERROR, or EXIST=OLDFILE when the old file does not exist), the newly created file is not available for access by other users until after it is closed for the first time with DISP=SAVE.

The default causes this parameter to be ignored.

CVOL = {YES|NO}. YES specifies that the user desires to be notified (via ALTRET on M$READ or M$WRITE) of end-of-volume conditions. For labeled tapes the user may then M$CVOL to cause the next volume to be mounted or perform another I/O request which automatically causes the next volume to be mounted. CVOL is primarily useful when user trailing labels are desired and may cause the ALTRETurning M$WRITE to fail in which case it must be retried.

CVOL should only be specified for SPANNED=NO tape files since spanned tape files may have records which cross volume boundaries. These records are difficult to process if the user takes CVOL control.

For free tapes, end-of-tape will be reported to permit the user to M$CVOL to the next output tape or M$WRITE which automatically mounts the next output tape. For read operations, two file marks are interpreted to indicate end-of-volume. A read encountering two file marks will 1) if CVOL was specified, return an end-of-file error and leave tape position after the first file mark or 2) if CVOL was not specified, cause the next volume to be automatically mounted and read.

The default causes this parameter to be ignored.

DCB = DCBNAME. Specifies the name of the DCB. DCB is used to refer to the DCB in !SET commands and data manipulation FPT invocations. This parameter is required.

Certain names listed in Table 3-1 cause the monitor to provide appropriate defaults for FUN and RES when the DCB is initialized by a process other than the M$DCB macro.

Table 3-1. DCB Defaults as a function of DCBname

DCBN	FUN	RES	DCBN(d)	FUN	RES
(a)	(f)	ME	F$xxx (e)	(f)	ME
M$LO	CREATE	LO	F$101	IN	ME
M$LL	CREATE	LO	F$102	CREATE	ME
M$DO	CREATE	LO	F$103	IN	NO
M$PO	CREATE	NO	F$104	CREATE	NO
M$SI	IN	CR	F$105	IN	CR
M$SO	CREATE	NO	F$106	CREATE	CP
M$UI	IN	CR	F$107	UPDATE	ME
M$OU	CREATE	(c)	F$108	CREATE	LO
M$EI	IN	CR	F$5	IN	CR
M$EO	CREATE	NO	F$6	CREATE	LO
M$ME	UPDATE	ME	F$7	CREATE	CP
M$DR	CREATE	(g)			
#1 (b)	IN	CR			
#2 (b)	IN	CR			
#3 (b)	CREATE	(c)			
#4 (b)	CREATE	LO			

a Any other 1- to 31-character name that conforms to language requirements; refer to the appropriate language manual for character set restrictions.

Table 3-1. DCB Defaults as a function of DCBname (cont.)

IBEX also places restrictions on the DCB name specified in a !SET command; refer to the CP-6 Programmer Reference for details.

b These are actually positions on a command line. Typically, they are used by compiler/processor invoking commands to establish source (#1), update (#2), object (#3), and listout (#4) DCBs. The defaults for #1 through #4 take precedence over defaults for the DCB name, as do their (implied) SET commands.

c This DCB defaults to the 'GO' file (a disk file with the name *G).

d The FORTRAN compiler assigns FORTRAN program DCBs F$ names. Other processors or users may also create F$ DCB names.

e Any other DCB name beginning with F$.

f Depends on the first operation. If M$WRITE, FUN is set to CREATE, otherwise FUN is set to IN.

g M$DR is the DCB used to perform the DRIBBLE function, and as such, is always opened with an assign-merge record specifying a FID.

DCBCONLGL = {YES|NO} is meaningful only if ORG = CG. Specifies whether DCBs are allowed to open to this comgroup. The default is YES. See VLP_CGCP for additional information.

DELETE = {YES|NO} Applies to disk files only. YES specifies that the M$OPEN service is to delete the specified file. An open never occurs, but DELETE observes the rules of a successful FUN=IN or FUN=UPDATE open, prior to deleting a file. When FUN=CREATE, DELETE has no meaning. The default is DELETE=NO.

DENSITY = OPTION. Options are D556, D800, D1600, and D6250. The recording density at which a tape is to be written. DENSITY may be specified when creating the first file of a volume set or when the volume's density can not be determined. All following files are created at the same density as the first. For IN or UPDATE opens, DENSITY is determined from the tape volume containing the opened file unless the density could not be determined during AVR due to an I/O error.

The default causes this parameter to be ignored.

DISP = OPTION. Is meaningful for disk files for FUN=CREATE opens only. When a new file is created, it is either allowed to be cataloged as a named file or never allowed to be cataloged and is classified as a scratch file. The options are as follows:

NAMED - Specifies that when the new file is created it has been given a name that will allow it to be cataloged as a named file.

SCRATCH - Specifies that the new file is a scratch file. It will not have a name and can never be cataloged.

The default causes this parameter to be ignored.

DVFC = VALUE-CHAR(1). Specifies the default format control character for records which have no format control specified on the M$WRITE request.

The default causes this parameter to be ignored.

EBCDIC = {YES|NO}. YES specifies that tape file labels (including user labels) are to be translated from EBCDIC to ASCII on input and from ASCII to EBCDIC on output. Data is subject to translation (see DCB.CNVRT). For IN or UPDATE, DCB.EBCDIC is determined from the current volume. For CREATE it must be the same as for the current volume unless the current volume is the first volume of the volume set and is positioned to the beginning of the volume. EBCDIC files cannot be created on ASCII volumes and vice versa. EBCDIC applies only to VARIABLE and FIXED files on labeled and managed tape and to free tape. EBCDIC must be specified on output and input for free and managed tape files.

The default causes this parameter to be ignored.

EVENT = VALUE-DEC(1-?). Specifies an event number to be returned to the user of this ASN=DEVICE DCB when B$COMIO events or FPRG-related events are returned to the user. Zero indicates not to return these events. The default causes this parameter to be ignored.

EXIST = OPTION. Is meaningful at CREATE opens for disk files only. Named disk files (DISP=NAMED) are conditionally created according to the EXIST options listed next, provided there are no sharing conflicts (see SHARE parameter). Scratch files are unconditionally created (thus the EXIST parameter is not meaningful if DISP=SCRATCH).

ERROR - Requests new file creation if a file of this name does not exist. If a file of the same name exists the alternate return is taken.

NEWFILE - Requests new file creation even if a file of the same name exists. The new file replaces any existing old file if CTG=YES on the create open or if the new file is closed with DISP=SAVE.
NOTE: When EXIST=NEWFILE and FUN=CREATE, EXIST operates as if it were OLDFILE if all the following conditions are true:

1. The M$OPEN does not specify NAME.
2. The previous open of the DCB of this name in this session (not necessarily in the job step) specified FUN = CREATE, EXIST = NEWFILE.
3. A SET or RESET command for this DCB has not been issued since the previous open.

See the CTG parameter for a discussion of the types of concurrent use of the new and old files by other users. If another user already has the old file open with FUN=UPDATE or FUN=CREATE, the alternate return is taken.

OLDFILE - Requests that the existing file be opened to end-of-file. This is similar to opening a file with FUN=UPDATE. If the user is not allowed to write records to the existing file, then the altreturn is taken. If a file of the same name does not exist, a new file is created.

The default causes this parameter to be ignored.

EXPIRE = {ddd|NEVER}. Specifies the number of days to retain the file (ddd) or that the file is never to expire(NEVER). Files may be automatically purged from the public disk file system if they have expired whenever secondary storage space passes below an installation established threshold. The value specified may not exceed the maximum expiration period authorized for the user. If the maximum expiration period is exceeded or unspecified, the default expiration period authorized for that user will be used.

For labeled tape volumes in protected systems EXPIRE specifies the number of days to protect the volume against content changing operations (UPDATE or CREATE). In semi-protected systems, unexpired volumes require an OVER keyin to UPDATE or CREATE open. EXPIRE has no effect in unprotected systems. The expiration date of the entire volume set is that of the first file of the volume set.

If EXPIRE is not specified, the default value, as established in the authorization record for the user, determines the expiration date.

The default causes the parameter to be ignored.

FINDPOS = {YES|NO}. YES specifies that although an I/O error has been detected on a tape, the next HDR1 label (start of file) is to be located on a NXTF open. Note that more than 1 file may be skipped (check DCB.FSN to determine this) and that further I/O errors are ignored until a HDR1 label is found (might result in tape going off reel end). The default is NO.

FORM = VARIABLE. Locates an area containing a unit record form name. This area may be generated via the VLP_FORM macro which is described later in this section. The default is NIL.

FPARAM = VARIABLE. Locates an area where the monitor is to pass portions of the file information table (FIT). The area to receive the file attributes should be 1024 words in length to accommodate the complete set of file information. The file name, serial number and account are not passed to the user, but other permanent file attributes are returned. The library service XSF$LOCCODE is provided to give the user access to the contents of the FPARAM area. This service allows the user to specify a code describing a portion of the FPARAM data; the service returns a pointer to the desired data which allows the user to access the data by field names. For the codes, field names, and the format of the call to these services, refer to CP-6 Host Library Services Reference Manual (CE71). If NAME=ERASE is specified and NXTF is not specified, the FIT of the file directory (CODES 14 and 9) is returned.

If FPARAM locates an area that is less than 30 words long, nothing is stored in this area. On a TEST=YES open, an FPARAM area less than 30 words can be specified to cause the file information to be returned in the DCB, however.

FPARAM is not applicable for devices or for labeled tape of formats V, D, U, or F.

The default is NIL.

FPRG = VARIABLE. Locates an area containing a VLP_FPRG specifying a FEP rununit.

If this is an ORG=FPRG open of the user's terminal or of a resource device, the specified FEP rununit will be loaded as a result of this M$OPEN.

If this is symbiont device open or a file open, the VLP_FPRG specifies a FEP rununit to be loaded into the path to the destination device whenever the contents of this file are sent to a FEP device.

When an M$OPEN is performed and FPRG is specified, the profile of the destination is returned in the profile area of the VLP_FPRG structure.

The default is NIL.

FSN = VALUE-DEC(1-9999). File sequence number for labeled tape indicates the position of the tape file relative to the beginning of the set with first file of volume set numbered 1. For IN or UPDATE opens, FSN indicates the number of the desired file if no name is specified and the number of the file at which to begin searching for the named file if a name is specified. FSN is supplied for CREATE opens. The default causes this parameter to be ignored.

FUN = OPTION. Applies to comgroups, disk and tape files used to read, update or create a file. Unless otherwise specified, record positioning is to the first data record in the file. For devices, FUN specifies the I/O operations to be allowed.

IN - Used to read records from a file. Specifies the read only mode.

UPDATE - Used to update records in a file. Specifies the read and write mode. On opening an existing UR file for UPDATE, the file will be positioned at the end, not the beginning as for non-UR files.

CREATE - The type of file open that CREATE performs will depend on the parameters of the M$OPEN option EXIST. See EXIST for details. Specifies the write and read mode. If the file is cataloged at M$OPEN time (CTG=YES), the function effectively changes to UPDATE (FUN=UPDATE). This means that any of the SHARE options specified will apply.

The default causes this parameter to be ignored.

GHSTACS = VALUE-DEC(0-15). Specifies a function code to be passed to the system access ghost prior to opening the DCB to a disk file. The only currently implemented code is JRNLOPN, which asks the ghost to open a journal, either as the owner of this journal (see M$DCB.ACS) or to journal this comgroup (see FPT_CGCTL.JRNLSTA). The value NONE may be used to reset to zero a previous non-zero value (either in the DCB for M$OPEN, or in the file for M$CLOSE). The default causes this parameter to be ignored.

HDR = VARIABLE Locates an area containing a printer page header definition. The area may be generated by invoking the VLP_HDR macro which is described later in this section.

IFPARAM = VARIABLE. Locates an area containing data which is to be used for this CREATE open. The FIT data may be obtained from a previous call to M$OPEN specifying FPARAM. IFPARAM permits a user to copy all attributes of a file and to merge these attributes into the DCB before the merge of parameters from this open FPT. Thus attributes from the IFPARAM area may be overridden by the other open FPT contents. The default is NIL.

INSTATTR = VARIABLE. Locates an area containing installation attributes. Although the CP-6 system places no restrictions on setting this data, this parameter is reserved for use by the installation manager. This area can be generated by invoking the VLP_ATTR macro which is described later in this section. The default is NIL.

IRKEYS = VARIABLE. Locates an area containing Indexed Relational keys definitions. This area may be generated via the VLP_IRKEYS macro which is described later in this section. Meaningful on create opens of IREL files only. The default is NIL. NOTE: The IRKEYS option is not available in C00.

IXTNSIZE = VALUE-DEC(1- ?) For FUN=CREATE disk files or comgroups, size in granules of the initial extent. Not applicable to RELATIVE files. The default is 0.

JRNLBYPASS = {YES|NO}. YES specifies that for opens of comgroups that have the journaling attribute, journaling is not to be done for this open. AU=YES must also be specified. The default is NO.

KEYL = VALUE-DEC(1-255) For ORG=INDEXED, length of the key in the record. The default causes this parameter to be ignored.

KEYTYPE = {FLDID | COORD | NONE}. This field specifies the default key type for reads and writes to a form device. The default is NONE.

KEYX = VALUE-DEC(0-32K). For ORG=INDEXED, position of key in the record (byte offset). The default is 0.

LINES = VALUE-DEC(1-32K). Specifies the number of printable lines per page. The default causes this parameter to be ignored.

LOAD = {YES|NO}. is meaningful only for files with alternate indices. If LOAD=NO, then all alternate indices, along with the primary index are updated every time a record is written.

If "LOAD=YES", then only the primary index, and any index that is specified as being "UNIQUE" is updated when a record is written. Unless the user specifies otherwise, the alternate indices are updated when the file is closed. The default causes this parameter to be ignored.

LRDL0 = VALUE-DEC(1-511). Specifies the number of contiguous blocks that can be added to the KEYED, INDEXED, or IREL disk file's level 0 index since the current higher-level index structure was created. If the specified number is exceeded, the higher-level index structure will be rebuilt when the file is closed. A value of 511 specifies that the upper level will never be rebuilt for this reason. The default causes this parameter to be ignored.

LSLIDE = VALUE-DEC(1-511). Specifies the number of blocks that can be added to the KEYED, INDEXED, or IREL disk file's level 0 index since the current higher-level index structure was created. If the specified value is exceeded, the higher-level index structure will be rebuilt when the file is closed. If a value of 511 is specified, the higher-level index structure will never be rebuilt for this reason. The default causes this parameter to be ignored.

LSTAOR = {YES|NO}. Specifies, for comgroups only, when the VLP_STATION pointed to by DCB.LASTSTA$ is to be updated. If YES, it is updated only on reads. If NO, it is updated on both reads and writes. The default causes this parameter to be ignored.

MAXVOL = VALUE-DEC(1-511). Specifies the number of extra tape volumes which will be requested as scratch tapes after the volumes in the serial number list are used up. The default causes this parameter to be ignored.

MNTONLY = {YES|NO}. YES specifies that the initial tape volume is to be mounted without opening any files. The tape volume is positioned to its beginning. MNTONLY is only meaningful for tapes. The file sequence number (FSN) and file section number (FSECT) of the first file (section) are returned in the DCB for labelled tapes. DCB.ASN is set to TAPE for labelled tapes and to DEVICE for free or managed tapes. The default causes this parameter to be ignored.

NAME = VARIABLE. Specifies the name of the file to which the DCB is to be assigned. The named file will be maintained on disk storage or labeled tape volumes. For FIXED or VARIABLE tape file names, lower-case alphabetic characters are not allowed and, to conform to ANSI standards, this rule must be observed: a name must be composed of at most 17 characters including upper case alphabetic characters, numeric characters, space, and the following special characters: ! " % & ' ( ) * + , . / ? ; : < > = -. For other tape files and NAMED (see DISP) disk files, the name may consist of up to 31 alphanumeric characters from the following character set: A-Z,a-z,0-9,:,$,_,-. The name is specified in TEXTC form. For "star" disk files, the first character is an asterisk and may be followed by up to 30 arbitrary bytes. For FEP real device handlers (see ORG option below), NAME locates the handler name.

NAME locates an area containing the file name. This area may be generated by invoking the VLP_NAME macro. VLP_NAME specifies the name of the file to which the DCB is to be assigned.

NODENAME = VARIABLE Locates a VLP_NODENAME area which describes the node to which this device open is directed. It is meaningful only if RES='FE ' and is used to convert RES='FE ' into 'FEnn' where nn is a node number. See also RES.

NRECS = VALUE-DEC(1- ?). For FUN=CREATE of a relative disk file, specifies the number of records in the file. The default is 0.

NXTA = {YES|NO}. Has meaning only for disk files. NXTA=YES specifies that the user wants to access a different account which is obtained from the pack set account directory (PAD) if the SN parameter is specified or from the master account directory(MAD) if the SN parameter is not specified. The monitor obtains the account name following the account specified in the DCB and then changes the DCB to reflect this next account. If no account is specified in the DCB, the monitor places the first account in the directory into the DCB. If there are no more accounts available, the alternate return is taken. The default is NO.

NXTF= {YES|NO} is meaningful only for disk files or labeled tape files. NXTF=YES specifies that when the DCB is opened the monitor is to access the next file in sequence (following the one most recently accessed via the DCB). If no file name or file sequence number is specified (currently) in the DCB, the first file on the tape or in the account file directory is accessed. If there are no more files available, the alternate return is taken.

The effect of NXTF can be modified by other parameters specified for the M$OPEN call; see the parameters THISF and SRCHCOND. The default is NO.

ORG = OPTION. File organization.

When ASN=FILE, COMGROUP or TAPE, ORG is meaningful for FUN=CREATE opens only.

CONSEC specifies that the records in the file are consecutively organized and each record will be processed sequentially.

DBGCG is legal for ASN=DEVICE issued from the debugger domain only and specifies that this DCB is to be opened as a pathway across which the debugger will communicate with its FEP counterpart in order to debug a FEP user. The FEP user to debug is specified by the NAME, ACCT, PSN and SETSTA parameters which are used to describe a STATION on a COMGROUP that is to be debugged.

DBGDCB is legal for ASN=DEVICE issued from the debugger domain only and specifies that this DCB is to be opened as a pathway across which the debugger will communicate with its FEP counterpart in order to debug a FEP user. The FEP user to debug is specified by the DBGDCBNO (in the M$OPEN FPT) which specifies the DCB number to which the FEP user is connected.

DBGSYSID is legal for ASN=DEVICE issued from the debugger domain only and specifies that this DCB is to be opened as a pathway across which the debugger will communicate with its FEP counterpart in order to debug a FEP user. The FEP user to debug is specified by the DBGSYSID (in the M$OPEN FPT) which gives the FEP user's sysid and by RES='FEnn' specifying that the user resides on node nn, or by RES='FE ' in conjunction with the NODENAME (at M$OPEN) which gives the name of the node on which the user resides.

FIXED specifies ANS or EBCDIC tape format F - fixed-length (RECL) records with no control information. (See EBCDIC, CNVRT, BLOCKED.)

FPRG is legal for ASN=DEVICE and specifies FEP program access. The user program reads and writes simple strings of data which are the outputs of and inputs to FEP programs which control the actual presentation of the data on the media. If VFC is specified on an M$WRITE, the first character of the buffer is discarded. Any character in the data (including TAB) is sent unmodified to the FEP program. Appropriate forms programs may be invoked via the M$SETFP system service or via the FPRG variable on this FPT_OPEN.

FREE specifies real free tape (i.e., not managed tape for which FIXED or VARIABLE must be specified).

HANDLER specifies a real device handler and is legal for ASN=DEVICE only. RES must be either 'FEnn' specifying the handler be started on node nn, or RES is 'FE ' specifying the NODENAME at M$OPEN to identify the node on which the handler will reside. FEP HANDLERS are described in the FEP Monitor Services Reference Manual. The FEP HANDLER will continue to run even after this DCB is closed. Starting of FEP HANDLERS requires the MFEP privilege. (NAME specifies the name of this handler.)

IDS is a special case of RANDOM which is formatted by the IDS processor.

INDEXED is a special case of KEYED in which the key value is contained within the record data, is of fixed length, and may be as large as 255 characters. More than one index may be specified via the ALTKEYS option at M$OPEN.

IREL is an enhanced form of indexed files in which each key may be made up of more than one part of the record, and each key fragment may have a data type associated with it. See the IRKEYS option of M$OPEN and VLP_IRKEYS (for the key definitions). NOTE: IREL files are not implemented in C00.

KEYED specifies that the location of each record in the file is determined by an explicit identifier (key) that may be used to access the record directly. A key may consist of up to 255 characters.

RANDOM specifies that the records in the file are a collection of 1024 word granules. The first word of each granule is used by file management to indicate whether or not the granule has been written.

RELATIVE specifies that the file space is preallocated for NRECS number of records, the records are of fixed length(RECL), and may be accessed sequentially or by specifying the relative record number within the file.

RESTRICT is for internal system use only.

SE specifies that a video display terminal is to be operated in a screen-editing mode. ASN=DEVICE and RES=UCnn must be specified, and the UCnn must be used only in this DCB. In this mode, records are uniquely identified by 1 to 4 byte(KEYL), fixed-length keys. Several records are displayed on the screen at a time, and a call to M$READ permits the user to position and edit them at will, using the normal terminal editing facilities, until one of them is changed, at which time the M$READ completes, returning the key and the new data for the changed record (KEYR must be set to obtain the key, of course). The M$READ will also complete if the user enters any activation character. After M$READ, the DCB.ACTPOS field records the position of the cursor in the record as a byte index, starting at zero. If the user "joins" two adjacent records on the screen, the active M$READ completes, returning the new record, and the next M$READ altreturns with E$RECORD_DELETED and the key of the eliminated one. The FEP maintains a small cache of records whose size defaults to twice the length of the window, or the length of the screen, whichever is larger, unless NRECS specifies a non-zero value less than 50. This cache is replenished when needed by means of the E$RECORDS_NEEDED altreturn on a call to M$READ. For this altreturn, DCB.ARS contains a signed number, indicating by its value how many records are needed, and by its sign which ones. If DCB.ARS is positive, the records should be adjacent to, and greater than, the key that was returned. Otherwise, they should be less than the key, but still adjacent to it. The records are supplied with M$WRITE calls. If insufficient records are available, M$WEOF is used to abort the "records needed" phase. It is also possible for a call to M$READ to altreturn because the user attemped to insert a new record, and the FEP was unable to conjure up a key for it. In this case, the E$SPLIT_NO_ROOM error code accompanies the key, data, and cursor position of the attempted record-split. The program might then want to renumber some records by means of M$DELREC and M$WRITE. To eliminate the need for the user to reenter the split request, the program should also perform the split operation and reposition the cursor appropriately. M$SINPUT is used to position the cursor. M$ERASE removes all records from the screen and the cache. M$DEVICE causes the screen to be updated to reflect any changes made by program calls to M$WRITE and M$DELREC. The screen is otherwise only updated when M$READ begins. M$STRMCTL and M$GTRMCTL are also applicable in the ORG=SE mode, particularly with VLPTYPE=EDTCTL.

FORM specifies that an FEP-attached terminal or line printer is to be operated using a field-oriented form. The user program declares the position on the screen or page of each field and its attributes using the M$DCLFLD monitor service. The attributes may later be modified using M$MDFFLD. The data in the field may be erased with M$ERASE, or the entire field declaration released using M$RLSFLD. M$SLCFID is used to select which fields are to be modified, erased, released, or read from. An M$READ returns the data from an input field that is selected and has had data entered into it by the terminal operator. When all input data has been returned, an end-of-file ALTRET will occur. M$WRITE allows the program to display data in a field.

SYMB specifies a special type of UR file used by the operating system. It is not useful for users.

TERMINAL is legal only for ASN = DEVICE or COMGROUP and specifies terminal control type access. The page control formatting available in ORG = UR is not available, although TAB and VFC (if appropriate) still apply. Unlike ORG = UR, characters are sent untranslated to the media, and the TRANS option on M$WRITE controls whether the media handler is to manipulate the output (e.g., append CR,LF). The terminal control monitor services (e.g., M$STRMATTR) are available only via DCBs with TERMINAL organization.

UR specifies a Unit Record formatted file. All options available for formatting unit record output devices (such as HDR, TAB, FORM, etc.) are available in UR files. UR is the default organization for unit record devices. Whenever an M$WRITE is done through a DCB with ORG = UR, the written data is translated into the 96 ASCII graphics and TAB with all characters outside this set translated to space. This translation is inhibited by specifying TRANS = YES on the M$WRITE and this is the only effect of TRANS in this organization.

VARIABLE specifies ANS tape format D for unspanned files, ANS tape format S for spanned files, and format V for EBCDIC files - variable length records. (See EBCDIC, CNVRT, BLOCKED, SPANNED.)

The default causes this parameter to be ignored.

OVERFDE = {YES|NO} YES means that EFT should be allowed to make a new file over the current file directory entry. NO means that EFT should not be permitted to make a new file over a current file directory entry. The default is NO.

PASS = VARIABLE. Is 8 characters with no restriction on the choice of characters. Passwords do not apply to FIXED or VARIABLE tape files, or to STAR disk files. PASS locates an area which contains a password. This area may be generated via the VLP_PASS macro.

The default causes this parameter to be ignored.

PROCATTR= VARIABLE Locates an area containing processor attributes. Although the CP-6 system places no restrictions on setting this data, this parameter is reserved for use by processors. This area may be generated via the VLP_ATTR macro which is described later in this section.

QISS = {YES|NO}. is meaningful only for CREATE opens of comgroups. YES specifies that the station tree is to be the queue tree. NO specifies that the message type tree is to be the queue tree. The default causes this parameter to be ignored.

REASSIGN= {YES|NO} YES requests that the assign/merge record (values from the SET command) is to be remerged into the DCB. Following the merge of the values from the SET command into the DCB, the parameters from the open FPT are merged into the DCB.

The REASSIGN parameter permits the user to return a DCB to its original state at the start of the program, by opening with SCRUB=YES and REASSIGN=YES. The REASSIGN parameter also allows a user to override the usual merge sequence as follows: (1) the user calls M$GETDCB with MERGE=NO to obtain a DCB, (2) next the user calls M$OPEN with SETDCB=YES to initialize but not open the DCB, (3) then the user calls M$OPEN with REASSIGN=YES as the only parameter in the FPT; as a result the values from the SET command are merged into the DCB after the program-specified OPEN values, instead of before them. The default is NO.

NOTE: For DCBs which were present at program invocation or were obtained via M$GETDCB with the MERGE option, the SET command values are already in the DCB unless it has been altered by M$OPEN.

RECFIELD = VARIABLE Locates an area containing the record field definitions. This area may be generated via the VLP_RECFIELD macro which is described later in this section. Meaningful on create opens only. NOTE: The RECFIELD option is not available in C00.

RECL = VALUE-DEC(1-32K). For fixed record length formats such as RELATIVE and FIXED, RECL is the maximum data record length in bytes. For VARIABLE formats, RECL is the maximum record segment length in bytes. For unspanned files, at most one record segment is used per record, thus limiting each record to RECL bytes which includes 4 bytes of record information. For spanned files, RECL is the maximum length of the data record not including record information bytes. RECL may be zero for spanned files indicating that there is no maximum.

RECL is determined for IN or UPDATE opens from the file opened except for managed tape files for which RECL must be specified.

The default causes this parameter to be ignored.

RES = VALUE-CHAR(4). Specifies a device ('dvnn' or 'dv'). dv is the 2-character device mnemonic or special name (e.g., MT for tape, special name ME). nn is a 2-digit number identifying a device allocated via the RESOURCE command, a reserved logical device name (such as LP01 or CP01), or a logical device name defined via the LDEV command or M$LDEV service. FE[nn] specifies opening of either a real device handler or a FEP-user debugging path. nn, if specified, is the FEP number.

The reserved device types and special names are as follows:

7T[nn]	FT[nn]	LT[nn]
CR01	JE	ME
CG	JF	MT[nn]
CP[nn]	LP01	NO
CR[nn]	LO	CP01
DP[nn]	LP[nn]	UC[nn]
FE[nn]		

(See the Files, Devices, and Comgroups section in the CP-6 Programmer Reference Manual for detailed information.) When RES is a 2-character device mnemonic, see the WSN parameter which influences the disposition of output to the device. For the default, see M$OPEN, the DCB parameter.

SCRUB= {YES|NO} YES specifies that the DCB is to be completely erased and then reinitialized with default conditions. The scrubbing of the DCB occurs prior to the remerge of values from the SET command (if REASSIGN=YES) and prior to the merge of parameters from this open FPT. The default is NO.

SEQ = {YES|NO}. YES specifies that sequencing is to occur on each output record. The default causes this parameter to be ignored.

SEQCOL = VALUE-DEC(1-255). Specifies the column number in which sequencing is to be performed. The sequence ID is printed followed by the record sequence number. Sequencing is controlled by the SEQ = YES parameter. The default causes this parameter to be ignored.

SEQID = VALUE-CHAR(4). Specifies a 4 character sequence identification which is to be appended to each output record when record sequencing has been requested. The default causes this parameter to be ignored.

SETDCB = {YES|NO} YES specifies that the M$OPEN process is to merge the parameters of this open FPT into the DCB (including the SCRUB or REASSIGN operations, if requested). However, the DCB is not to be opened. Only those options that can be stored in a closed DCB can be successfully placed there via the SETDCB option. Such options consist of those that are COMMON to the M$DCB and FPT_OPEN macros. The default for the SETDCB option is NO.

SETSTA = VARIABLE. Locates a VLP_SETSTA that sets the STATION defaults for reads and writes through this COMGROUP DCB. This DCB's name on the COMGROUP may also be specified in the VLP_SETSTA.

SHARE = OPTION. Is meaningful for disk files and comgroups only, and is applicable for disk files only if ORG is not CONSEC or UR. SHARE restricts concurrent usage of a file by multiple DCBs. The SHARE options are: NONE, IN, and ALL. File management allows a DCB to be opened to a file only after examining the FUN, EXIST, and SHARE parameters specified for this user and the FUN and SHARE options in effect for any other DCBs currently open to the file. Table 3-2 shows the conditions under which a DCB is permitted to open the file that is to be shared with other DCBs. If parameters on this open call are incompatible with current usage of the file, the alternate return is taken with a 'file busy' indication given. If this open succeeds, the FUN and SHARE parameters on this open may alter or set the current usage status of the file. (See Table 3-2 for complete information on this topic.)

The default causes this parameter to be ignored.

Normally, file management will be able to maintain the user's current position when SHARE=ALL has been specified, and updates are being made to the file. The exception to this is with alternate indexes in Indexed and IREL files. In order to access a record on an alternate index when SHARE=ALL has been specified, a key MUST be specified.

Table 3-2. Conditions for Shared Use of Disk Files

Intent of Current M$OPEN	Parameters on current M$OPEN:	Current Usage (1) of the file
Replace existing file	FUN=CREATE, DISP=NAMED EXIST=NEWFILE SHARE=n/a	FUN=IN SHARE=NONE or IN
Open existing file for input	FUN=IN SHARE=NONE or IN	FUN=IN SHARE=NONE or IN
	FUN=IN SHARE=ALL	FUN=IN SHARE=NONE, IN or ALL
		FUN=UPDATE SHARE=IN or ALL (2)
Update a file	FUN=UPDATE (3) SHARE=NONE	(no other DCBs may be open to the file)
	FUN=UPDATE (3) SHARE=IN	FUN=IN (4) SHARE=ALL (4,2)
	FUN=UPDATE (3) SHARE=ALL	FUN=IN (5) SHARE=ALL (2)
	FUN=UPDATE (3) SHARE=ALL	FUN=UPDATE SHARE=ALL (2)

1. The current usage status is initialized from the FUN and SHARE options of the first opener of the file.
2. The current usage status of a consecutive or unit-record file is forced to SHARE=NONE. A consecutive file may be used as a common journal, however (see ACS=JRNL).
3. The options FUN=CREATE, EXIST=OLDFILE, DISP=NAMED are equivalent to FUN=UPDATE for purposes of determining if file sharing is allowed.
4. The Current Usage status of the file (FUN=IN, SHARE=ALL) is changed to FUN=UPDATE and SHARE=IN. Following the file close by the DCB which opened to update the file, current usage status reverts to FUN=IN and SHARE=ALL.
5. The Current Usage status of the file (FUN=IN) is changed to FUN=UPDATE and thereafter no other DCBs may open the file with SHARE=IN until all DCBs currently open to the file are closed.

SN= VARIABLE Locates an area containing a list of tape serial numbers or a pack set name. This area may be generated via the VLP_SN macro which is described later in this section.

For tapes, this list must be in the same order in which the volume set was created and cannot contain duplicate or blank serial numbers. Blank serial numbers occurring at the end of the list will be counted in MAXVOL. To conform to ANSI standards, each serial number must be composed of characters from the center four columns of the code table specified in ANSI X3.4-1968 except for position 5/15 and those positions where there is a provision for alternative graphic representation.

SPANNED = {YES|NO}. YES specifies whether logical records may be divided between physical tape records. Spanned records may exceed the record limit imposed by RECL since RECL applies to record segment size. SPANNED applies only to VARIABLE labeled and managed tape files. SPANNED is determined for IN or UPDATE opens from the tape file opened. Spanned and unspanned files may be freely intermixed on a volume. SPANNED should be used in conjunction with BLOCKED for optimum tape record utilization. (SPANNED must be specified for INPUT or UPDATE managed tape files.)

CP-6 format tape files are always spanned.

The default causes this parameter to be ignored.

SPARE = VALUE-DEC(1-511). Specifies in words the amount of spare space to be left unused at the end of each index block while a KEYED, INDEXED, or IREL disk file is being created or updated with sequential access. The value specified may not exceed 511 words. If it does, it is treated modulo 512. This spare space is reserved so that additional keys can be inserted in a minimum time when updating the file with direct access (as in EDIT). If the file will never be updated with direct access, a spare value of 1 should be specified. The default causes this parameter to be ignored.

SRCHCOND. Specifies a mask of bits that are used to select what files are to be found when NXTF=YES. The user sets mask bits via the keywords listed below. For example, specifying ANYF=YES sets the SRCHCOND.ANYF field in the FPT. If SRCHCOND is zero, NXTF returns the next existing file. If SRCHCOND is non-zero, the next file with any of the selected descriptor bits set is found (the descriptor bits are stored in the DESC portion of the DCB). The default is zero.

SRCHCOND.ANYF={YES|NO} When NXTF=YES and ANYF=YES, the open process stops on every entry in the file directory, even deleted entries.

SRCHCOND.ARCHIVE= {YES|NO} When NXTF=YES and ARCHIVE=YES the open process searches for a file that is known to the archive system.

SRCHCOND.BAD= {YES|NO} When NXTF=YES and BAD=YES, the open process searches for a file with a file inconsistency.

SRCHCOND.BUF= {YES|NO} When NXTF=YES and BUF=YES, the open process searches for a file that is to be backed up.

SRCHCOND.DELF= {YES|NO} When NXTF=YES and DELF=YES, the open process searches for a deleted file.

SRCHCOND.NOLIST= {YES|NO} When NXTF=YES and NOLIST=YES, the open process searches for a file with the 'no list' attribute.

SRCHCOND.QS= {YES|NO} When NXTF=YES and QS=YES, the open process searches for a file that is queued for the EFT ARChive or RETrieve process.

SRCHCOND.REV= {YES|NO} When NXTF or NXTA=YES and REV=YES, the open process searches the file or account directory in reverse order.

TAB = VARIABLE Locates an area containing a horizontal tabulation definition. This area may be generated via the VLP_TAB macro which is described later in this section.

TERMCONLGL = {YES|NO} is meaningful only if ORG = CG. Specifies whether terminals are allowed to open to this comgroup. The default causes this parameter to be ignored. See VLP_CGCP for additional information.

TEST = {YES|NO}. Is meaningful for disk and labeled tape files only. TEST=YES specifies that this is a test operation: the DCB is not opened and subsequently does not require a close. TEST=YES is normally used in conjunction with the NXTA, NXTF, and FPARAM parameters to obtain information regarding files via the DCB.

If FPARAM is specified or the function is not IN, security checks are made and the alternate return is taken if the user does not have access to the file. If FPARAM is not specified, the FIT is not accessed, returning only the filename and directory descriptors in the DCB.

Under no circumstances will the error 'file busy' be returned when TEST=YES.

The default is NO.

THISF = {YES|NO} is meaningful only in conjunction with NXTF. THISF=YES specifies that the search for the next file is to start with the file named in the DCB, not the file following the one named. For tapes, the file may be specified by name or by file sequence number (FSN).

TYPE = VALUE-CHAR(2). Is the processor specified file type supplied for convenience and is not used by file management. This 2-character field may be supplied by users or system processors when creating files.

```
   1st Character:
      D = Data
      I = Database
      O = Object Unit
      R = Run Unit
      S = Source
      U = Update
      W = Workspace
      X = Work file
      Y = reserved
      1 = schema for databases
      2 = subschema for databases
      * = file reserved for system use: can only be created by
          operating system; can be read or deleted by user.
      blank = undefined
   2nd character for processors:
      A = APL
      a = ARES
      B = BASIC
      C = COBOL
      D = TP (TRADER)
      E = EDIT
      F = FORTRAN
      f = FPL
      G = GMAP
      I = IDS
      J = IMP
      K = reserved
      P = Performance Monitor or PARTRGE
      Q = IDP
      R = RPG
      S = SORT
      T = TEXT
      1 = reserved
      6 = PL-6
   2nd character for data:
      A = ASCII
      B = ASCII and single precision
      D = Double precision
      S = Single precision
      a = APL data block attributes
      c = APL component file
      blank = undefined or unformatted
```

The default causes this parameter to be ignored.

UATTR= VARIABLE Locates an area containing user attributes. Although the CP-6 system places no restrictions on setting this data, this parameter is available for user processors. This area may be generated via the VLP_ATTR macro which is described later in this section.

UHL = VARIABLE Locates the VLP_ULBL into which User Header Labels (UHLs) are to be read during an IN or UPDATE open of a labeled tape file or from which UHLs are to be written during a CREATE open. User header labels are 80 bytes each and must begin with 'UHL'. The number of labels read is returned in VLP_ULBL.NUM#.

The user is responsible for determining the proper label number (fourth byte) of each written UHL. To conform to ANSI standards, the label number and contents can be any character from the center four columns of the code table specified in ANSI X3.4-1968 except for position 5/15 and those positions where there is a provision for alternative graphic representation.

UOPT0 - UOPT8 = {YES|NO}. Specify flags controllable by !SET, M$DCB, and M$OPEN for use by programs. These flags are ignored by the monitor and are available to pass information to a DCB. The default causes this parameter to be ignored.

VIRTUAL = VARIABLE. Locates an area containing a VLP_VIRTUAL which describes the parameters for a LARGE VIRTUAL segment. Its presence during the open of a KEYED file invokes the large virtual segment mechanism for access and modification of the data in the file.

VOL = VALUE-DEC(1-511). On open, VOL specifies which volume of the volume set (as specified by the serial number list) is to be initially mounted. A value of 0 or 1 indicates the first volume, 2 the second, etc. VOL is used in conjunction with XTEND to determine which volume of the volume set to extend. VOL indicates which volume is currently mounted if the DCB is open. The default causes this parameter to be ignored.

VOLACCESS = OPTION. Indicates labeled tape volume set access limitations.

NONE specifies that only the owner of the volume set (account of user creating first file on the volume set) will be allowed access to this volume set.

IN specifies that all may read the volume set but only the owner may write on it.

ALL specifies that all may read or write the volume set.

The default causes this parameter to be ignored.

WINDOW = VARIABLE Locates an area containing the description of the window to use. The area may be generated by the VLP_WINDOW macro.

WSN = VARIABLE. Specifies a workstation name or the '@' symbol to mean the user's workstation of origin. This parameter is meaningful in combination with the RES parameter when RES is set to a 2-character device mnemonic. When WSN is specified, output is sent when the associated DCB is closed or at job step (rather than at end-of-job, as happens if WSN is blank). Also, when WSN is specified, output through multiple DCBs open to the same RES is kept separate (instead of being merged, as occurs if WSN is blank).

NOTE: WSN must be blank if RES is to specify a special name, a logical device defined by LDEV or M$LDEV, a device allocated by the RESOURCE command, or the reserved logical device name LP01 or CP01.

WSN locates an area containing a workstation name. This area may be generated via the VLP_WSN macro. The default causes this parameter to be ignored.

XONLY = {YES|NO}. Specifies that M$DCB.FFLG.EXEC# is to be set unconditionally once the DCB is open. This will prevent another program (associated in a different domain) from using or closing the DCB. The default is NO.

XTEND = {YES|NO}. For CREATE opens of labeled tape files YES causes the volume set to be positioned after the last file if VOL = 0. If VOL is non-zero, the volume selected is positioned after its last file. XTEND = NO causes the next file to be created at current volume position. The default causes this parameter to be ignored.

XTNSIZE = VALUE-DEC(1-2**17) For FUN=CREATE disk files, size in granules of the secondary extents (used to add on extents). Not applicable to RELATIVE, RANDOM or IDS organizations. The default causes this parameter to be ignored.

M$CLOSE - Close DCB

The M$CLOSE service terminates and inhibits I/O through a specified DCB, until the DCB is again opened. In addition, unique positioning and updating operations occur for the following devices and files.

For tape updating, the M$CLOSE service performs these additional functions:

- Performs end-of-file processing appropriate to tapes:

  For free or managed tape, if the last operation performed was a write, two file marks are written and the tape positioned between them. If the last operation was a write-end-of-file, one file mark is written and the tape positioned before it.

  For labeled tape User Trailing Labels may be specified to be written following the End Of File label group for CREATEd files. A buffer may be specified for receipt of any UTLs present for IN or UPDATE file if the file was not modified. Tape position is left following the file mark which follows the end of file label group (including UTLs).

- Positions the tape following end-of-file processing, if the POS parameter is specified.

For disk the M$CLOSE service performs these additional functions:

- Handles file disposition, which may include releasing a file or entering it in the account file directory. The disposition of a file is determined in part by parameters specified when the DCB was opened. If a file was opened with DISP=SCRATCH for any function, the close process always releases the secondary storage allocated to the file. If a file was opened with DISP=NAMED, the close process disposes of the file as specified by the user in the DISP parameter of the M$CLOSE service.

- Permits the user to modify file attributes (see CHGATTR).

- Permits the user to release unused granules (see RELG).

- Allows the user to request rebuilding of the upper level index for a keyed, indexed or IREL file (see REBLD).

The form of the call for this service is as follows:

CALL M$CLOSE (FPT_CLOSE) [ALTRET (label)];

The parameters for the service are as follows:

ACCESS = VARIABLE Locates an access control list area (see the VLP_ACCESS macro described in this section). If CHGATT=YES and the user has the required permission, the specified list becomes the new access control list for the file.

ACCT = VARIABLE Locates an area containing an account name (see the VLP_ACCT macro which is described later in this section). If CHGATT=YES and the user has the required permission, the specified account becomes the new account in which the file resides, provided the account already exists on the same packset as the old account. This facility is not yet implemented.

ACSVEH = VARIABLE Locates an access vehicle control list (see the VLP_ASCVEH macro described in this section). If CHGATT=YES and the user has the required permission, the specified list becomes the new access vehicle control list for the file.

ALTBLD = {YES|NO} YES specifies that the alternate indices for this indexed or IREL file are to be rebuilt. NO specifies that they are not to be rebuilt. The default is to rebuild them only if necessary.

In order to update the alternate indices on the close, the monitor initiates a load and link to ALTKEY_MON.:SYS. This processor uses X$SORT to sort the alternate indices into their proper order. The new keys are then written to the file. If any errors are detected during the process, an error is returned to the user. Under some circumstances, errors may be written directly to the user through the M$DO DCB.

ALTKEYS = VARIABLE Locates a VLP_ALTKEYS which describes the alternate indices to replace the existing alternate indices in the indexed file, if CHGATT=YES, and the user has the required permission. The alternate keys may only be redefined if the last key in the new definition does not extend beyond the last key in the old definition. The default causes this parameter to be ignored.

BUPM = {YES|NO} Specifies a change in BUPM if CHGATT is YES. See DESC.BUPM in M$DCB.

CHGATT = {YES|NO} YES requests modification of one or more of the following file attributes: BUPM, TYPE, XTNSIZE, EXPIRE, LSLIDE, LRDL0, SPARE, NAME, PASS, ACCESS, ASCVEH, UATTR, INSTATTR, PROCATTR, ALTKEYS, and IRKEYS. The modifications are permitted if the file has been opened specifying FUN=UPDATE, or FUN=CREATE, and the user has REATTR permission, except for NAME, PASS, ACCESS and ACSVEH, which require DELF permission.

DCB = DCBNAME Specifies the DCB to be closed.

DISP = {SAVE|RELEASE} Applies to disk and tape files created with DISP=NAMED. On a close, DISP specifies whether the NAMED file is to be saved or released. (At open the DISP parameter specifies NAMED if the user intends to retain the file and SCRATCH if the user does not intend to retain the file. Any file opened with SCRATCH disposition is always released at close. Thus the DISP parameter on the close is not meaningful for scratch files.)

When DISP=SAVE the file is cataloged if it is not already represented in the file directory. If the file replaces an existing file (the file was opened with CTG=NO and EXIST=NEWFILE), the new file replaces the old file which is deleted from the file directory. The old file is released when all current file users close the associated DCBs.

When DISP=RELEASE, the file is deleted from the file directory (if it was previously cataloged). The file is released when all current file users close the associated DCBs.

The default is SAVE for cataloged files and RELEASE for uncataloged files. (Any file declared as a SCRATCH file is always released.)

EXPIRE = {ddd|NEVER} Specifies a change in EXPIRE if CHGATT is YES. See EXPIRE in M$OPEN.

GHSTACS = VALUE-DEC(0-15). Specifies a change in GHSTACS if nonzero and CHGATT is yes. See GHSTACS in M$OPEN. The value NONE specifies that GHSTACS is to be reset. If specified for a COMGROUP, the change does not have any effect until the COMGROUP is reinitialized (opened with no current users).

IFPARAM = VARIABLE Locates an area containing FIT data which may be used in conjunction with CHGATT=YES to alter various portions of a file's FIT (see CHGATT for a list of the alterable ones).

INSTATTR = VARIABLE Locates an area containing installation attributes (see the VLP_ATTR macro described in this section). If CHGATT=YES and the user has the required permission, the specified installation attributes become the new installation attributes for the file.

IRKEYS = VARIABLE Locates a VLP_IRKEYS which describes the alternate indices to replace the current definitions, if CHGATT=YES, and the user has the required permission. The user may redefine the keys only under special circumstances. First of all, the primary key in the new definition must match the existing primary key definition. Also, the last key in the new definition must not extend beyond the last key in the current definition. This can only be used for IREL files.

LRDL0 = VALUE-DEC(1-511). Specifies a change in LRDL0 if CHGATT is YES. See LRDL0 in M$OPEN.

LSLIDE = VALUE-DEC(1-511). Specifies a change in LSLIDE if CHGATT is YES. See LSLIDE in M$OPEN.

NAME = VARIABLE Locates an area containing a file name (see the VLP_NAME macro described later in this section). If CHGATT=YES and the user has the required permission, the specified name becomes the new name for the file.

NOMARKS = {YES|NO} For free tapes only. YES specifies that file marks are not to be written on a free tape when it is closed after having been modified. Normally, two file marks are written to indicate end of tape. YES specifies if tape is not to be positioned after the next file mark when it is closed after not having been modified. Normally, the tape is positioned to the beginning of the next free tape file. The default is NO.

PASS = VARIABLE Locates an area containing a password (see the VLP_PASS macro which is described later in this section). If CHGATT=YES and the user has the required permission, the specified password becomes the new password for the file.

POS = OPTION Specifies positioning for magnetic tape. IF POS is not specified, the default close positioning is retained with one exception. If DISP=RELEASE is specified for a CREATEd TAPE file, POS is assumed to be PTL.

PTL: For labeled tapes, the volume (set) is positioned before the Header label group preceding the file just closed. For free tapes, the volume is left after the file mark preceding the data section just closed. A device data section includes the file mark immediately following.

PTV: For labeled, managed, and free tapes, the tape is rewound.

REM: For labeled, managed, and free tapes, the current volume is unloaded. If the volume and drive were reserved with a RESOURCE command, the drive remains allocated to the user.

REL: For labeled, managed, and free tapes, the current volume is unloaded and the drive is released to the system.

The default for labeled tapes is to position following the file mark following the EOF label group. The default for free and managed tapes is to leave position unchanged unless the file was modified in which case two file marks are written and tape left positioned between them.

PROCATTR = VARIABLE Locates an area containing processor attributes (see the VLP_ATTR macro described in this section). If CHGATT=YES and the user has the required permission, the specified processor attributes become the new processor attributes for the file.

REBLD = {YES|NO} YES requests rebuilding of the upper level index for a keyed, indexed or IREL disk file.

(Re)building an upper-level index structure increases the number of granules in the file, but has no effect upon the actual content of the level-0 index granules. REBLD (re)creates a pyramid structure which File Management can use to locate keys in the level-0 granules without having to search through the level-0 granules in a sequential fashion.

A level-1 structure should not exist if there are 3 or fewer granules at level 0. At an attempt to read a record whose key is not located in whichever level-0 granule is currently in memory (when there is an upper-level index structure), then File Management goes directly to the first granule at the highest level, and searches "downward" toward level-0 to locate the key. If there are 1, 2, or 3 granules located at level 0, then this multi-level search requires (on the average) more disk accesses than would a sequential search at level-0 (which occurs if there is no upper-level index). Creating 4 granules of level-0 index is a "break-even" point at which a multi-level search requires (on the average) just as many reads as a sequential search. File Management will therefore create a level-1 index when the DCB is closed. For files with 5 or more granules of level-0 index, the multi-level index pyramid can save a significant number of disk I/Os.

Instead of using the REBLD option whenever creating (or updating) a keyed file, it is advisable to have File Management (re)build the index pyramid whenever there would be a significant benefit from doing so. File Management normally rebuilds the pyramid whenever there are at least 3 adjacent level-0 index granules which cannot be reached from level-1, or when there are at least 510 level-0 granules anywhere in the file which can't be reached. For a file that is read and updated frequently, and not opened and closed often, then it might be advisable to change these figures. The former (3) can be changed by specifying LRDLO=n, and the latter can be changed by specifying LSLIDE=n. A value of LSLIDE=1 will cause the upper-level pyramid to be rebuilt at M$CLOSE time, if a new level-0 granule has been added anywhere in the file. This will guarantee a high-quality pyramid but incurs the overhead involved in the frequent rebuilding of the pyramid at file-closure time.

RELG = {YES|NO} YES requests release of unused granules allocated for this disk file. RELG=YES is typically specified when the user anticipates little or no updating to a file.

SN = VARIABLE Locates a VLP_SN area. This area will contain the tape serial numbers which were used to create new tape volumes from scratch tapes.

described later in this section for a definition of serial number list.

SPARE = VALUE-DEC(1-511). Specifies a change in SPARE if CHGATT is YES. See SPARE in M$OPEN.

TYPE = VALUE-CHAR(2) is the processor-supplied file type. If CHGATT=YES and the user has the required permission, the type code becomes the new type code for the file. See M$OPEN for a list of the file type codes.

UATTR = VARIABLE Locates an area containing user attributes (see the VLP_ATTR macro described in this section). If CHGATT=YES and the user has the required permission, the specified user attributes become the new user attributes for the file.

UTL = VARIABLE Locates a VLP_ULBL area for User Trailer Labels (UTLs). UTLs are read into the buffer at close of the file if the file was opened with FUN=IN or UPDATE if the file was not modified. UTLs are written from this buffer during the close of a file opened with FUN=CREATE. User labels are 80 bytes each and must begin with 'UTL'. Labels are packed into the buffer contiguously.

XTNSIZE = VALUE-DEC(1-?). Specifies a change in XTNSIZE if CHGATT is YES. See XTNSIZE in M$OPEN.

A listing of additional fields in the V area follows. These fields are not set via parameters for M$CLOSE.

EFTOPER.ASTOW# is used by EFT to indicate that the file close should take the contents of the 0'th entry of the VLP_SN area described by FPT.SN_ along with the file open time and merge them into the area of the file FIT described by the %CODE06 structure.

EFTOPER.BACKUP# is used by EFT to indicate that the file close should take the contents of the 0'th entry of the VLP_SN area described by the FPT.SN_ along with the file open time and merge them into the area of the file FIT described by the %CODE05 structure. See Appendix C of CE75.

EFTOPER.DELFDE# is used by EFT to indicate that the file directory entry describing the file should be removed, compacting the directory, and removing all knowledge of the file.

EFTOPER.ISTOW# is used by EFT to indicate that the file close should result in the granules being released, and a special directory entry created that may later be accessed through an M$OPEN that asks for FPARAM data. The %CODE06 macro may then be used to determine which tape the file resides upon. See Appendix C of CE75. ISTOW and ASTOW must both be used if the file is being stowed inactive the first time.

EFTOPER.UPDDESC# is used by EFT to indicate that the file close should move the descriptor bits from the FPT into the appropriate areas of the file FIT and the file directory.

VLP_VIRTUAL

The VLP_VIRTUAL macro describes the parameters for a large virtual segment when a keyed file is being opened. It contains parameters that influence the use of a large virtual segment.

INITIALIZE = {YES|NO}. YES specifies that any page that has not been accessed before, and does not exist in the paging file, will be initialized to the value specified in INITVALUE. NO specifies that initialization will not be done; the contents of the page is unchanged. The default is NO.

INITVALUE = VALUE-DEC(0-?). Value will be used when INITIALIZE=YES is specified as the value that the page is to be set to on the initial access of the page (if it does not exist in the paging file). The default is 0.

MINPHYS=VALUE-DEC(1-2048). Specifies the minimum number of physical pages backing the virtual area. This parameter determines the least number of pages that are held within memory at any given time. The parameter has a great effect on the access speed of the virtual segment. The default is ten pages.

NOFILE = {YES|NO}. Specifies if the user wants control of the missing page faults or not. If YES is specified, whenever a page needs to be paged in, the user will get an event. The user then can use M$CVM or M$GVP to allocate that page.

PHYSICAL=VALUE-DEC(1-2048). Specifies the number of physical pages backing the virtual area. This parameter determines how many pages are held within memory at any given time. The parameter has a great effect on the access speed of the virtual segment. The default is ten pages.

PTR$=POINTER. is returned on successful open with a pointer to the segment that was used for this large virtual area. It should be used on all references to the area.

SCALE = VALUE-DEC(0-63) See VLP_RECFIELD for more information.

SEGNUM = VALUE. Specifies which of the three large virtual segments is to be used for this file. The values are VS1, VS2, and VS3.

SEGSIZE = VALUE-DEC(1-2097152) specifies in words the size of the large virtual segment. The actual bounding of a virtual segment is done on a page basis so the SEGSIZE will be rounded up to a number of virtual pages. Any page access that is not within bounds will cause a missing page fault.

WSQ - This number is returned to the user containing the WSQ allocated. This is needed in conjunction with the NOFILE option to determine what WSQ to use.

M$CVOL - Close Volume

The M$CVOL service causes the monitor to terminate the reading or writing of data in the magnetic tape reel currently associated with a specific DCB, and to advance to the next reel of the data set.

For free and managed tapes the current reel is closed and the next reel positioned to its beginning.

For input files on labeled tape, the current file section is positioned to its end and determination made (by reading the following label group) of the existence of a subsequent file section. If no such file section exists, M$CVOL will ALTRET with and end-of-file error and leave the current file positioned to its end. If a subsequent file section exists, the current volume is dismounted and the next mounted and positioned to the first record of the next file section. Any UTLs following the EOV label group are returned if UTL is specified. Any UHLs following the HDR label group of the next file section are returned if UHL is specified. Both UTLs and UHLs are returned in VLP_ULBL format.

For output files on labeled and managed tape, the current file section is truncated after the last whole record.

In addition, for output files on labeled tape, an EOV label is written followed by any UTLs specified by UTL. The current volume is dismounted and the next mounted and positioned ready to write the next record of the file (first of the next file section) after writing necessary VOL1, HDR and UHL labels as required or specified by UHL.

Note: On input, UTLs and UHLs are returned to their separate buffers contiguously packed. The first word of the user label areas will contain the number of labels of that type returned. On output, UTLs and UHLs are written from the user's label buffers which are contiguously packed. The first word of the user label areas will contain the number of labels of that type to write. Any labels not containing the proper first three characters will cause termination of user label writing.

CVOL control should only be used with unspanned tape files since spanned tape files may have records which cross volume boundaries. These records are difficult to process if the user takes CVOL control.

The form of the call for this service is as follows:

CALL M$CVOL (FPT_CVOL) [ALTRET (label)];

The parameters for this service are as follows:

DCB = DCBNAME Specifies the DCB associated with the file.

UHL = VARIABLE Locates the VLP_ULBL into which User Header Labels (UHLs) are to be read during an IN or UPDATE open of a labeled tape file or from which UHLs are to be written during a CREATE open. User header labels are 80 bytes each and must begin with 'UHL'. The number of labels read is returned in VLP_ULBL.NUM#.

The user is responsible for determining the proper label number (fourth byte) of each written UHL. To conform to ANSI standards, the label number and contents can be any character from the center four columns of the code table specified in ANSI X3.4-1968 except for position 5/15 and those positions where there is a provision for alternative graphic representation.

UTL = VARIABLE Locates a VLP_ULBL area for User Trailer Labels (UTLs). UTLs are read into the buffer at close of the file if the file was opened with FUN=IN or UPDATE if the file was not modified. UTLs are written from this buffer during the close of a file opened with FUN=CREATE. User labels are 80 bytes each and must begin with 'UTL'. Labels are packed into the buffer contiguously.

VLP_NAME

The VLP_NAME macro may be used to generate file names (or any other entity that is in TEXTC format, such as a key for a keyed file). It generates a length byte and a character string of fixed length. See also VLP_NAMEV.

L = VALUE-DEC(0-511). Specifies the number of significant characters in the name. It defaults to the length of NAME, or to 31 if NAME is not specified.

LEN = VALUE-DEC(0-511). Specifies the number of bytes to be provided for the name.

NAME = VALUE-CHAR. Specifies a file name in TEXTC format. The length of the CHAR field generated is the length of NAME (31 if NAME is not specified). The LEN parameter can be specified to override the length (e.g. LEN=5). See M$OPEN for the character set for file names. If a non-initializing STCLASS is used, the default for LEN causes generation of a variable-length character string whose length is in the length byte (VLP_NAME.L#). If STCLASS=AUTO is used, LEN must be specified. See also VLP_NAMEV for a different type of based structure.

VLP_NAMEV

The VLP_NAMEV macro generates a based version of the VLP_NAME macro with the length of the character string being the value currently in the length byte.

VLP_ACCT

The VLP_ACCT macro may be used to generate an account name.

ACCT = VALUE-CHAR(8) Specifies an account name which specifies the file directory in which the file is to be cataloged. See M$OPEN for the character set for account names. The VLP_ACCT macro should be used to generate the area which is referenced in various file management services. The default is blanks.

VLP_PASS

The VLP_PASS macro may be used to generate a password.

PASS = VALUE-CHAR(8) Specifies a password. See M$DCB for the character set for passwords. The VLP_PASS macro should be used to generate the area containing the password for the various file management services.

The default is blanks, which is equivalent to "no password".

VLP_ATTR

The VLP_ATTR macro may be used in conjunction with such keywords as ACCESS, ASCVEH, PROCATTR, INSTATTR, and UATTR. The VLP_ATTR macro generates an attribute area or the head for a list. The VLP_ATTR macro has three parameters, the name (FPTN) for the structure and the storage class(STCLASS) and the number of words to reserve in the FIT of the file for future expansion of the corresponding attribute area (NAW). Specification of NAW is not required for such expansion, but it does insure that it will be possible.

When creating an attribute area, the user invokes the VLP_ATTR macro to generate the first two levels of the structure, including the size fields and a field called Q. The following sample shows the invocation of the VLP_ATTR macro with the third level of structure which the user supplies, and also the assignment of a value to that field.

Example:

```
%VLP_ATTR (FPTN=VLP_PROCATTR);
3 PROCATTR(0:2) UBIN INIT(5,2,3);

VLP_PROCATTR.Q.PROCATTR(2)=9;
```

Note: The attributes generated via VLP_ATTR and referenced by INSTATTR, PROCATTR, or UATTR are stored in the file information table (FIT). The combined size of all fields in the FIT must not exceed 1019 words.

When creating an access control list or an access vehicle control list, the user invokes the VLP_ATTR macro to generate the first two levels of the list structure and the list size. See VLP_ACCESS and VLP_ACSVEH for examples of use of the VLP_ATTR macro to generate a list head.

VLP_ACCESS

This macro generates one entry for an access control list. An entry sets the access permissions granted to a specified account. Each permission must be explicitly specified and implies no other permission. For example, the permission to delete records (DELR=YES) does not imply permission to read the file. The user must separately specify read permission by READ=YES. The VLP_ACCESS parameters are listed below.

The access control list consists of a list head (generated via the VLP_ATTR macro) and a series of entries generated by invocations of the VLP_ACCESS macro. The VLP_ACCESS macros must immediately follow the VLP_ATTR macro. (The list size is generated automatically.) The list name is assigned via the VLP_ATTR macro. A name is unnecessary for each entry generated via VLP_ACCESS; thus a default is associated with each entry. To indicate the last entry, the list must contain LAST=";" as the final parameter. The storage class attribute (STCLASS) must be the same for %VLP_ATTR and each %VLP_ACCESS used to form an access control list.

This access control list may be applied to either a file or an account.

EXAMPLE:

```
%VLP_ATTR(FPTN=ACT);
%VLP_ACCESS(READ=YES,UPD=YES,DELF=YES,ACCT='?0781');
%VLP_ACCESS(WNEW=YES,UPD=YES,ACCT='JOHN');
%VLP_ACCESS(NOLIST=YES,ACCT='SAM',LAST=";");
```

ACCT = VALUE-CHAR(8) Specifies the account to which the permissions in VLP_ACCESS.FFLG are to be granted. A wildcard character '?' may be included in the account specification.

AU = {YES|NO} Meaningful for comgroups only. If set, this user is allowed to be the administrative user. This parameter is used to initialize the field VLP_ACCESS.FFLG.AU#. The default is NO.

AURD = {YES|NO} Meaningful for comgroups only. If set, this user is allowed to issue monitor services normally reserved for the administrative user that examine but do not change the comgroup. This parameter is used to initialize the field VLP_ACCESS.FFLG.AURD#. The default is NO.

DELF = {YES|NO} If set, file may be deleted or its name, password, or access control may be changed (via these options on M$CLOSE: DISP=RELEASE, NAME, PASS, ACCESS, or ACSVEH). This parameter is used to initialize the field VLP_ACCESS.FFLG.DELF#. The default is NO.

DELR = {YES|NO} If set, existing records may be deleted. This parameter is used to initialize the field VLP_ACCESS.FFLG.DELR#. The default is NO.

EXEC = {YES|NO} Specifying YES causes a search of any access vehicle list for a match with the running processor. If no match is found, DCB.FFLG is set to VLP_ACCESS.FFLG and DCB.FFLG.EXEC is reset. Otherwise, DCB.FFLG is set to VLP_ACSVEH.FFLG for the matching vehicle and DCB.FFLG.EXEC is set. This parameter is used to initialize the field VLP_ACCESS.FFLG.EXEC#. The default is NO.

NOLIST = {YES|NO} If set, file appears not to exist. This parameter is used to initialize the field VLP_ACCESS.FFLG.NOLIST#. The default is NO.

READ = {YES|NO} If set, records may be read. This parameter is used to initialize the field VLP_ACCESS.FFLG.READ#. The default is NO.

REATTR = {YES|NO) If set, disk file attributes, with the exception of ACCESS and ACSVEH, may be modified at M$CLOSE. For modification of ACCESS and ACSVEH, see FFLG.DELF. This parameter is used to initialize the field VLP_ACCESS.FFLG.REATTR#. The default is NO.

TCTL = {YES|NO} Meaningful for comgroups only. If set, this user is allowed to issue terminal control monitor services (such as M$PROMPT and M$STRMATTR). For ORG=FPRG comgroups, M$SETFP is allowed. This parameter is used to initialize the field VLP_ACCESS.FFLG.TCTL#. The default is NO.

UPD = {YES|NO} If set, existing records may be updated (overwritten). This parameter is used to initialize the field VLP_ACCESS.FFLG.UPD#. The default is NO.

WNEW = {YES|NO} If set, new records may be written. This parameter is used to initialize the field VLP_ACCESS.FFLG.WNEW#. The default is NO.

VLP_ACSVEH

The VLP_ACSVEH macro generates one entry for an access vehicle control list. An entry sets the access permissions granted to a specified processor (that is, a vehicle). Each permission must be explicitly specified and implies no other permission. For example, the permission to delete records (DELR=YES) does not imply permission to read the file. The user must separately specify read permission by READ=YES.

The access vehicle control list consists of a list head (generated via the VLP_ATTR macro) and a series of entries generated by invocations of the VLP_ASCVEH macro. The access vehicle control list is constructed in the same manner as the access control list, using VLP_ACSVEH instead of the VLP_ACCESS macro. The VLP_ASCVEH name defaults to *. The storage class attribute (STCLASS) must be the same for %VLP_ATTR and each %VLP_ACSVEH used to form an access vehicle control list.

EXAMPLE:

```
%VLP_ATTR(FPTN=VEH);
%VLP_ACSVEH(READ=YES,UPD=YES,DELF=YES,VEHICLE='EDIT');
%VLP_ACSVEH(WNEW=YES,UPD=YES,VEHICLE='COBOL');
%VLP_ACSVEH(NOLIST=YES,VEHICLE='PCL',LAST=";");
```

AU = {YES|NO} Meaningful for comgroups only. If set, this user is allowed to be the administrative user. This parameter is used to initialize the field VLP_ACCESS.FFLG.AU#. The default is NO.

AURD = {YES|NO} Meaningful for comgroups only. If set, this user is allowed to issue monitor services normally reserved for the administrative user that examine but do not change the comgroup. This parameter is used to initialize the field VLP_ACCESS.FFLG.AURD#. The default is NO.

DELF = {YES|NO} If set, file may be deleted or its name, password, or access control may be changed (via these options on M$CLOSE: DISP=RELEASE, NAME, PASS, ACCESS, or ACSVEH). This parameter is used to initialize the field VLP_ACSVEH.FFLG.DELF#. The default is NO.

DELR = {YES|NO} If set, existing records may be deleted. This parameter is used to initialize the field VLP_ACSVEH.FFLG.DELR#. The default is NO.

LEN = VALUE-DEC(0-31) Specifies the size of the field to be allocated for VEHICLE. The default is 0, which causes a list of initialized VLP_ACSVEH structures to function properly, since PL6 interprets 0 in this context as LENGTHC(VEHICLE). This option is thus most useful when STCLASS is SYMREF or BASED, where a specification of "...VLP_ACSVEH.C" would accurately describe the field.

READ = {YES|NO} If set, records may be read. This parameter is used to initialize the field VLP_ACSVEH.FFLG.READ#. The default is NO.

REATTR = {YES|NO} If set, disk file attributes, with the exception of ACCESS and ACSVEH, may be modified at M$CLOSE. For modification of ACCESS and ACSVEH, see FFLG.DELF. This parameter is used to initialize the field VLP_ACSVEH.FFLG.REATTR#. The default is NO.

TCTL = {YES|NO} Meaningful for comgroups only. If set, this user is allowed to issue terminal control monitor services (such as M$PROMPT and M$STRMATTR). For ORG=FPRG comgroups, M$SETFP is allowed. This parameter is used to initialize the field VLP_ACCESS.FFLG.TCTL#. The default is NO.

UPD = {YES/NO} If set, existing records may be updated (overwritten). This parameter is used to initialize the field VLP_ASCVEH.FFLG.UPD#. The default is NO.

VEHICLE = VALUE-CHAR(31) Specifies the vehicle to which the permissions in VLP_ACSVEH.FFLG are to be granted. This field is TEXTC, with the count byte at VLP_ACSVEH.C.

WNEW = {YES|NO} If set, new records may be written. This parameter is used to initialize the field VLP_ACSVEH.FFLG.WNEW#. The default is NO.

VLP_ALTKEYS

The VLP_ALTKEYS macro creates an area describing alternate index keys for indexed files. It is used on create opens or on closes.

The VLP_ALTKEYS area includes the field VLP_ALTKEYS.SIZ# followed by an array, K. K is an array of dimension (0:N-1), with one entry for each alternate index. For create opens SIZ# contains the number of indices that are being created at this time, while N is the number to leave room for in the File Information Table (FIT).

KEYL = VALUE-DEC(1-255) Specifies the length of the key for this index. The default is one.

KEYX = VALUE-DEC(0-4000) Specifies the byte offset into the record of the beginning of this key. The default is zero, which indicates the first column in the record.

N = VALUE-DEC(0-80) specifies the number of alternate indices to leave room for in the FIT of the file. This value is used to initialize VLP_ALTKEYS.SIZ#, but the SIZ# field may be changed at create open to establish the number of alternate indices to be created.

UNIQUE = {YES|NO} if YES, then the keys in this index must be unique. If a record is written which contains a key with the same value as an existing key, then the write will fail. If UNIQUE = NO, then duplicate entries will be created if several records have the same key value. The order of the duplicate keys is the order of arrival of the records.

VLP_ID

The VLP_ID macro creates an area to be used to either specify or return record identifiers, and to either specify or return presence bits.

A record identifier is used to uniquely identify a single record within a set of duplicate keys along an alternate index. A record identifier is specified to file management as a double-word. The first word is the count of the number of identifiers. This field is ignored when specified to file management. It is only used in conjunction with the IDALL option on M$PRECORD. The second word is a unique record identifier that may be obtained from file management through M$READ, M$WRITE, or M$PRECORD.

VLP_ID may be used to specify or obtain presence bits. Presence bits indicate whether or not fields are present. The bits correspond directly with the VLP_RECFIELD definition, e.g. the first bit indicates whether or not the first field is present, etc. If presence bits are not specified on an M$WRITE, then all ones are assumed.

The first two words of VLP_ID are always reserved for specifying or returning IDs. The space following that is assumed to be for presence bits.

If a field used to sort a key is not present, then the data in that field is assumed to be invalid. A null value is used to sort the field. Null values sort before any user specifiable values. Presence bits do not affect the contents of a record. Records are returned on a read exactly as they were specified on the write.

Presence bits may also be used for M$READs and M$PRECORDs. If presence bits and a key are specified on an M$READ or M$PRECORD, then it is possible to locate records containing null values. The presence bits are used only to indicate which fields in the key presented are null. With the exception of the fields corresponding to the key, the presence bits specified do not have to match the presence bits for the record returned.

Presence bits may also be returned by using the PBR option on M$READ and M$PRECORD.

COUNT = VALUE-DEC(0-n) specifies the number of record identifiers. This value should always be 1 when IDS=YES.

ID = VALUE-DEC(0-n) specifies the record identifier to search for.

N = VALUE-DEC(0-n) specifies the number of presence bits.

PBITS = VALUE-BIT(1) specifies the presence bit(s).

VLR_ID

The VLR_ID macro is used to generate an area to describe the values returned by the IDALL option for M$PRECORD.

COUNT = VALUE-DEC(0-n). Specifies the number of record identifiers that exist for the specified key.

ID = ARRAY VALUE-DEC(0-n). Is the array of record identifiers that are returned by file management. The size of the array is specified by N.

N = VALUE-DEC(0-n) specifies the amount of space to allocate for returning record identifiers.

VLP_WSN

This macro generates an area containing a workstation name. The workstation name is meaningful in conjunction with a device fid or special name (RES on M$DCB or M$OPEN, DEV on M$LDEV). The workstation specifies a cluster of unit record devices; the device type selects those devices in the workstation of that type.

A symbol ('@') can be substituted for the workstation name to mean the user's workstation of origin. The definition of workstation names and assignment of devices to workstations are made by the installation manager.

WSN = VALUE-CHAR(8) Contains the workstation name which is referenced by WSN in various file management services.

VLP_SN

The VLP_SN macro creates an area containing the list of serial numbers of the volumes or tape reels that are to be accessed; or the pack set name for pack set access. The serial numbers and pack set names may consist of up to 6 alphanumeric characters. The pack set names are defined by the installation manager.

The following examples demonstrate the use of this macro.

Example for disk:

```
%VLP_SN (FPTN=PACK_SET,N=1,SN='PAC   ');
```

Example for tape:

```
%VLP_SN (FPTN=TAPE_SET,N=2,SN="'100001','100002'");
```

N = VALUE-DEC(1-?) Specifies the number of serial numbers which have been defined within this structure. The default for N is one.

SN = 1 to n VALUES-CHAR(6) Specifies the serial numbers of the volumes contained in the tape volume set. They must be given in the order that the volume set was created or is to be created.

VLP_ULBL

The VLP_ULBL macro creates an area containing a user label group for input or output. For output, N should be the number of user labels to write. On input, N is set to the number of user labels read; the size of VLP_ULBL determines the number of labels read.

N = VALUE-DEC(1-?) Specifies the number of user labels which are to be written. The default for N is zero.

ULBL = "'UxL#1[,UxL#2,...,UxL#N]'" specifies if x is 'H' the User Header Labels to be written during a M$OPEN or M$CVOL or if x is 'T' the User Trailer Labels to be written during a M$CLOSE or M$CVOL. The structure may be used to receive user labels on input.

VLP_FORM

The VLP_FORM macro generates an area containing the name of the form to be mounted on a unit record device. The form name is defined in the Form Definition File (created by the installation manager via Super). For devices which have pagination, the definition specifies paper width, printable length, and vertical format control information. For other unit record devices, the definition specifies device width.

The form name is meaningful for opens to UR disk files and to devices. The form name is also meaningful for calls to the M$LDEV monitor service.

FORM = VALUE-CHAR(6) Specifies the name of the forms to be mounted on a unit record device. The default is blanks. Blanks mean use standard forms for the device.

VLP_TAB

The VLP_TAB macro generates an area containing horizontal tabulation definition. Tabulation is only applicable on output to devices and Unit Record (UR) files.

MARGIN = VALUE-DEC(0-255) specifies the lefthand margin by column number. The leftmost column is considered to be column 1. Record from the user's M$WRITE request will have blanks inserted so the the user's text starts in the specified column. The default (0 or 255) causes this parameter to be ignored.

TABS = 1-31 VALUES-DEC(0-255) Specifies horizontal tabulation columns. There is a maximum of 31 tabulation columns, which must be in ascending numeric order and separated by commas. Records from user M$WRITE requests which have HT (Horizontal Tab) characters in the record will be processed as if each HT were replaced by a string of blanks up to the next-higher horizontal tabulation column.

If MARGIN is also specified, tabulation is displaced so that the MARGIN column on the output medium is TABS column 1. The default is no tabulation. (Each HT character in an output record is replaced by a single blank.)

VLP_HDR

The VLP_HDR macro generates an area containing page header information for a unit record device. The page header information is applicable to unit record files and devices which have pagination.

COUNT = VALUE-DEC(1-255) Specifies that the page number is to be printed on the TITLE line, with its lefthand digit in the specified column. The default is 0 (i.e., no page numbering).

HEADERHEIGHT = VALUE-DEC(0-15) Specifies the height of the page header area. The default is 1 (i.e., single space after printing title line).

INDENT = VALUE-DEC(1-255) Specifies the column in which TITLE text is to start (see TITLE). Lefthand column is column 1, and is the default.

RESETPAGE = {YES|NO} YES specifies page numbering is to be restarted at page 1. NO specifies that no change is to be made. The default is NO.

TITLE = VALUE-CHAR(1-?) Specifies the text which is to be printed as a title on the top of every page of output for the line printer. This field is TEXTC.

VLP_SETSTA

The VLP_SETSTA macro generates an area used to set the station defaults for M$OPENs to comgroups. SETSTA$ in comgroup DCBs points to a VLP_SETSTA that describes the current defaults.

ALLABSENT = {YES|NO} ALLABSENT = YES specifies the default value for ALLABSENT on M$WRITEs only through this DCB (and has no effect on M$DEACTIVATE).

If ALLABSENT is set here, ALLABSENT on the M$WRITE is ignored. (See ALLABSENT in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.OSTA.ALLABSENT#.

The default is NO.

ALLDCBS = {YES|NO} ALLDCBS = YES specifies the default value for ALLDCBS on M$WRITEs only through this DCB (and has no effect on M$ACTIVATE and M$DEACTIVATE).

If ALLDCBS is set here, ALLDCBS on the M$WRITE is ignored. (See ALLDCBS in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.OSTA.ALLDCBS#.

The default is NO.

ALLTRMS = {YES|NO} ALLTRMS = YES specifies the default value for ALLTRMS on M$WRITEs only through this DCB (and has no effect on M$ACTIVATE and M$DEACTIVATE).

If ALLTRMS is set here, ALLTRMS on the M$WRITE is ignored. (See ALLTRMS in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.OSTA.ALLTRMS#.

The default is NO.

ANYDCB = {YES|NO} ANYDCB = YES specifies the default value for ANYDCB on writes through this DCB. If ANYDCB is set here, ANYDCB on the M$WRITE is ignored. (See ANYDCB in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.OSTA.ANYDCB#.

The default is NO.

DIRONLY = {YES|NO} DIRONLY = YES specifies the default value for DIRONLY on reads through this DCB. If DIRONLY is set here, DIRONLY on the M$READ is ignored. (See DIRONLY in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.ISTA.DIRONLY#.

The default is NO.

EOFNONE = {YES|NO} EOFNONE = YES specifies the default value for EOFNONE on reads through this DCB. If EOFNONE is set on the M$READ, EOFNONE is ignored here. EOFNONE overrides EOFTIME in this VLP_SETSTA or on the M$READ. (See EOFNONE in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.ISTA.EOFNONE#.

The default is NO.

EOFONE = {YES|NO} EOFONE = YES specifies that only one message can be read through this DCB. After that message has been successfully read, all subsequent M$READs will return END-OF-FILE. EOFONE overrides EOFNONE in this VLP_SETSTA or on the M$READ. It also overrides EOFTIME in this VLP_SETSTA or on the M$READ.

The VLP field name of this parameter is VLP_SETSTA.ISTA.EOFONE#.

The default is NO.

EOFTIME = VALUE-DEC(0-?) is the number of seconds to leave this read pending before timing it out with END-OF-FILE. If EOFTIME is set on the M$READ, EOFTIME is ignored here. (See EOFTIME in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.ISTA.EOFTIME#.

The default is 0.

ILATCH = {YES|NO} ILATCH = YES specifies the default value for LATCH on reads through this DCB. If ILATCH is set, LATCH on the M$READ is ignored. (See LATCH in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.ISTA.ILATCH#.

The default is NO.

ILOCK = {YES|NO} ILOCK = NO specifies that the values given here for IMSGTYP, ISTATION, DIRONLY, EOFNONE, and ILATCH are simply defaults to be applied when an M$READ fails to specify their analogues. ILOCK = YES specifies that these values are not only defaults but may not be overridden on M$READ.

ILOCK = YES has the following specific effects. If STATION is specified on M$READ then the STATION and MSGTYP therein must either be blank or match ISTATION and IMSGTYP respectively, at least up to the wild-card character (if any) in the latter. Specifying blanks for ISTATION (respectively IMSGTYP) causes no restraint to be placed on STATION (respectively MSGTYP) on M$READ. For the control options (EOFNONE, LATCH, and DIRONLY), the specifications given here are honored instead of those specified on M$READ and furthermore an M$READ is illegal if it attempts to specify YES for a control option specified as NO here.

ILOCK = NO has the following specific effects. If an M$READ supplies no station parameter, then the values specified here are used. If an M$READ specifies blanks for STATION (respectively MSGTYP) then the ISTATION (respectively IMSGTYP) specified here is used. For the control options, the resultant specification is YES if and only if YES is specified either here or on M$READ. In other words, if the control is specified as YES here, it is ignored on M$READ and treated as YES; if the control option is specified as NO here then the value specified on M$READ is honored.

The VLP field name of this parameter is VLP_SETSTA.ISTA.ILOCK#.

The default is NO.

IMSGTYP = VALUE-CHAR(8) specifies the default value for MSGTYP on reads through this DCB. (See MSGTYP in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.ISTA.IMSGTYP#.

ISTA is the area containing the options applicable to M$READ.

ISTATION = VALUE-CHAR(8) specifies the default value for STATION on reads through this DCB. (See STATION in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.ISTA.ISTATION#.

MYSTATION = VALUE-CHAR(8) specifies the station name this DCB will have on the comgroup. If blanks, a unique name is assigned by the system. This name will be different at each open, however. Thus MYSTATION should be specified if directed writes are to be performed to this DCB. The default is blanks.

OLATCH = {YES|NO} OLATCH = YES specifies the default value for LATCH on writes through this DCB. If OLATCH is set, LATCH on the M$WRITE is ignored. (See LATCH in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.OSTA.OLATCH#.

The default is NO.

OLOCK = {YES|NO} OLOCK = NO specifies that the values given here for OMSGTYP, OSTATION, WAS, ALLTRMS, ALLDCBS, ANYDCB, and OLATCH are simply defaults to be applied when an M$WRITE fails to specify their analogues. OLOCK = YES specifies that these values are not only defaults but may not be overridden on M$WRITE.

OLOCK = YES has the following specific effects. If STATION is specified on M$WRITE then the STATION and MSGTYP therein must either be blank or match OSTATION and OMSGTYP respectively, at least up to the wild-card character (if any) in the latter. Specifying blanks for OSTATION (respectively OMSGTYP) causes no restraint to be placed on STATION (respectively MSGTYP) on M$WRITE. For the control options (ANYDCB, LATCH, ALLTRMS, ALLDCBS, and WAS), the specifications given here are honored instead of those specified on M$WRITE and furthermore an M$WRITE is illegal if it attempts to specify YES for a control option specified as NO here.

OLOCK = NO has the following specific effects. If an M$WRITE supplies no station parameter, then the values specified here are used. If an M$WRITE specifies blanks for STATION (respectively MSGTYP) then the OSTATION (respectively OMSGTYP) specified here is used. For the control options, the resultant specification is YES if and only if YES is specified either here or on M$WRITE. In other words, if the control is specified as YES here, it is ignored on M$WRITE and treated as YES; if the control option is specified as NO here then the value specified on M$WRITE is honored.

The VLP field name of this parameter is VLP_SETSTA.OSTA.OLOCK#.

The default is NO.

OMSGTYP = VALUE-CHAR(8) specifies the default value for MSGTYP on writes through this DCB. (See MSGTYP in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.OSTA.OMSGTYP#.

The default is NO.

OSTA is the area containing the options applicable to M$WRITE.

OSTATION = VALUE-CHAR(8) specifies the default value for STATION on writes through this DCB. (See STATION in VLP_STATION.)

The VLP field name of this parameter is VLP_SETSTA.OSTA.OSTATION#.

The default is NO.

WAS = {YES|NO} WAS = YES specified the default value for WAS on writes through this DCB. If WAS is set here, WAS on the M$WRITE is ignored. (See WAS in VLP_STATION).

The VLP field name of this parameter is VLP_SETSTA.OSTA.WAS#.

The default is NO.

VLP_STATION

The VLP_STATION macro generates an area used by reads and writes to comgroups to describe a message being read or written. LASTSTA$ in the DCB also points to a VLP_STATION describing the last message read or written through that DCB.

ALLABSENT = {YES|NO} applies when STATION is wild-carded, and the operation is M$WRITE or M$DEACTIVATE. See ALLDCBS.

The VLP field name of this parameter is VLP_STATION.CTL.ALLABSENT#.

The default is NO.

ALLDCBS = {YES|NO} applies when STATION is wild-carded, and the operation is M$WRITE, M$ACTIVATE, or M$DEACTIVATE.

The parameters ALLTRMS, ALLDCBS, ALLABSENT, and WAS restrict (in various combinations) the set of stations to be affected by a wild-carded M$WRITE, M$ACTIVATE, or M$DEACTIVATE. WAS is meaningful only on M$WRITE. ALLABSENT is meaningful only on M$WRITE and M$DEACTIVATE.

The possible set of stations consists of all stations known to the comgroup whose names qualify under the wild-card given in STATION.

For M$ACTIVATE, the parameters interact as follows. Absent stations are never affected. If ALLDCBS and ALLTRMS are both NO, then at most one station is activated, it being the first station found to qualify under the wildcard. For example, if STATION is 'AB?', then of the stations ABC and ABZ, ABC will be activated, regardless of what types of stations (DCB or TERMINAL) the two happen to be. If, on the other hand, ALLDCBS is YES, then all qualifying DCB stations will be activated, and similarly for all qualifying terminals if ALLTRMS is YES. If both are YES then all qualifying connected stations are activated.

For M$DEACTIVATE, a similar situation exists, except that absent stations are processed as controlled by the ALLABSENT parameter. Specifically, if ALLABSENT, ALLDCBS, and ALLTRMS are all NO, then at most one station is affected (selected as in M$ACTIVATE). Setting any or all of the three parameters to YES causes all qualifying stations of the corresponding type(s) to be processed.

M$WRITE acts like M$DEACTIVATE with a single difference. Absent or deactivated stations will be skipped if WAS is NO. Note that this means that in the case where all of ALLABSENT, ALLDCBS, ALLTRMS, and WAS are NO, no message will be written if the first qualifying station is absent.

The VLP field name of this parameter is VLP_STATION.CTL.ALLDCBS#.

The default is NO.

ALLTRMS = {YES|NO} applies when STATION is wild-carded, and the operation is M$WRITE, M$ACTIVATE, or M$DEACTIVATE. See ALLDCBS.

The VLP field name of this parameter is VLP_STATION.CTL.ALLTRMS#.

The default is NO.

ANYDCB = {YES|NO} ANYDCB = YES specifies that this M$WRITE is directed to any DCB station. The message is treated as though it came in from a terminal (i.e. is placed in the anonymous queue), and STATION is ignored. NO, the default, says to use STATION to direct the write.

The VLP field name of this parameter is VLP_STATION.CTL.ANYDCB#.

DIRONLY = {YES|NO} DIRONLY = YES specifies that an M$READ will only read messages written to this station, not those in the anonymous queue. NO, the default, allows both directed and undirected messages to be read. Note that if the reader is only interested in directed messages, then setting DIRONLY to YES is more efficient (in terms of internal comgroup overhead), as well as safer (in the sense that no undirected messages will be received accidentally).

The VLP field name of this parameter is VLP_STATION.CTL.DIRONLY#.

The default is NO.

EOFNONE = {YES|NO} EOFNONE = YES specifies that if a read cannot be satisfied from data currently in the comgroup (as opposed to messages received later from other stations) END-OF-FILE will be given on the M$READ. NO specifies that the read will remain active until an appropriate message arrives in the comgroup. EOFNONE has no effect on M$WRITE.

The VLP field name of this parameter is VLP_STATION.CTL.EOFNONE#.

The default is NO.

EOFTIME = VALUE-DEC(1-?) is the number of seconds to leave this read pending before timing it out.

The VLP field name of this parameter is VLP_STATION.EOFTIME#.

The default is 0 which means don't time out this read.

LATCH = {YES|NO} LATCH = YES specifies that the message read or written is to be latched. In the case of a read this means that the message must be read fully (no errors) before another message can be read and output messages may be latched to it. On writes, LATCH = YES specifies that the output message is to be latched to the input message last read by the DCB or, in the streamed case, the DCB to which it is streamed. A latched output message is not delivered until it is unlatched, typically when the associated input message is unlatched. (See M$UNLATCH.)

The VLP field name of this parameter is VLP_STATION.CTL.LATCH#.

The default is NO.

MSGID = VALUE-DEC(1-?) is the system-generated ID of a message. It is ignored on M$READ, and M$WRITE. The VLP_STATION pointed to by LASTSTA$ in the DCB contains the parameters of the last message read or written through that DCB; in particular it contains the MSGID of that message. The VLP_STATION which is part of the no-wait I/O parameters passed to a user event procedure contains the parameters of the message whose read/write complete is being reported.

With the exception of M$WRITE with LATCH specified, each message created in a comgroup is assigned a unique MSGID. For a discussion of the latched write case, refer to MSGIDXT.

Since MSGID is never supplied by a user in VLP_STATION, it has no default value and cannot be given one when this macro is invoked. Generation of VLP_STATION in static form results in a MSGID of zero.

MSGIDBXT = VALUE-DEC(0-?) is the alias of MSGIDXT, for use when message ID extensions take their second possible form. See MSGIDXT.

MSGIDXT = VALUE-BIT(36) is the extension to MSGID. Unless the message was generated by M$WRITE with LATCH, MSGIDXT is zero. For messages generated by M$WRITE with LATCH, the MSGID of the latched output is the same as that of the current latched input message, and the MSGIDXT is unique and nonzero. Thus, in general, a comgroup message is guaranteed to have a unique MSGID-MSGIDXT pair, but not necessarily a unique MSGID.

A given comgroup can generate MSGID extensions in one of two forms, selected by the BIGMXT option in VLP_CGCP supplied to M$CGCTL. The difference between the two forms is that the first provides a unqiue 'ancestry trail' detailing the ancestry of the message, but imposes limitations on the number of messages which may be latched to a single input. The second form imposes no such limitation, but provides no ancestry information.

The first form of extension is selected by BIGMXT = NO (the default). In this form, the extension consists of six fields or levels. When a message is written with LATCH, its MSGID is the same as that of the latched input, and its MSGIDXT is one level deeper. For example, when the message with MSGID 7 is created without LATCH, its MSGIDXT is 0 (that is to say 0-0-0-0-0-0). When this message is latched in, its first latched output is 7-1-0-0-0-0-0, its second, 7-2-0-0-0-0-0, and so on. When message 7-1-0-0-0-0-0 is latched in, it creates latched outputs with message IDs of 7-1-1-0-0-0-0, 7-1-2-0-0-0-0, 7-1-3-0-0-0-0, and so on. The maximum number of latched outputs which may be created at any one level is 62, that is to say the maximum value in any of the six fields is 62. Since there are only six fields, a given message may only spawn six generations of descendants.

The fields of MSGIDXT are named MSGIDXT.MIDXT1, MSGIDXT.MIDXT2, ..., MSGIDXT.MIDXT6.

This form of MSGIDXT has two uses. First, one can tell by inspection of MSGIDXT what the ancestry of a message is. Second, MSGIDXT may be supplied in 'wild-carded' form to M$FWCG, making it simple, for example, to find all the descendants of the message 7-2-3-0-0-0-0.

The second form of MSGIDXT is selected by BIGMXT = YES. In this form, each time a latched output message is created, a new message ID extension is allocated by the comgroup, and used for MSGIDXT. The comgroup maintains the next value to be used for these extensions, in the same way that it maintains the next message ID to use.

Thus, for example, when message 7-0 is latched in, if it creates three latched outputs, they might have message IDs of 7-8, 7-13, and 7-64. When the message 7-13 is subsequently latched in, it might create an output 7-102; when 7-77 is latched in, it might create an output 7-497. Thus, all descendants of the original latched input bear its MSGID, but their MSGIDXTs have no order or particular meaning, and are simply unique. In particular, there is no limit to the number of output messages which may be latched to a particular input, nor is there a limit to the number of generations of its descendants. On the other hand, no ancestry information is contained in the MSGIDXT.

Note that when this form of message ID extension is being used, it is more conveniently referred under its alias of MSGIDBXT (which is of type integer rather than bitstring).

Since MSGIDXT is never supplied by a user in VLP_STATION, it has no default value and cannot be given one when this macro is invoked. Generation of VLP_STATION in static form results in a MSGIDXT of all zeroes.

MSGIDXT.MIDXT1 = VALUE-DEC(0-62). (See MSGIDXT.) The VLP field name of this parameter is VLP_STATION.MSGIDXT.MIDXT1#.

MSGIDXT.MIDXT2 = VALUE-DEC(0-62). (See MSGIDXT.) The VLP field name of this parameter is VLP_STATION.MSGIDXT.MIDXT2#.

MSGIDXT.MIDXT3 = VALUE-DEC(0-62). (See MSGIDXT.) The VLP field name of this parameter is VLP_STATION.MSGIDXT.MIDXT3#.

MSGIDXT.MIDXT4 = VALUE-DEC(0-62). (See MSGIDXT.) The VLP field name of this parameter is VLP_STATION.MSGIDXT.MIDXT4#.

MSGIDXT.MIDXT5 = VALUE-DEC(0-62). (See MSGIDXT.) The VLP field name of this parameter is VLP_STATION.MSGIDXT.MIDXT5#.

MSGIDXT.MIDXT6 = VALUE-DEC(0-62). (See MSGIDXT.) The VLP field name of this parameter is VLP_STATION.MSGIDXT.MIDXT6#.

MSGTYP = VALUE-CHAR(8) specifies the message type being processed. MSGTYP may be wild-carded as in STATION.

On M$READ, MSGTYP specifies the legal message type(s) that may be read. Specifying '?' only or blanks means that any type is legal.

On M$WRITE, MSGTYP sets the message type of the message being written. Blanks or a wild-card ('?') value causes the message type default from the DCB to be used.

Unless MSGTYP is blanks, its first character must be '?' or belong to the list given under NAME# in VLP_CGTYPL.

The default is blanks.

STATION = VALUE-CHAR(8) specifies the name of the station being addressed. STATION may be wildcarded by using '?' as the last non-blank character. In this case, any station with the matching first 'n' characters up to the '?' is valid.

On M$READ, exactly '?' or blanks means any station is valid.

On M$ACTIVATE and M$DEACTIVATE, wild-carding is permitted. See STATION in FPT_ACTIVATE.

On M$WRITE, STATION is ignored if ANYDCB is set, and otherwise specifies the station to which the message is to be delivered. If STATION is wild- carded, then VLP_STATION.CTL.LATCH and CONTINUE (see FPT_WRITE) may not be set, else an error is returned. Note that when STATION is wild-carded, only stations currently extant in the comgroup may be addressed; new stations are never created by this type of write operation. See also VLP_STATION.CTL.ALLTERMS and VLP_STATION.CTL.ALLDCBS. Specifying blanks for STATION means to use the DCB default.

Unless STATION is blanks, its first character must be '?', or belong to the list given under NAME# in VLP_CGTYPL.

The default is blanks.

WAS = {YES|NO} applies to M$WRITE. WAS = YES requests that writing to absent or deactivated station be permitted if the comgroup permits it.

WAS has no meaning for writes to the anonymous queue, that is, when ANYDCB is YES. For directed write operations, that is, when ANYDCB is NO, the meaning of WAS depends upon whether the destination is wildcarded or not.

For single-destination write operations, WAS = NO requests that an error be returned if the addressed station is absent or deactivated. If the station is connected and activated then the write operation takes place irrespective of WAS. If the station is known to the comgroup but not connected (or connected but not active) then the write operation is allowed if WAS is YES and the comgroup permits writing to absent stations. If the station is not known to the comgroup, then if the comgroup has a restricted station list and the specified station is not in the list, the write operation is disallowed. Otherwise, if WAS is YES and the comgroup permits writing to absent stations, then a new station is created under the specified name and the write operation takes place.

For wildcarded write operations, WAS has a slightly different meaning. See ALLDCBS. Note that writing to absent stations under a wildcard is always restricted to those stations currently known to the comgroup (whether connected or absent).

The VLP field name of this parameter is VLP_STATION.CTL.WAS#.

The default is YES.

Data Record Manipulation

CP-6 file management provides a variety of monitor services to manipulate file data on the record level. In addition to read, write, and delete services which function on the record level, the monitor permits a user to check I/O completion, to position or extend a file, or to mark end-of-file.

The CP-6 system is designed to allow maximum flexibility in file access and creation. Thus numerous options are available via the M$READ and M$WRITE monitor services to control such functions as:

- o Record access method - The record access method is not limited by the DCB open process. Instead the user may specify either sequential or direct record access by key to KEYED, INDEXED, RANDOM, IREL, RELATIVE, and IDS files.
- o Data translation - The user can control encryption/decryption and translation on the record level.
- o Data representation - The normal read/write functions transfer ASCII data; however, the user can request binary read/write functions for data stored in non-ASCII formats to transfer 9-bit bytes without altering or checking the high-order bit.

The user supplies buffers for both read and write functions. Buffer requirements are cited in Table 3-3. For write functions, the user adjusts the buffer boundary to control record length. In addition, file management permits reading and writing of partial records under user control.

Table 3-3. User I/O Buffer Requirements

Type	ORG	Requirements
Disk/CP-6 Tape	Random	Starts and ends on a word boundary
	IDS	Starts and ends on a word boundary
	Other	Starts and ends on a byte boundary
Tape	U	Starts and ends on a word boundary
ANS or Managed	D S F	Starts and ends on a byte boundary
EBCDIC	F V	Starts and ends on a byte boundary
Free tape for ASCII write		Starts and ends on a word boundary
Free tape for binary writes		Starts and ends on a double word boundary
Other devices		Starts and ends on a byte boundary

M$READ - Read Record

The M$READ service causes a specified data record to be read into a user buffer in memory. The M$READ service is used for all types of files and devices for which input is appropriate.

The user normally provides a buffer that is large enough to contain the maximum length record. In the normal case if a record exceeds buffer size, the record is truncated and this condition is reported as an error. If a record is smaller than buffer size the remainder of the buffer is unchanged from its previous contents. The M$READ service also provides the option (CONT) to issue several calls to read successive portions of a single record. With continued M$READs there is virtually no maximum record size except for time sharing terminals and comgroups for which the limits are defined as follows:

time sharing terminals	2048 characters
comgroup	4096 characters

Warning: Because of a hardware limitation, no 'lost data' condition (E$LO) is reported when an attempt is made to read from freetape a record larger than the buffer size.

Additional options provided by M$READ to accommodate a variety of record formats are as follows:

o KEY, KEYR, and KEYS determine whether a file is read sequentially or directly by key. (The ACS parameter of M$DCB and M$OPEN does not restrict user access to the file on the record level.)

o SEED determines whether encryption is performed.

o FULL — for random and IDS files — determines whether the granule stamp is read into the user buffer.

o TRANS requests transparency (no translation) on the record.

o BIN requests a 'binary read' instead of the normal 'ASCII read'.

o REREAD — for input originating from a time-sharing terminal — allows the echoing of the previous record.

For most files the user expects read completion before processing continues. When the WAIT=YES parameter is specified or is assumed by default, the normal return is taken when the read completes successfully. If an error occurs, the alternate return is taken with an error code reported. If multiple errors occur, these conditions are reflected in the TYC field of the DCB.

The user may specify WAIT=NO to request that processing continue at the next statement as soon as the read operation is started. In this case the user may call M$CHECK at a later point in the code. The user can also call M$EVENT prior to any I/O requests and specify the EVENT parameter on M$READ; as a result an asynchronous procedure named on M$EVENT takes control when I/O completes and receives the EVENT code specified on M$READ to identify the particular I/O operation. Using M$EVENT and the EVENT parameter allows the user program to be notified when each read or write completes, and the completion status for the individual operation. M$CHECK allows the program to wait for all I/O to complete, and reports the combined completion status of all no-wait operations since the last M$CHECK.

If M$READ specifies a closed DCB, the monitor attempts the M$OPEN service. If the open procedure is successful, the read operation is then performed.

The service call is of the form:

```
CALL M$READ (FPT_READ) [ALTRET (label)];
```

The parameters for this service are as follows:

BIN = {YES|NO} YES specifies that the data is present on the media as a string of bits rather than one character per 9 bit byte. (Used primarily for free tape.) The FPT field name for this parameter is FPT_READ.V.DVBYTE.BIN#. The default is NO.

BP = {YES|NO} YES specifies, for ORG=SE devices only, that trailing blanks are to be considered in determining whether a record has been updated. The effect of this option is quite considerable when the user moves the cursor down the right side of a screen full of short records. The default is NO.

BUF = VARIABLE Locates the user's buffer into which data is to be read. BUF_ contains two subfields: BOUND and BUF$. BOUND contains the buffer size minus one (in bytes); BUF$ is a pointer to the start of the buffer. See REREAD and SINPUTSIZE.

CONT= {YES|NO} specifies a continuation read. CONT=YES is applicable only to consecutive, keyed, and indexed disk files; labeled tape formats F, V, D, S, C and K; and comgroups. CONT=YES is illegal if the file is compressed, encrypted. If SHARE=ALL for a keyed or indexed disk file, KEYS must be specified. To perform multiple reads on a record, the user specifies CONT=NO for the first read, and CONT=YES for subsequent reads on the same record. CONT=YES causes data to be read into the buffer continuing from the end of the last move. When CONT=YES and the 'lost data' error is not reported, the record has been completely read. CONT=NO causes data to be moved to the buffer from the next record, not the current record. The FPT field name for this parameter is FPT_READ.V.DVBYTE.CONT#. The default is NO.

DCB=DCBNAME specifies the DCB associated with the file.

EVENT = VALUE-DEC(0-?). If non-zero, specifies an event number to be reported to this user when this operation completes. If zero, no event is reported. EVENT is ignored (treated as if zero) unless WAIT=NO.

If EVENT is non-zero, the event procedure established by M$EVENT is given control at completion of the read operation whether an error occurred or not. If an event control procedure is not established at the time the I/O completes, the user is aborted. If M$CHECK is also called, the alternate return specified on that call is taken if any errors were reported in the TYC field of the DCB since the last M$CHECK function. The default is zero.

FULL= {YES|NO} YES specifies that the 1-word granule header on random and IDS disk files is to be read into the user buffer, giving 1024 words per block. FULL=NO specifies a block size of 1023 words, which does not include the header word. The default is NO.

ID = {YES|NO} YES specifies that for keyed, indexed, and IREL files only, the record identifier of the record accessed is to be returned into the area framed by the IDBUF parameter. This area must be double-word aligned and must be at least two words long. The first word will be set to the count of the number of record identifiers returned; the second word will contain the record identifier. Each record in a given file has a unique identifier which may be used for any purpose such as enqueuing. The default is NO.

IDBUF = VARIABLE Locates an area into which record identifiers are to be returned. Must be double-word aligned. The first word will contain the number of record identifiers returned, and each word following will contain one record identifier. See ID.

If this is an IREL file, then presence may be specified to locate a key. If the file has a VLP_RECFIELD definition, then presence may be returned through this vector. See the PBS and PBR options.

Both the ID and presence bits always occupy the same vector. The ID is in the first two words of the area, and the presence bits occupy the remainder of the space.

The VLP_ID macro may be used to generate this area.

IDS = {YES|NO} YES indicates that positioning within a set of duplicate keys is to be by record identifier (as returned by the ID option). RECNUM will be ignored. The default is NO.

INDX = VALUE-DEC(0-511). Specifies, for indexed and IREL files only, the index number that is to be read. Zero indicates the same index as the last operation, or the primary index if this is the first operation since the file was opened. One specifies the primary index. Two specifies the first alternate index. The default is zero.

Note: For alternate keyed files, file management maintains a pointer to the next key for each alternate index. Reading with a primary key alters the pointer to the primary index but has no effect on the pointers to the alternate key indices and vice versa.

KEY = VARIABLE Locates a key buffer associated with this read operation. The values of KEYS and KEYR indicate the purpose(s) of this buffer. For keyed files, the key may be up to 255 bytes long and must be preceded by a byte that contains the length of the key in number of bytes. For indexed files, the key is 1 to 255 bytes with no length byte; therefore the key buffer must contain the exact number of bytes for the key. For IREL files, the key is 1 to 511 bytes with no size byte. The key is composed of fields, in the order that they were defined for the key. Each field begins on a byte boundry. The size of the key buffer determines the size of the key. If the M$READ call altreturns with the error E$BADIRKEY, then the bad segment number will be in F$DCB.ARS#. For random, IDS, relative, unit record and consecutive files the key is a 4-byte binary granule or record number. Each key segment must be on a character boundry. The default is NIL.

KEYCHNG = {YES|NO} YES specifies, for indexed and IREL files only, that when reading an alternate index with KEYS=NO that an error (E$KEYCHNG) is to be given if an attempt is made to read a record that has a key that is different from the key of the last record read. The default is NO.

KEYR = {YES|NO} YES specifies that the key of the record read is to be returned to the user in the buffer described by KEY. If no record was located by the operation, the key returned will be the next higher existing key. If there is no next key, then for KEYED files the length byte will be zero and for INDEXED and IREL files the entire key will be zero. The default is NO.

KEYS = {YES|NO} YES indicates that the user has specified in KEY the key of the record that is to be read. If there is no such key, the record with the next greater existing key is read. If NO, the next sequential record is read. This parameter is ignored for consecutive files. The default is NO.

KEYTYPE = {FLDID | COORD | NONE}. This specifies the type of key for this read to a form access device.

NODAT={YES|NO}. This options specifies that the user is to be informed if there is no data available instead of waiting for the data. This option is useful when communicating to either a terminal or a FPRG to not halt processing trying to do a read. If the read can not be completed immediately, an error is returned. The default is NO.

PBR = {YES|NO} YES specifies that presence bits are to be returned. The default is NO.

PBS = {YES|NO} YES specifies that presence bits are specified for this read. KEYS must be specified. If KEYS=YES and PBS=NO, then all ones are assumed for presence bits. The default is NO.

RECNUM = VALUE-DEC(0-262143). Specifies, for indexed and IREL files only, the record number desired within a group of records with duplicate keys. Zero indicates no record number specified, which will give either the first or next record, depending on whether or not a key is specified. The default is zero.

REREAD= {YES|NO} applies to input originating from a time-sharing terminal. REREAD=YES specifies that the last input line is to be echoed and set to be the current input line. The line to be echoed is either the last line typed by a time-sharing user, the line referenced via a call to the M$SINPUT service (see Section 5) or the line being passed on this M$READ via SINPUTSIZE. The FPT field name for this parameter is FPT_READ.V.DVBYTE.REREAD#. The default is NO.

REREADPOS = VALUE-DEC(0-2048). Specifies, for terminal devices only, the position on the device the cursor is to be placed for this read. This parameter only has effect if SINPUTSIZE is nonzero. The default is zero.

SEED=VALUE-BIT(36) specifies an octal or bit string which is to be used as the seed for decrypting this record. Encryption is permitted for disk files (except for indexed files) and is illegal for any file with the compressed attribute. The default is '0'B.

SINPUTSIZE = VALUE-DEC(0-2048). Specifies, for terminal devices only, the size of the data contained in BUF (to set the input to) before the read is performed. If this value is zero the input will not be set. This parameter only has effect if REREAD=YES. The default is zero.

STATION = VARIABLE. Locates a VLP_STATION that describes the station(s) and message type(s) that may be returned on this read to a comgroup. If STATION is not specified on the read or in the DCB, "any station-type" is assumed.

STRPVFC = {YES|NO} specifies that the VFC character from the record, if it exists, is to be stripped from the record and stored in VFCCHAR# in the DCB. The first byte returned in the record buffer will be the character after the VFC character. The default is NO.

TRANS= {YES|NO} YES specifies transparency, i.e., that data is to be placed in the user buffer without translation. This option is useful primarily for devices which normally perform translation (for instance, time-sharing terminals). The FPT field name for this parameter is FPT_READ.V.DVBYTE.TRANS#. The default is NO.

WAIT= {YES|NO} is only meaningful for random and IDS disk files with FULL=YES and for comgroups. WAIT=YES specifies that the operation is to be completed before control is returned to the user program. WAIT=NO requests the monitor to transfer control to the next user statement after the read operation is started. If WAIT=NO, the user typically uses the M$CHECK monitor service to check the termination of the I/O operation or uses EVENT to receive control when the I/O operation terminates. If an error is encountered when WAIT=NO, the alternate return from M$READ is taken for all errors that do not appear in the TYC field of the DCB; the alternate return from M$CHECK is taken from all errors that do appear in the TYC field of the DCB. Many I/O media complete the operation before returning control to the user's program regardless of the value of WAIT, however, M$CHECK EVENT processing always take place as just described. The default is YES.

M$WRITE - Write Record

The M$WRITE service causes a data record stored in a buffer in memory to be written. The M$WRITE service is used for all types of files and devices for which output is appropriate.

The user normally provides a buffer that is large enough to accommodate the maximum length record. To write records of varying sizes, the user adjusts the FPT field BUF_ which contains a vector: BUF_.BUF$ points to the start of the buffer and BUF_.BOUND specifies the record length minus 1. The user typically adjusts the BOUND field before writing each variable-length record to the record length minus 1. M$WRITE provides options comparable to the options available on the M$READ service. With continued M$WRITEs there is virtually no maximum record size except for time sharing terminals and comgroups for which the limits are defined as follows:

time sharing terminals	2048 characters
comgroup	4096 characters

In addition, M$WRITE can accommodate format options particular to output:

- o The system normally removes trailing blanks from unit record and TERMINAL record output. BP allows the user to retain trailing blanks in such output.

- o VFC specifies where the vertical format control information is to be obtained for this write: in the first byte of the buffer or from the DCB.

- o NEWKEY specifies whether writing of a new record with an existing key is to be reported as an error, or whether writing of a record for which there is no existing key is to be reported as an error. ONEWKEY causes a temporary override of NEWKEY=YES.

- o REWRITE allows rewriting a record in a consecutive disk file while retaining all subsequent records in the file.

If M$WRITE specifies a closed DCB, the monitor attempts the M$OPEN service. If the open procedure is successful, the write is then performed.

For most files the user expects write completion before processing continues. When the WAIT=YES parameter is specified or is assumed by default, the normal return is taken when the write completes successfully. If an error occurs, the alternate return is taken with an error code reported. The error code reflects the first bit set in the DCB TYC field, if the write operation was actually started. The user may perform further tests on the DCB TYC field in the alternate return code sequence. The user may specify WAIT=NO to request that processing continue at the next statement as soon as the write operation is started. In this case the user may call M$CHECK at a later point in the code. The user may also specify EVENT to cause a procedure to be called when the I/O completes.

If a write to a file with a VLP_RECFIELD definition altreturns with the error E$BADRFIELD or E$PARTFIELD, then the field number in error will be in F$DCB.ARS#.

The service call is of the form:

CALL M$WRITE (FPT_WRITE) [ALTRET (label)];

The parameters for the services are as follows:

BIN={YES|NO} YES specifies that the data is to be presented to the media as a string of bits rather than one character per 9 bit byte. The FPT field name for this parameter is FPT_WRITE.V.DVBYTE.BIN#. The default is NO.

BP={YES|NO} YES specifies that trailing blanks are not to be removed on devices which perform this operation most notably the timesharing terminal. NO, the default, specifies that blank stripping is to be performed. The FPT field name for this parameter is FPT_WRITE.V.DVBYTE.BP#. The default is NO.

BUF = VARIABLE Locates the user's buffer from which data is to be written. BUF_ contains two fields: BOUND and BUF$. BOUND, which specifies the buffer size minus one, must be adjusted by the user when variable-length records are written. BUF$ is a pointer to the start of the buffer.

CONT= {YES|NO} specifies a continuation write. This option is useful for writing very long records or for writing data stored in various parts of memory without collecting the data in a contiguous memory area. CONT=YES is applicable to comgroups; consecutive, keyed, and indexed disk and labeled tape files; labeled tape in V format; and labeled and managed tape in D, S, and F format. CONT=YES is illegal if the file is compressed, encrypted, or if SHARE=ALL on a non-comgroup file. To perform multiple writes for the same record, the user specifies CONT=YES for all writes except the last write for this record which must specify CONT=NO. If CONT=YES was specified on the previous M$WRITE, the contents of BUF are appended to the current record. For KEYED and INDEXED files the KEY parameter must be specified and must be the same for all writes to a single record. For writes to a file with a VLP_RECFIELD definition, the portions of the continued record written must contain only whole fields. The record is not allowed to end in the middle of a field. CONT=NO specifies that the data to be written is the last or only portion of a record. The FPT field name for this parameter is FPT_WRITE.V.DVBYTE.CONT#. The default is NO.

DCB=DCBNAME specifies the DCB associated with the file.

EVENT = VALUE-DEC(0-?). See M$READ.

FULL = {YES|NO} YES specifies that the random or IDS file block size is 1024 words and that the user supplies the first data word in word 1 and leaves word 0 for the granule stamp. FULL=NO specifies that the block size is 1023 words, all of which contains user data. The default is NO.

ID = {YES|NO} YES indicates that the record identifier for the record being written is to be returned into the area framed by IDBUF. The default is NO.

IDBUF = VARIABLE Locates an area into which record identifiers are to be returned. Must be word aligned. The first word will contain the number of record identifiers returned, and each word following will contain one record identifier. See ID.

If the file has a VLP_RECFIELD definition, then presence bits may be specified on the write. If no vector is specified for the presence bits, then all ones will be assumed. If the record written is too short to contain all of the fields in the VLP_RECFIELD definition, then zeroes will be assumed for the fields that were not written.

The write to a file with a VLP_RECFIELD definition must not end in the middle of a field.

Both the ID and presence bits always occupy the same vector. The ID is in the first two words of the area, and the presence bits occupy the remainder of the space.

The VLP_ID macro may be used to generate this area.

KEY = VARIABLE Locates a key buffer associated with this write operation. If KEY is not specified, the record is written at the current file location. For keyed files, the key may be up to 255 bytes long and must be preceded by a byte that contains the length of the key in number of bytes. For indexed and IREL files, the key is extracted from the user's buffer. Therefore, no key buffer is required to perform indexed writes since it will not be used. For random, IDS, and relative files, the key is a 4-byte binary granule or record number. The default is NIL.

KEYTYPE = {FLDID|COORD|BIN10|BINHLF|BIN521|STRING}. This field specifies the default KEYTYPE for reads and writes to a form device and the KEYTYPE for ORG=SE. The KEYTYPEs are described below:

FLDID refers to a field identifier (i.e., FPT_DCLFID.ID, a byte-aligned 2-byte value). See M$DCLFLD for details.

COORD refers to the coordinates of a field (i.e., a byte-aligned, 2-byte structure consists of the line and column of the field's location).

BIN10 specifies that KEYINCR is to be divided by 10 until it fits. If KEYINCR=1000, then the first of 1000, 100, 10, and 1 that fits is used.

BINHLF specifies that KEYINCR is to be halved. If KEYINCR=32, then the first of 32, 16, 8, 4, 2, and 1 that is small enough will be used.

BIN521 specifies that division by 5, 2.5, and 2 is to be repeated. If KEYINCR=1000, then the sequence 1000, 500, 200, 100, 50, 20, 10, 5, 2, and 1 is used.

STRING specifies that key incrementation is not possible.

NEWKEY = {YES|NO} applies to KEYED, INDEXED, IREL, and RELATIVE disk files only and is meaningful only if ONEWKEY is NO. NEWKEY=YES specifies that the key is a new key in the file. That is, the key of the record to be written must not already exist; if it does exist, an error is reported. NEWKEY=NO specifies that the key is an old key in the file. That is, a record with this key must already exist; if it does not exist, an error is reported. The default is NO.

NODAT={YES|NO}. This options specifies that the user is to be informed if there is no throttling available instead of waiting for the throttling (i.e., the other end has not done a read). This option is useful when communicating to either a terminal or a FPRG to not halt processing trying to do a write. If the write can not be completed immediately, an error is returned. The default is NO.

NOTIME={YES|NO} applies to IDS files only. NOTIME=YES specifies that the placing of the time stamp in the granule is to be inhibited. Thus the first word in the granule is available for user data. The default is NO.

ONEWKEY={YES|NO} applies to KEYED, INDEXED, IREL, or RELATIVE files only. ONEWKEY=YES specifies that the NEWKEY option is to be overridden. That is, a record is written with the specified key whether it existed previously or not. ONEWKEY=NO specifies that NEWKEY is applicable. The default is YES.

REWRITE={YES|NO} applies to consecutive disk files only. REWRITE=YES specifies that a currently existing record is to be rewritten. The previous operation must have been a read or position operation. This option is intended for use when rewriting a record of the same size. If the new record is shorter than the existing record, the original record size is maintained and the original data occupies the excess area. If the new record is longer than the existing record, the excess is truncated. An error (E$LD) is returned if one of these conditions occurs. REWRITE=NO specifies that the new record is to be written at the current file position and any records from this position to the end of the file are to be deleted. The default is NO.

RRR = {YES|NO} specifies, if YES, Return Receipt Request. This option specifies that the function is not complete until it reaches its destination correctly. Functions to a user terminal are normally considered complete when the data leaves the user buffer. There is no way to know if the data reached the terminal correctly or if any errors occurred performing the function. If RRR is set, the function is not considered complete until the data reaches the terminal and any error status has been returned. The default is NO.

SEED = VALUE-BIT(1-36). See M$READ.

STATION = VARIABLE. Locates a VLP_STATION that describes the destination station and the message type to be used on a write to a comgroup. If STATION is not specified either on the write or in the DCB, the default is to use the values from the last I/O.

TRANS={YES|NO} YES specifies transparency, i.e., that data written to the medium is to be sent without translation. If TRANS=NO the data is sent with the normal translation, which is as follows. For magnetic tape, translation to EBCDIC occurs if EBCDIC=YES or CNVRT=YES was specified when the file was opened. For unit record devices, blanks replace any non-printable characters. For TERMINAL organization, the handler performs translation appropriate to the terminal type. The FPT field name for this parameter is FPT_WRITE.V.DVBYTE.TRANS#. The default is NO.

VFC={YES|NO} YES specifies that the first character of the record is to be interpreted as a vertical format control character. Refer to the CP-6 Programmer Reference Manual for a list of VFC codes. The FPT field name for this parameter is FPT_WRITE.V.DVBYTE.VFC#. The default is NO.

WAIT = {YES|NO} See M$READ.

M$PRECORD - Position to Record

The M$PRECORD service permits the user to change position within a disk or tape file. Based on the parameters supplied, the monitor positions by key or positions forward or backward by a specified number of records.

Positioning by key applies to all file organizations except consecutive and unit record. To position by key the user specifies these parameters: DCB, KEY, KEYS=YES, and optionally KEYR=YES and N. If KEYR=YES the monitor returns the key of the record found, or if the requested key does not exist the next larger key. If N is also specified, the file is positioned by key and then forward or backward by the number of records specified by N.

The M$PRECORD service positions the file so that the next record read will be the record with the specified key, if a record with the specified key exists. The alternate return is taken with an error reported (E$NOKEY), if a record with the specified key does not exist; the file remains positioned to read the record with the next higher key.

NOTE: For keyed, indexed, and IREL labeled tape files opened with ACS=DIRECT, and for disk files the M$PRECORD service with KEYS=YES searches the entire file for a record with the specified key. If the key is not found, the file remains positioned to the record with the next higher key. For keyed, indexed, and IREL labeled tape files opened with ACS=SEQUEN, the file is searched in the forward direction; the search is terminated if a key of equal value is found or if end-of-file is reached.

The user may request relative positioning by specifying KEYS=NO and N as the number of records to skip from the current position. The number of records to skip may be a positive number to move forward or a negative number to move backward . At the time the M$PRECORD is called, current position is considered to be (1) the next record to be read if the previous operation was a read, (2) end-of-file if the previous operation was a write for a consecutive disk file or tape file, (3) the next record if the previous operation was a write for other disk files, and (4) the record which would be read next if the previous operation was a call to M$PRECORD or M$PFIL. If the N parameter specifies a value which would cause positioning to the end of the file, then a normal return will be taken, and the file will be positioned such that the next read would return an end of file condition; if KEYR was specified, then no key would be returned. If the N parameter specifies a value which would cause positioning beyond the limits of the file, the alternate return is taken; the file remains positioned before the first record or after the last record depending on the value of N.

NOTE: If the same FPT is used to position by key at times and relatively by number of records at other times, N must be cleared before positioning by key.

After the relative positioning operation, the number of records actually skipped is returned in the ARS field of the DCB.

The service call is of the form:

CALL M$PRECORD (FPT_PRECORD) [ALTRET (label)];

The parameters for this service are as follows:

DCB=DCBNAME specifies the DCB associated with the file.

ID = {YES|NO} YES specifies that for keyed, indexed, and IREL files only, the record identifier of the record accessed is to be returned into the area framed by the IDBUF parameter. This area must be double-word aligned and must be at least two words long. The first word will be set to the count of the number of record identifiers returned; the second word will contain the record identifier. Each record in a given file has a unique identifier which may be used for any purpose such as enqueuing. The default is NO.

IDALL = {YES|NO} YES specifies that record identifiers for all records with the specified key are to be returned, instead of just the current record identifier. If all of the record identifiers will not fit in IDBUF, then the first word of IDBUF will be the number of duplicate records, not the number of identifiers returned. IDALL may not be specified if PBS or PBR is specified. The default is NO.

IDBUF = VARIABLE Locates an area into which record identifiers are to be returned. Must be double-word aligned. The first word will contain the number of record identifiers returned, and each word following will contain one record identifier. See ID and IDALL.

Presence bits may be specified and returned through this vector. See PBS and PBR.

If the ID or IDS options are used, then VLP_ID should be used to generate the area used for IDs and presence bits. If ID or IDS is specified, then the ID and presence bits will both be in the same vector. The ID will occupy the first two words of the vector, and the presence bits will occupy the remainder of the space.

If IDALL is specified, then presence bits may not be specified, or returned. The VLR_ID macro should be used.

IDS = {YES|NO} YES indicates that positioning within a set of duplicate keys is to be by record identifier (as returned by the ID option). RECNUM will be ignored. The default is NO.

INDX = VALUE-DEC(0-511). For indexed and IREL files only, specifies the index number in which the positioning is to occur. See M$READ. The default is zero.

KEY = VARIABLE Locates a buffer which contains the key of the record to be found in the positioning operation. If KEYR=YES the monitor returns in this buffer the key of the record found as a result of the positioning operation.

KEYR={YES|NO} KEYR=YES specifies that the key of the record found as a result of the positioning operation is to be returned in the KEY buffer. If no key was located by the positioning operations, the key returned will be the key of the next higher existing key. If there is no next key, then for KEYED files the length byte will be zero and for INDEXED and IREL files the entire key will be zero. The default is NO.

KEYS = {YES|NO} YES specifies that the file is to be positioned to the record with the key equal to the key specified by the KEY buffer, or if there is no such record to the record with the next greater key. If KEYS=YES and N is other than zero, positioning according to N occurs after positioning by key. The default is NO.

N=VALUE-DEC(1-?) specifies the number of records to be skipped. A positive number causes the file to be positioned forward; a negative number causes the file to be positioned backward. The default is 0.

PBR = {YES|NO} YES specifies that presence bits are to be returned. The default is NO.

PBS = {YES|NO} YES specifies that presence bits are specified. KEYS must be YES. If KEYS=YES and PBS=NO, then all ones are assumed for presence bits. The default is NO.

RECNUM = VALUE-DEC(0-262143). See M$READ. The default is zero.

M$DELREC - Delete Record

The M$DELREC service deletes one or more records from a CONSECUTIVE, KEYED, INDEXED, IREL, or RELATIVE disk file. The user must have DELR permission to perform record deletion on a file (see VLP_ACCESS or VLP_ASCVEH macros for a discussion of file access permissions). The deletion may be performed at the current record or at the record specified by key. Once a record is deleted, it cannot be read. The deleted record continues to occupy physical space on the medium until the file is copied over itself or for KEYED, INDEXED, and IREL files until another record is written in that location of the file.

For KEYED, INDEXED, IREL, or RELATIVE files the user may call M$DELREC specifying DCB and KEY to request positioning and deletion of the record specified by KEY. In addition, the user may supply the LKEY (last key) parameter to specify that records in the range KEY to LKEY are to be deleted. For files with alternate indices the user may also specify an ID or RECNUM to locate a duplicate key in an alternate index. The ALLDUPS option allows deletion of all or part of a list of duplicate keys within an alternate index. In all cases, the M$DELREC service sets the ARS field in the DCB to the number of records deleted. If LKEY is NIL or is less than KEY, only the record identified by KEY is deleted. It is not considered an error if there are no records in the range defined by KEY and LKEY; the file remains positioned to the record with the next higher key. If there is no record with the specified KEY and LKEY is not specified, the alternate return is taken; the file remains positioned to the record with the next higher key. If KEY is not specified, or if the file is CONSECUTIVE, then the previous operation must have been a successful M$READ or M$PRECORD, and the record read or positioned to is deleted.

If the user calls M$DELREC specifying a DCB that is closed, the DCB is opened automatically using information currently available in the DCB.

If M$DELREC is issued to a RANDOM or IDS file, then the block number specified by KEY, to the range of blocks specified by KEY and LKEY, is overwritten. An attempt to read those blocks will result in the same error that occurs when an attempt is made to read a block that has never been written.

M$DELREC is ignored for labeled, managed, and free tapes.

The service call is of the form:

CALL M$DELREC (FPT_DELREC) [ALTRET (label)];

The parameters for this service call are as follows:

ALLDUPS = {YES|NO}. Specifies that all duplicates of an alternate key, from the specified position to the end of the list of duplicates are to be deleted.

DCB = DCBNAME   specifies the DCB associated with the file.

IDBUF = VARIABLE Locates an area which specifies a record identifier to locate a specific record in a list of duplicates. If an IDBUF is specified, then RECNUM is ignored.

If this is an IREL file, then presence bits may be specified to locate a key. If a vector is not specified, then all ones will be assumed for the presence bits.

The VLP_ID macro may be used to generate this area.

INDX = VALUE-DEC(0-511). For indexed and IREL files only, specifies the index number of the key to delete. See M$READ. The default is zero.

KEY = VARIABLE Locates a buffer containing the key of the record to be deleted or the first record in a range of records to be deleted.

LKEY = VARIABLE Locates a buffer containing the key of the last of a series of records to be deleted.

RECNUM = VALUE-DEC(0-262143). For indexed and IREL files only, specifies the record number within a set of duplicate keys of the record to be deleted. See M$READ. The default is zero.

M$REW - Rewind

The M$REW service applies to disk and tape. For disk and labeled and managed tape files, M$REW positions to the beginning-of-file (just the same as M$PFIL with BOF=YES). For free tapes, M$REW rewinds to beginning-of-tape. If the user calls M$REW with a closed DCB, the DCB is opened automatically using the information currently in the DCB. If no DCB is specified, the serial number parameter specifies which tape volume to rewind. The volume must currently belong to the user and cannot be open.

The service call is of the form:

CALL M$REW (FPT_REW) [ALTRET (label)];

The parameter for this service is as follows:

DCB=DCBNAME specifies the DCB associated with the disk or labeled tape file or with the device tape.

INDX = VALUE-DEC(0-511). For indexed and IREL files only, specifies the index number that is to be positioned. See M$READ. The default is zero.

PSN=VALUE-CHAR (6) specifies the serial number of the tape volume to rewind if no DCB is specified.

M$PFIL - Position File

The M$PFIL service causes positioning of the medium to the beginning or the end of the current file. For disk and labeled and managed tape files, the position is set before the first record or beyond the last record in the file. Free tapes are positioned beyond the next file mark in either the forward or reverse direction. If the user calls M$PFIL with a closed DCB, the DCB is opened automatically using the information currently in the DCB.

The service call is of the form:

CALL M$PFIL (FPT_PFIL) [ALTRET (label)];

The parameters for this service are as follows:

BOF= {YES|NO} BOF=YES specifies that the file is to be positioned at its beginning. For free tapes, the monitor positions the tape immediately before the previous file mark or to beginning-of-tape if there is no previous file mark. BOF=NO specifies that the file is to be positioned to its end. For free tapes, the monitor positions the tape immediately following the next file mark. If the monitor does not encounter a file mark on the device tape, the tape will run off the end of the reel. The default is NO.

DCB=DCBNAME specifies the DCB associated with the file or device.

INDX = VALUE-DEC(0-511). For indexed and IREL files only, specifies the index number in which the positioning is to occur. See M$READ. The default is zero.

M$WEOF - Write end-of-file

The M$WEOF service is appropriate for only certain devices that require special end-of-file procedures. The M$WEOF service causes a file mark to be written on device tape. (For managed tape, the current buffer is truncated before the file mark is written.) If the user calls M$WEOF with a closed DCB, the DCB is opened automatically using the information currently in the DCB.

The service call is of the form:

CALL M$WEOF (FPT_WEOF) [ALTRET (label)];

The parameter for this service is as follows:

DCB=DCBNAME specifies the DCB associated with the file or device.

M$EXTEND - EXTEND File

The M$EXTEND service allows the user to increase the size of a RANDOM, IDS, or RELATIVE disk file, or a COMGROUP. (The user must have WNEW permission.) If the user calls the M$EXTEND service with a closed DCB, the DCB is opened automatically using information currently in the DCB.

The service call is of the form:

CALL M$EXTEND (FPT_EXTEND) [ALTRET (label)];

The parameters for this service are as follows:

DCB=DCBNAME specifies the DCB associated with the file.

SIZ=VALUE-DEC(1-?) specifies the number of units by which the file size is to be increased. The units are granules for RANDOM files, IDS files, and COMGROUPS, and records for RELATIVE files. The default is zero.

M$REM - Remove or Release Volume

The M$REM service permits a tape volume to be dismounted and optionally permits its respective resource to be released by its volume number.

There is no DCB associated with an M$REM call. The specified tape volume must belong to the user and cannot currently be open.

The form of the call for this service is as follows:

CALL M$REM(FPT_REM) [ALTRET (label)];

The parameters for this service are as follows:

DISP = OPTION Specifies the action to be performed:

REM	Remove specified volume.
REL	Remove specified volume and release associated resource.
PTV	Rewind volume only.

The default is REM.

PSN = VALUE-CHAR(6). The serial number of the volume to be removed.

M$TRUNC - Truncate Buffers

The M$TRUNC service releases any blocking (POOL) buffers associated with a DCB after completion of any outstanding I/O operations. The M$TRUNC service applies to disk files; consecutive, keyed, indexed, IREL, relative, and unit record files on labeled tape; labeled tapes in V format; and labeled and managed tapes in D, S, and F formats. For any subsequent read or write operations for the DCB, the system assigns blocking buffers automatically as needed.

The service call is of the form:

CALL M$TRUNC (FPT_TRUNC) [ALTRET (label)];

The parameter for this service is as follows:

DCB=DCBNAME specifies the DCB associated with the file.

M$CHECK - Check I/O Completion

The M$CHECK service is used in conjunction with read or write operations specifying WAIT=NO. The M$CHECK service waits for all outstanding I/O on this DCB to complete, then returns the combined status of all no-wait reads and writes issued since the last M$CHECK was issued.

If no errors occurred, or if no no-wait I/O has been done since the last M$CHECK on this DCB, the normal return is taken. The alternate return from M$CHECK is taken and the appropriate error code is placed in the TCB, if the type of completion is other than normal.

Note: I/O errors which are not reflected in the DCB type of completion code (TYC) cause the alternate return from the M$READ or M$WRITE service that attempted to initiate the I/O operation.

The service call is of the form:

CALL M$CHECK (FPT_CHECK) [ALTRET (label)];

The parameter for this service is as follows:

DCB=DCBNAME specifies the DCB associated with the file or device for which I/O completion is to be checked.

Unit Record Manipulation

The monitor services described in this section provide control over output data with unit record organization. Unit record formatting generally applies to a group of records: printer page formatting or record sequencing for punch cards, for example. Such format options are be specified via the SET command, via M$DCB or M$OPEN, or by services discussed in this section.

In addition the M$LDEV service provides extensive control over unit record data associated with logical devices. For logical devices identified by special name such as LP, LO, etc., the user accepts default attributes for the logical device. The M$LDEV service provides the capability to modify these attributes, to control the format, timing, and content of output to a logical device, and to define new logical device names.

M$LINES - Remaining Print Lines Information

The M$LINES service allows a user to determine the current position of the printer page. The monitor service returns both the current page number and the number of lines remaining on the page. (The Forms Definition File defines page dimensions.) The M$LINES service is appropriate for a DCB associated with unit record data with line printer characteristics; the DCB may be associated with a logical device (including a time-sharing terminal with ORG=UR), a resource device, or a unit record file destined for disk only.

The service call is of the form:

CALL M$LINES (FPT_LINES) [ALTRET (label)];

The parameters for this service are as follows:

DCB=DCBNAME specifies the DCB associated with the unit record data.

LINES= VARIABLE locates an area which may be generated via the VLR_LINES macro which is explained next. The printer page number and current page position are returned in this area.

VLR_LINES

The VLR_LINES macro defines an area into which the M$LINES monitor service places the printer page position information. The area generated by this macro contains the following fields:

PAGE = VALUE-DEC(0-?) Contains the current printer page number. If nothing has yet been printed, this field is zero and RLINES is zero.

RLINES = VALUE-DEC(0-?) Contains the number of print lines remaining on the current page. RLINES is zero, if it is no longer possible to print on the current page, for instance, if the user called M$DEVICE with PAGE=YES. If the DCB is not associated with a device with line printer characteristics, RLINES is set to minus one.

M$DEVICE - Change Device Attributes

The M$DEVICE service allows modification of formatting attributes while output is being created. A number of the options available on M$DEVICE override options which can be specified by the SET command, by M$DCB, or M$OPEN. In addition, the PAGE parameter may be used to leave part of a line printer page blank.

The M$DEVICE service is appropriate for unit record data; the DCB may be associated with a resource device, a logical device, or a unit record file destined only for disk.

The service call is of the form:

CALL M$DEVICE (FPT_DEVICE) [ALTRET (label)];

The parameters for the call are as follows:

DCB=DCBNAME specifies the DCB associated with the unit record data.

DVFC = VALUE-CHAR(1). See M$OPEN.

HDR= VARIABLE Locates an area containing the page header definition. This area may be generated by invoking the VLP_HDR macro. Page headings may be discontinued by specifying HDR=ERASE.

LINES = VALUE-DEC(1-32K). See M$OPEN.

PAGE={YES|NO} PAGE=YES specifies that the rest of the current page is to be left blank.

SEQ = {YES|NO} See M$OPEN.

SEQCOL = VALUE-DEC(1-255). See M$OPEN.

SEQID = VALUE-CHAR(4). See M$OPEN.

TAB= VARIABLE Locates an area that contains the settings for horizontal tabulation. This area may be generated by invoking the VLP_TAB macro. Tabulation may be discontinued by specifying TAB=ERASE.

M$LDEV - Change Logical Device

The M$LDEV service permits the user to perform a variety of functions related to logical devices. Often the user can simply use special names listed in the CP-6 Programmer Reference Manual to access a logical device and bypass use of M$LDEV altogether. The M$LDEV service is only needed in the special cases which are described next.

If the default attributes for a logical device are not appropriate to a user's needs, the user calls M$LDEV to modify the attributes. The critical attributes associated with a logical device are:

- o Workstation of the destination device: default is the user's workstation of origin.
- o Device type: default is the first two characters of the name of the logical device.
- o Form to be mounted on the device: default is defined in the workstation definition file by the installation manager.
- o Number of copies to produce: the system default is one copy.

(If the defaults are satisfactory, M$LDEV is not called. The user assigns a DCB to a special name or logical device name via a SET command, or invokes M$OPEN specifying the special name or logical device name as the RES parameter.)

The user may modify any of the attributes mentioned above by calling M$LDEV. Parameters on this call include: STREAMNAME set to a 4-character logical device name (for example, LP01 or CP01 which are logical devices that are always available to the user) and any one of these parameters: WSN, DEV, FORM, COPIES. If M$LDEV is called prior to writing any output to the logical device, the new symbiont file reflects the attribute modifications. If output has been written to the logical device within this job, the current output retains its attributes and subsequent output is created with the modified attributes. When modifying the attributes mentioned above, a user does not need to be concerned with the additional parameters available on the M$LDEV service with one exception: the default parameter CONTINUE=NO must be in effect when WSN, DEV, or FORM is modified.

Because the most frequent use of M$LDEV is to modify attributes of LP01 or CP01, the other capabilities of M$LDEV are discussed separately below. The following list summarizes the capabilities provided to the user by the M$LDEV service and the concept of logical devices:

(1) Modify logical device attributes such as WSN, DEV, FORM, COPIES (when CONTINUE=NO).

(2) Define new logical devices including formatting attributes.

(3) Modify formatting attributes for a logical device.

(4) Cancel logical device definitions except LP01 and CP01.

(5) Control timing of transmission of output for a logical device.

(6) Delete accumulated output for a logical device.

(7) Create multiple symbiont files within one job for different device types or for the same device type. (This usually involves calling M$LDEV to define new logical devices.)

(8) Permits writing to one logical device via multiple DCBs. (This is an implicit capability and does not require calling M$LDEV.)

The following paragraphs place these functions in the context in which they are likely to be used. The mandatory parameters required for each different function of M$LDEV are also discussed. The discussion is keyed by number to the preceding list of capabilities.

Output to a logical device is accumulated in a symbiont file which, by default, is sent to the destination device when the job terminates. The process of preparing a symbiont file for transmission is called packaging and implies that the next time the user program writes to the logical device, that data is placed in a new symbiont file.

The user may request (5) packaging of accumulated output for a logical device in advance of job termination by calling M$LDEV specifying only these parameters: an existing logical device name (STREAMNAME) and the default parameter CONTINUE=NO. However it is typical that the user requests packaging (1,3) in order to change logical device attributes on the same call to M$LDEV.

The user may also request (6) that accumulated output for a logical device be deleted rather than sent to the destination device. The user deletes a symbiont file by calling M$LDEV specifying these parameters: the logical device name (STREAMNAME) and DELETE=YES. (The next time the user program writes to the logical device, that data is output to a new symbiont file.)

Formatting attributes are assigned to logical devices LP01 and CP01, and to any user-defined logical device. These attributes— DVFC, HDR, LINES, SEQ, SEQCOL, SEQID, TAB —become the default format attributes for any DCB opened to the logical device. (3) The user may modify these formatting attributes by calling M$LDEV specifying STREAMNAME and any one of the formatting attributes listed above. If a DCB is open to the logical device when the user calls M$LDEV, the current symbiont file is packaged and a new symbiont file is started by default. If the user wishes to continue accumulating output in the same symbiont file, he may also specify the parameter CONTINUE=YES on the call to M$LDEV. (If CONTINUE=YES the following attributes cannot be changed: DEV, FPRG, WSN, FORM, COPIES.) The user may override formatting attributes assigned to the logical device via identically-named parameters of MDCB, MOPEN, and M$DEVICE which always take precedence over formatting attributes specified via M$LDEV.

The user may also establish new attributes for an existing logical device by calling M$LDEV with these parameters: STREAMNAME and SCRUB=YES; any new attributes specified on the same call to M$LDEV are applied after defaults are supplied by M$LDEV.

If the user needs to access a device type other than the destination device types associated with LP01 and CP01, the user can call M$LDEV to define a logical device name with attributes describing the device. (2) In this case, the user calls M$LDEV with these parameters: a unique logical device name (STREAMNAME), device type (DEV), and workstation (WSN). Optionally, the user may specify any other appropriate attributes such as FORM, COPIES, and any of the formatting attributes. A total of 16 logical device names can be defined for a job at one time. Two logical devices are always available to the user: LP01 and CP01. Thus the user may define up to 14 logical device names. (4) The user may cancel a logical device name by calling M$LDEV specifying the logical device name (STREAMNAME) and REMOVE=YES.

The timesharing user may call M$LDEV specifying STREAMNAME='UC' to assign attributes to the timesharing terminal (3). Meaningful attributes include DVFC, HDR, LINES, and TAB. In addition, the FORM attribute may be specified to "mount" a form on the terminal. Note that this form "mounting" process does not involve the complex paper-change and alignment processes required at line printer devices; the CP-6 system simply makes itself aware of the characteristics of the requested form. The user may define new logical device names (7) in order to create multiple, separate symbiont files which are sent separately to the destination device(s). By defining one or more logical devices, the user can assign a separate DCB to each logical device. The user program may then perform concurrent writes to the various logical devices. However, the logical device output is kept in separate symbiont files and appears separately when sent to the destination device(s). For example, the user may create two printer listings concurrently within one job. By using the logical device LP01 and defining another logical device name which could be LP02, the user can output through separate DCBs to the two logical devices. The attributes for LP01 and LP02 may be identical or different. Thus the listings could be directed to the same or to different destination devices; formatting may or may not be the same for the listings produced.

The logical device concept also permits the user (8) to open multiple DCBs to the same logical device. When multiple DCBs are open to the same logical device concurrently, then data written through the DCBs is interleaved in one symbiont file; the output appears interleaved at the destination device also. The user program may also send output to the logical device first through one DCB and then through another DCB; the output appears at the destination device in the chronological sequence in which it was written by the user program.

The service call is of the form:

CALL M$LDEV (FPT_LDEV) [ALTRET (label)];

The parameters for this service are as follows:

CONCURR = {YES|NO} CONCURR=YES specifies that output to the logical device may start being transmitted concurrently with creation of later output. This option is seldom useful, since the physical device to which the output is directed must stand idle when it has caught up to program output creation. In specific cases with semi-dedicated devices, however, it may decrease the elapsed data processing time by overlapping physical output with internal computing time.

CONTINUE = {YES|NO} CONTINUE=YES specifies that output to the logical device is to be continued, but with new formatting attributes such as tabstops (TAB), page headers (HDR), page length (LINES), default vertical format control (DVFC), or record sequencing (SEQ, SEQCOL, SEQID). If CONTINUE=YES the destination (WSN), device type (DEV), and form (FORM) must not be changed. CONTINUE=NO specifies that current output to the logical device is to be packaged and any new output to the logical device is sent to a new symbiont file. The default is NO.

COPIES = VALUE-DEC(1-511) Specifies the number of times that the output is to be transmitted to the destination device. This option may be specified on any call to M$LDEV. When CONTINUE=NO, the number of copies pertains to subsequent output to the logical device rather than to output currently being packaged for transmission to the destination device.

DELETE = {YES|NO} DELETE is meaningful when CONTINUE=NO only. DELETE=YES specifies that the accumulated output for the logical device is to be deleted instead of being sent to the destination device.

DEV = VALUE-CHAR(2) Specifies a destination device type. Device type codes are defined by the installation manager in the workstation definition file; typical devices are 'LP' and 'CP'. DEV cannot be changed if CONTINUE=YES.

DVFC = VALUE-CHAR(1) Specifies a default vertical format code character; e.g., 'A' for post-print single spacing. This parameter overrides the intrinsic vertical formatting defined in the forms definition file.

FORM = VARIABLE Locates an area containing the name of the form to be mounted at the destination device. The area may be generated by invoking the VLP_FORM macro. FORM cannot be changed if CONTINUE=YES.

FPRG = FID Specifies the fid of the FPRG to be run at the physical device when output from this logical device arrives there.

FRMSTRM = VALUE-CHAR(2) FRMSTRM is the stream number portion ('00' through '99') of the UC stream specified in DEV (as 'UCnn'). It specifies the window through which STREAMNAME is to be directed (or redirected, for an existing stream). At compile time, this option is specified with the DEV option (DEV=UCnn). At runtime, in order to keep the FPT compatible with previous versions of the system, it must be initialized in two parts (FPT_LDEV.DEV#='UC' and FPT_LDEV.FRMSTRM#='nn'). The default is blanks, which uses the default window for this domain (or doesn't redirect an existing stream).

HDR = VARIABLE Locates an area containing the page header definition. The area may be generated by invoking the VLP_HDR macro. Page headings may be discontinued by specifying HDR=ERASE.

HOLD = {YES|NO} HOLD=YES specifies that output is not to be sent to the destination device, but is to be held for later delivery (see the UNHOLD option of M$JOBSTATS). Multi-copy output may be held after one or more copies have been made; see the HOLDCOPIES option.

HOLDCOPIES = VALUE-DEC(0-511) Specifies, when HOLD=YES, that this number of copies of the output are to be transmitted to the destination device, and then any other untransmitted copies are to be held for later delivery. This option might be useful if, for example, several copies of a report are to be created but the user wishes to check one copy for accuracy before printing the rest (HOLD=YES, HOLDCOPIES=1, COPIES=n). The default value of HOLDCOPIES is 0.

LINES = VALUE-DEC(1-32K) Specifies the number of printable lines per page and overrides the intrinsic page length from the form definition file.

LOGON = {YES|NO} LOGON is meaningful when STREAMNAME is UCnn. LOGON=YES causes the specified stream to be disconnected from the current session and reconnected to CP-6 in the logon state. LOGON=YES must not be specified on the same call to M$LDEV that creates the stream (i.e., the stream must already exist). The default is NO.

REMOVE = {YES|NO} REMOVE=YES cancels the logical device definition. No new output can be written to the logical device name; any DCB open to the logical device is closed automatically. Any parameters specified on this call to modify attributes are ignored when REMOVE=YES. The default is NO.

SCRUB = {YES|NO} SCRUB=YES specifies that any previously defined attributes for the logical device are to be scrubbed, that is, forgotten. New logical device attributes are taken from parameters on this call. In the absence of DEV, WSN, or FORM parameters on this call, the following defaults are used: DEV= first two characters of STREAMNAME (which must correspond to an actual device type defined in the workstation definition file), WSN= user's workstation of origin, FORM= system default form (defined in the workstation definition file for this device type at the user's workstation of origin).

SEQ = {YES|NO} SEQ=YES specifies that record sequencing is to be performed for output to the logical device according to the SEQCOL and SEQID parameters on this call. Record sequencing is typically used on output to card punch devices. Record sequencing can be discontinued by specifying SEQ=NO.

SEQCOL = VALUE-DEC(1-255) See M$OPEN. SEQCOL is meaningful only if SEQ=YES is specified on this call.

SEQID = VALUE-CHAR(4). See M$OPEN. SEQID is only used if SEQ=YES.

STREAMNAME = VALUE-CHAR(4) Specifies the name of the logical device to be defined or modified by this call to M$LDEV. The name can be of the following forms:

1. The name can consist of two alphabetic characters followed by two numeric characters (the names 'LP01' and 'CP01' are always defined).

2. The name can be 'UC' which identifies the current timesharing terminal.

3. The name can be the word 'ALL' which identifies all currently defined logical devices.

The default is blanks.

TAB = VARIABLE Locates an area containing horizontal tabulation settings for the logical device. This area may be generated by invoking the VLP_TAB macro.

WIDTH = VALUE-DEC(1-255) When DCBs are opened to a logical device defined by M$LDEV, the WIDTH field in those DCBs is usually the width of the forms that have been specified or defaulted to in the M$LDEV. If it is desired to not print across the entire forms, a nonzero WIDTH value may be specified in M$LDEV; that WIDTH value is then inserted in DCBs if it is less than form width. The default for WIDTH is zero, which means form width is to become DCB WIDTH.

WINDOW = VARIABLE Locates an area containing the description of the window to use. The area may be generated by the VLP_WINDOW macro.

WSN = VARIABLE Locates an area containing a workstation name of the destination device. This area may be generated by invoking the VLP_WSN macro. WSN cannot be changed if CONTINUE=YES.

Comgroups

The following discussion of comgroups is intended to give an overview of the nature and usage of CP-6 comgroups. Some of the information provided is of a technical and detailed nature, and is given in order that the sophisticated user may intelligently use all of the services which the CP-6 system provides in connection with administration and efficient usage of this facility. The material presented here is intended to bring together, as well as complement, the information provided in the discussions of the individual monitor services.

INTRODUCTION

A comgroup (communications group) is a private message-switching mechanism. Its function is to transmit messages between stations. A station is a 'member' of a comgroup, is the element which transmits and receives messages, and has a unique name. CP-6 provides user program access to comgroups as unique media, through the standard input/output mechanisms.

A comgroup always resides in a disk file. When a comgroup is inactive, that is, no stations are connected to it, it resides entirely in its disk file. When it is active, it resides partly in its file and partly in main memory. The latter permits messages to be transferred without requiring access to disk. When the comgroup goes from active to inactive state, the active context is packaged and stored in the disk file.

A comgroup may be listed with PCL at any time, but may only be copied using EFT, and then only when it is inactive. PCL may be used to delete a comgroup, but only when it is inactive.

THE PHYSICAL COMGROUP

This subsection discusses the aspects of comgroups as files. Although comgroups are used to transmit messages, they have physical existence as disk files, and are therefore subject to many rules as discussed below.

A comgroup is created when a DCB is opened in CREATE mode specifying ASN=COMGROUP. The resultant file is subject to most of the rules and means of access which apply to disk files in general. That is, the file resides in a particular file management account, has a creator, a list of accounts which may READ, WRITE, DELETE and otherwise access it, takes up storage space which is charged to the creator. Thus most of the considerations which apply to KEYED, RANDOM, INDEXED, RELATIVE and CONSECUTIVE files also apply to COMGROUP files. In particular, once such a file has been created it will appear in a PCL list of the account in which it resides, just as any other file. Note, however, that a comgroup cannot be created as a STAR or SCRATCH file.

Certain special attributes of a comgroup file are notable. As discussed in a following subsection, an active comgroup may have a special station called the administrative user (AU). In brief, an AU is a station with wide-ranging powers to control and manipulate the comgroup. The comgroup creator specifies a list of accounts which may be the AU of his comgroup, in the same way that he specifies the list of accounts which may READ the comgroup. This type of access is called AU. Also, CP-6 provides certain services normally used only by an AU which provide information about, but do not change, an active comgroup. The creator may provide a list of accounts which may use these informational services. This type of access is called AURD. A third special type of access control, called TCTL, lists accounts which are permitted to issue terminal control services (e.g., M$STRMATTR) directed at terminals connected to the comgroup.

A user program may not normally open a DCB to a comgroup other than as a comgroup, that is, other than with ASN=COMGROUP. This ASN indicates that the DCB wishes to join the comgroup as a station, and is of course the usual way of opening. The other legal ASN, FILE, indicates a desire to treat the comgroup as a file. If the file is opened by a user who does not possess FMDIAG privilege, then all accesses are turned off, with the exception of DELF, REATTR, EXEC, and NOLIST. Thus a user with REATTR access to a comgroup file may use PCL to modify the file (i.e., change its attributes). When a comgroup is opened as a file by a user with FMDIAG privilege, it may be treated as a RANDOM file, subject of course to access restrictions. Since the data thus read and written is the raw structure of the comgroup, random access mode is of essentially no use other than to special CP-6 processors.

The comgroup file serves multiple purposes. It is, as is the case with all files, the repository of access control information as discussed above. Also, it provides an overflow area for use in the case where the information in the active comgroup is too large to hold in main memory. Lastly, every comgroup file has an area reserved for the active context. When a comgroup becomes inactive, the memory-resident portion of the comgroup is written to this 'save area' of the file. The size of this area must therefore always be at least as large as the memory-resident portion of the comgroup. The size of the comgroup file, and the limits on the size of the memory-resident area, are discussed in detail in the subsection concerning administration.

COMGROUP STATIONS

The unit of membership in a comgroup is the station. A station is either a DCB or a device. DCBs join a comgroup via M$OPEN. Devices join a comgroup by logging on to the CP-6 system. Each station has a name unique within the particular comgroup, the name being limited to 8 characters in length.

A DCB joins a comgroup by issuing an M$OPEN which identifies the comgroup file in the usual way (name, account, password, and possibly packset name), and specifies ASN=COMGROUP. The M$OPEN may specify the station name with which the DCB wishes to join, or may (in absence of such specification) request that an arbitrary name be assigned by the CP-6 system. Note that this latter name is not reproducible, so that if it is used, none of the DCB's business with the comgroup must depend on its station name. If the comgroup is inactive at the time the M$OPEN is issued, it is made active. If any other stations are to be permitted to join the comgroup while this DCB is connected, the DCB must be opened in SHARE=ALL mode (the default). Once connected to the comgroup, the DCB may read and write messages, subject to restrictions detailed in the subsection concerning DCB STATIONS. Reading and writing of messages is accomplished by the M$READ and M$WRITE services.

A device (or TERMINAL, as devices are called within a comgroup) joins a comgroup by logging on to the CP-6 system with a logon which specifies that the device's mode of connection to the system is as a comgroup TERMINAL. Note that among the other possibilities for the connection mode for a given logon is timesharing user. Note also that operator consoles which log on to the CP-6 system do so as comgroup TERMINALs on a special system-controlled comgroup.

The system manager, in creating such a logon, specifies the comgroup to which the device is to be connected in the usual way, and gives the station name with which the device is to join the comgroup. It is important to mention on this context that an active comgroup MUST possess at least one DCB station. In other words, a device which attempts to log on to the CP-6 system with a logon that specifies a comgroup that happens to be inactive cannot be connected to the comgroup.

COMGROUP MESSAGES

A comgroup message has two important parts: the data, which is supplied by the writing station, and attributes, partly assigned by the comgroup and partly assigneu by the writing station (called the origin station). This separation is of great importance to an understanding of the way in which comgroups operate. The content of the data part of a message is of no interest whatsoever to the CP-6 system in terms of managing the comgroup; it is of interest only to the station which will receive the message. The values of the attributes, on the other hand, are of first importance since they control to a large extent how and when the message will be delivered.

The attributes of a comgroup message are here listed:

ID -	An integer message identifier assigned by the comgroup, and unique to this message.
TYPE -	A textual message type assigned by the origin station, and a maximum of 8 characters in length.
TIME STAMP -	A system time stamp assigned by the comgroup at the instant the message is written.
ORIGIN -	The name of the origin (writing) station.
DESTINATION -	A message is destined either for a specific station (called a directed message) or for the anonymous queue. This is a queue into which any station may write messages and from which any DCB station may read messages. DCB stations assign destination when writing messages. When a terminal station writes a message, destination is always the anonymous queue.
PRIORITY -	The priority of a message is determined from its type and origin. Each message type is known to the comgroup, and each has an intrinsic priority. Similarly, each station has an intrinsic priority of submittal. These two values are concatenated to produce the message priority. The most significant contribution to the priority comes from the attribute which keys the anonymous queue (see the discussion of the anonymous queue below). In other words, if the anonymous queue is keyed by message type, then the priority of each message in the comgroup is formulated with the priority of its message type being most significant and the priority of its originating station being least significant, and vice versa if the anonymous queue is keyed by origin station.
SIZE -	The size attribute of a message is the length (in bytes) of the data portion.

The attributes of a message, being necessary to the delivery and manipulation of the message, are kept together and are always logically in main memory, that is, in the active portion of the comgroup. It is convenient to view these attributes as a block which completely describes the message, in other words a message block, hereafter referred to as an MBLK. Should the number of messages in a comgroup be so large that the MBLKs describing them will not fit in main memory, they are written to the file, but this is viewed as an exceptional situation. This treatment is possible because the MBLK is of a fixed and small known size, and facilitates keeping track of a large number of messages in a small amount of main memory.

The data portion of a message is, on the other hand, not necessary to the manipulation of the message, and furthermore is of an arbitrary size since it is assigned when the message is written. Thus the management of the data portions of the messages in a comgroup is a much more unpredictable and complex task. For this reason, the data portion of a comgroup message is always logically on disk, in the comgroup file. The comgroup maintains a disk data cache in main memory. Thus, given a reasonable relationship between the activity level in the comgroup and the amount of main memory available, the number of file accesses for message data is small. The data portion of a message is referred to as a data block or DBLK. Thus a message consists of an MBLK which contains the attributes, and the disk address of the data or DBLK, plus the DBLK itself which contains the data.

Every message in a comgroup belongs to exactly one queue. A queue is a linear list of messages blocks. Therefore even though a queue is said to 'contain messages', it physically only contains the MBLK for each message, with the data part of the message being kept in the data cache. This makes it possible for a comgroup to have a large number of long message queues without necessarily occupying a great deal of main memory. A single comgroup contains many queues. In particular, each station possesses a queue containing the messages which have been written directly to that station but have not been read yet. All comgroup queues are identical in structure and function. Each is ordered by priority, and within priority by time of submittal with the older messages preceding newer ones. When a message is to be read from a queue, it is read from the front. In other words, the message selected is that having highest priority, and within that priority, least recent time of arrival.

COMGROUP STRUCTURE

This section describes the basic structure of a comgroup, in order to provide a framework within which the following discussions can be placed. The nature of this structure is important because it makes it possible to understand what types of operations a comgroup performs in a natural and efficient manner.

Every comgroup contains a height-balanced binary tree with one node for every station known to the comgroup, keyed by station name. In brief, such a structure facilitates rapid lookup by station name, such that the time required to find a particular station is very small irrespective of the number of stations. This tree is called the station tree. The node for a particular station contains all the information that the comgroup requires in order to control the activities of the station. In particular, the queue of messages written directly to the station but not read yet, is headed in the station node. The station tree is always kept in main memory when the comgroup is active, thus a comgroup must always be able to acquire sufficient main memory to hold all the nodes of this tree. Note that the stations listed therein are not only those which are currently connected to the comgroup; stations which are absent but which have messages saved against their future connection are also listed. One of the uses of the station tree is for directed write operations: when one station writes a message to another specifying a direct station name, the destination station is found by looking it up in the station tree; the message is then inserted into the destination station's queue.

Every comgroup contains a height-balanced binary tree with one node for every message type known to the comgroup, keyed by message type name. This tree is called the message type tree. Each node in the tree contains information pertaining to a particular message type, for example statistics pertaining to activity of messages of that type. As in the case of the station tree, the message type tree is always kept in main memory.

The anonymous queue is a comgroup structure which in some ways may be viewed logically as a linear queue. Messages may be written into the anonymous queue by any station, and may be read from it by any DCB station. Terminal stations always write to the anonymous queue. It is called 'anonymous' because it is intended for use in applications where a group of stations submit messages without caring which station processes them, and thus their destination is 'anonymous'. The stations which read this queue may however be interested only in certain message types, or possibly only in messages originating from certain stations. In order to make such searches of the queue faster, the anonymous queue is physically organized either by origin station or by message type. The type of organization is selected when the comgroup is created via the QISS parameter in M$OPEN. If, for example, the anonymous queue is to be organized by message type (the default), then all messages in the anonymous queue of a given message type will be kept in a queue headed in the message type tree node for that message type. The result of this division of the anonymous queue is that a read request for some particular message type from the anonymous queue simply and quickly finds the appropriate node in the message type tree, and then need only examine that shorter queue, rather than search through many messages of the wrong type. Notice that if the anonymous queue is organized by station name, then all messages in the anonymous queue submitted by a particular station will be kept in a queue headed in that station's station tree node. It is important not to confuse this queue with the queue of messages written TO that station. In sum, the anonymous queue is also kept as a binary tree, utilizing either the station tree or the message type tree. The tree in which the anonymous queue is embedded is sometimes referred to as the Q-tree.

It is important to note that the anonymous queue is not strictly a linear queue when viewed by a reader. Anonymous messages are not strictly ordered by priority and time of arrival. Specifically, messages are always read in priority order, BUT not necessarily in order of arrival. This occurs because of the organization of the anonymous queue, in that messages are kept in separate queues, one per Q-tree node. Thus we have the following anomaly:
if the anonymous queue is keyed by message type, and consists of the message types ABC and XYZ, having the same intrinsic priority, and there is one message of each type present, then the message of type ABC will be read first even if it arrived later than the one of type XYZ. This is because ABC occurs alphabetically before XYZ, and thus the ABC subqueue is searched before the XYZ subqueue.

The third binary tree in a comgroup is the data cache tree. Recall that the data cache is used to keep data blocks in main memory as much as possible, despite the fact that they are always logically in the disk file. When a data block is created, it is allocated some part of a granule from the disk file. Each such data granule contains some header information regarding allocation of data blocks within it, some allocated data blocks, and some free space. Once a data block has been allocated, it cannot be moved about within the granule (note that this may result in fragmentation of the granule which the comgroup makes no attempt to alleviate). Subject to main memory limitations (see the section regarding administration), the comgroup will keep as many data granules as possible in memory. Each such granule is entered as a node in the data cache binary tree, which is keyed by disk address. Thus the fetching of a data block is an extremely rapid process if the data block's granule is in the data cache. It is important to note that this cache is NOT write-through, that is, when a data granule is updated, it is not automatically written to the file. This means that a comgroup may be active for a long period of time, with many messages being written and read, without ever having to access the file.

Thus a comgroup contains four basic structures: the station tree, the message type tree, the disk cache tree, and the anonymous queue (which is really superimposed on one of the first two). These structures, together with the various message queues, form the essential elements of an active comgroup.

DCB STATIONS

This section discusses the facilities available to a DCB station on a comgroup.

Like all stations, a DCB station's activities consist of reading and writing messages. This is accomplished by means of the M$READ and M$WRITE monitor services. However, the ability to do so is limited in various ways by the administrative user (if any) of the comgroup, and by the access that the user has to the comgroup. First, reading and writing require, respectively, READ and WRITE access to the comgroup file. Second, neither can be done until the station has been activated. A detailed discussion of this aspect may be found in the subsection on administration; but, in brief, a station is restricted from reading and writing until it is 'activated'. Activation and deactivation of stations takes place at the whim of the AU. M$READ and M$WRITE service requests return an error if the DCB station is deactivated. Third, the AU may restrict, or supply defaults for, the options applicable to M$READ and M$WRITE directed at the comgroup.

Associated with each DCB may be a set of defaults for I/O options applicable to M$READ and M$WRITE. These defaults are contained in a structure called VLP_SETSTA. Refer to the discussion of this structure for details regarding the way in which it supplies defaults for, and restricts, the options available. VLP_SETSTA may be associated with the DCB in any one of three ways. First, it may be specified on a !SET command. Next, it may be supplied on M$OPEN, in which case it will unconditionally replace that specified on !SET. Thus, by the time the DCB attempts to join the comgroup, it may have a VLP_SETSTA associated with it. The comgroup itself may have a VLP_SETSTA, supplied by the AU, which is to be applied to all DCB stations. This is a complex procedure if the DCB already has a VLP_SETSTA. The process which takes place is exactly the same as that which takes place when an M$READ or M$WRITE supplies a VLP_STATION through a DCB which possesses a VLP_SETSTA, and is described in detail in the discussion of VLP_SETSTA. In brief, however, the ILOCK and OLOCK bits in the comgroup's VLP_SETSTA specify whether the DCB's VLP_SETSTA is to be overridden (LOCK = YES) or simply augmented (LOCK = NO) by the comgroup VLP_SETSTA. The result of this process is a single VLP_SETSTA associated with the DCB which cannot be altered by the AU as long as the DCB remains connected to the comgroup.

The M$WRITE service creates a message in the comgroup, and supplies the data, message type, and destination. The data is supplied in the BUF parameter, and the other options of interest in the STATION parameter, which frames a VLP_STATION. These parameters may be supplied as defaults in, or be overridden by, the VLP_SETSTA in the DCB and LASTSTA in the DCB. The latter is a VLP_STATION describing the message most recently read or written by the station. If LSTAOR is set in the DCB, then LASTSTA is updated only by M$READ; otherwise it is updated by M$WRITE as well (see M$OPEN). Note that in the case of a streamed DCB, the LASTSTA involved is that in the 'real' DCB. This means that if two DCBs correspond to a single comgroup station via streaming, an M$READ through the first DCB will set the effective LASTSTA for the next M$WRITE through either DCB. Currently the only use of LASTSTA is to assign destination on M$WRITE if it is not otherwise specified, that is, if neither M$WRITE nor VLP_SETSTA in the DCB specify destination, then LASTSTA is used.

The message may be directed to the anonymous queue or one or more specific stations. Writing to stations not currently connected to the comgroup is controlled by an option in VLP_STATION. The message may be latched to a currently-latched input message (see the discussion of LATCH below). All of these possibilities are discussed in detail under VLP_STATION.

If the CONT option is specified on M$WRITE, it means that the service is not to create a single message, but rather to accept the data being written as one part of a message. This option can be used to create a message of arbitrary length, irrespective of the AU-set limit on message size, since there is no limit to the sum of the sizes of the parts of a message created in this way. A message which is to be created in N parts is built by issuing N M$WRITE services, where the first (N-1) of them specify CONT = YES, and the last specifies CONT = NO. Such a message is referred to as a continued message. When read by a DCB station, a continued message is seen as a single 'record' of data, that is to say the DCB station has no way of knowing that the message was written with CONT, much less any knowledge of the sizes of the individual parts. Note then that CONT on M$WRITE has nothing to do with CONT on M$READ (see below). On the other hand, when a continued message is read by a terminal station, the situation is reversed. The terminal receives, in succession, each part of the message exactly as supplied by the M$WRITE. By 'in succession' we mean that when a terminal 'reads' for a message and happens to find a cd tinued one, then the data supplied in response to that 'read' is the first part of the message (corresponding to the first M$WRITE), and subsequent 'reads' receive the rest of the parts in order. The terminal station may not read the continued message as a single concatenated 'record'. Note that in future releases TERMINAL stations may read CONTinued messages in the same manner that DCB stations read them. If so, however, a method will be provided to produce the current behaviour.

The M$READ service reads one message from the comgroup. The message data is to be read into the buffer specified by BUF. The maximum size of the buffer is 2048 bytes (this is independent of the comgroup control parameter MAXMC). The options controlling the read are specified by the STATION parameter which frames a VLP_STATION. The first important distinction provided therein is which queues are to be searched. If DIRONLY is specified, then only messages written directly to this station are to be considered. Otherwise, the anonymous queue will be searched if no directed message is found. Next, the reader may limit the set of messages to be considered by restricting message type and/or origin station. This is done through the MSGTYP and STATION options in VLP_STATION. Thus specifying MSGTYP = AB? and STATION = GORGO causes the read request to search for a message from station GORGO with message type beginning with AB. The ability to do this may be limited by the AU, who may define a restricted set of stations and/or message types in his comgroup; thus attempting to read for message type ABX may produce an error if that message type could never exist.

Within the set of available such messages, that of highest priority and least recent submittal time is selected. If no qualifying message is available immediately, then normally the read request is left pending, to be satisfied when one such message is written. The reader may, however, specify EOFNONE, requesting an end-of-file error in such a case. The reader may also specify EOFTIME= n requesting an end-of-file error if n seconds go by without any messages entering the comgroup to satisfy this read.

When the DCB is closed, its station disconnects from the comgroup. Depending upon control parameters established by the AU, this may or may not result in any unread messages written to the disconnecting station being deleted. In the latter case the messages will still be available when a DCB (or TERMINAL) re-connects to the comgroup under the same station name. Note, however, that messages written by the disconnecting station are in no way affected by the disconnection. This fact illustrates the essential disjunction between member stations. Once a message has been written, it 'disappears' into the comgroup, to be delivered if and when the destination station wishes to read it, and is not visible in any way to the originator.

When a message is found as a result of an M$READ, it is attached to the reading station as its current input message. Once this has occurred, then LASTSTA in the DCB (which is a VLP_STATION pointed to by DCB.LASTSTA$) will be updated to reflect the message found. Note that this updating will take place even if an error condition is encountered while transferring the data. Errors encountered before a message is found (illegal parameters, for example) do NOT update LASTSTA. The CONT option (see FPT_READ) may be used to read the message in sections, if the buffer is not sufficiently large.

The LATCH option for M$READ has multiple effects. When an M$READ with LATCH finds a message, that message is attached to the reading station as its current latched input message. Such a message may NOT be deleted until it has been read successfully and in entirety. Once it has been so read, it may be unlatched and deleted in one of two ways: by issuing another M$READ, or by issuing the M$UNLATCH service and specifying DELETE. If a message is latched but has not been successfully read, it can only be unlatched through M$UNLATCH, and then cannot be deleted. Instead, it can be put back in the queue from which it came (RERUN), or placed in hold status for the AU to examine it (HOLD).

Note that re-reading a message takes different forms depending on whether CONT or simply LATCH is used. If LATCH is used without CONT, then every M$READ will attempt to transfer the entire message. When CONT is used, each M$READ attempts to transfer only the unread portion of the message.

If a DCB station possesses a successfully-read latched input message, it can 'latch' output messages to the input message. This is accomplished by specifying LATCH on M$WRITE. Such latched outputs are invisible, that is, they do not really go to their destination but are saved at the originating station, to be transmitted when they are unlatched. The message ID created for a latched output message is related to that of the latched input message. See the discussion of MSGID and MSGIDXT in VLP_STATION. The M$UNLATCH service may be used to manipulate these messages. It can delete the most recently-written latched output, or all of them. It can also, in conjunction with unlatching the input message, send all the latched outputs to their respective destinations, or delete them.

Closing a DCB causes an implicit M$UNLATCH. If the DCB is closed SAVE, then the latched input is deleted and the latched outputs sent to their destination; otherwise, the latched input is RERUN and the latched outputs deleted. Note that this latter action is taken in ALL cases other than an explicit M$CLOSE with SAVE in particular if the reading program exits or aborts.

The LATCH feature is therefore useful if the program wishes to ensure that the input message and any resulting output messages are processed completely before any irrevocable action (i.e., deleting the input or sending any outputs) is taken.

TERMINAL STATIONS

A terminal station joins a comgroup by logging on to CP-6 with a logon that has been specified by the system manager to cause connection to a particular comgroup. Such a logon has associated with it a parameter indicating to the CP-6 system the action to be taken in the event that the comgroup is not active. This action may be one of the following:
disconnect the terminal (to permit another logon), send an explanatory message and wait, or simply wait. The latter two cases cause the terminal to remain connected to the CP-6 system, waiting for the comgroup to become active. A terminal in this state is said to be in 'limbo'. When the comgroup becomes active, the terminal will be connected to it. Note that the same sequence of events takes place for all TERMINAL stations on a comgroup when the last DCB leaves the comgroup and thus causes it to become inactive.

The CP-6 system makes no attempt to force a comgroup to become active when a terminal attempts to connect to it.

Once the terminal has been connected, it may or may not be activated, depending on the whim of the comgroup's AU and/or the pre-established control parameters in the comgroup. Until it is activated, it can neither input data nor receive data. When activation occurs, the comgroup will begin to read and/or write the terminal. This activity is controlled by parameters associated by the system manager with the logon used: terminals may be designated as INPUT or OUTPUT or both. An INPUT terminal may only submit data to the comgroup; an OUTPUT terminal may only receive data from the comgroup. The classic examples of such terminals are, respectively, card readers and line printers. Terminals may be designated as both input and output. For example, operator consoles are INPUT/OUTPUT terminals on a special system-controlled comgroup, and as such are permitted to input data to the comgroup (KEYINs), and receive data (operator console messages).

Terminal stations, unlike DCB stations, have their input controlled by the comgroup. Whereas a DCB station may write a message at any time, a terminal station may only submit a message when the comgroup 'reads' for one. Thus INPUT terminals 'write' messages into the comgroup in response to the comgroup 'reading' for data. When a terminal is activated, the comgroup will perpetually 'read' for data from the device; this process ceases when the terminal is deactivated by the AU. This control is necessary because most terminals require the comgroup to request data from them before they will submit data (e.g., card readers). On the other hand, certain terminal devices (operator consoles, for example) will submit data of their own volition. In either case it suffices to have the comgroup perpetually requesting data. Note that terminal stations under normal conditions only submit messages to the anonymous queue. The only exceptions to this rule are for certain special cases. All of the exceptions may be classified under the heading of 'control messages'; thus terminal stations cannot submit true 'data' messages other than to the anonymous queue.

OUTPUT terminals are a slightly different case. From the comgroup's point of view, they, like DCB stations, request data from the comgroup by 'reading' for it. Note that a terminal station may only read messages written directly to it; it may not read from the anonymous queue. Most terminal devices are unintelligent, in the sense that they read for 'any' message and thus will always read the message at the front of their queue. Connection of an OUTPUT terminal to a comgroup causes the CP-6 system to perpetually 'read' the comgroup for data on behalf of the terminal. Thus when an operator console logs on to the CP-6 system, it will perpetually request the comgroup for messages, and output them on the device as they become available.

It is now appropriate to state the following example. All operator consoles are connected as terminal stations to a single system-controlled comgroup. This comgroup is managed by a CP-6 process which has a single DCB open to it. KEYINs typed on the consoles are input by their terminal stations into the comgroup's anonymous queue. The DCB station perpetually reads the anonymous queue. When it reads a message (i.e., a KEYIN), it processes it as appropriate. If the KEYIN requires a response, the DCB station writes that response directly to the station which submitted the original message. Since the terminal is perpetually reading for messages from the comgroup, this response message is eventually read, and appears on the originating console. In addition, each console receives a certain subset of all system activity messages (e.g., requests for tape mounts). The DCB station takes each such message and writes it to every terminal station that corresponds to a console designated for that information. Thus each terminal station is continually receiving messages with data consisting of informational text. Again, since the terminal station is perpetually reading the comgroup, these messages are read and output on the console.

FPRG TERMINAL STATIONS

An FPRG terminal station is a terminal station attached to an FPRG (Front-end user PRoGram) instead of a device. This FPRG is in turn attached to a device. This type of connection allows an intelligent process to be inserted between the comgroup and a device allowing more customized device interactions. FPRG terminal stations are created via the M$ACTIVATE monitor service.

Two forms of FPRG terminal station exist; pure FPRG terminal stations and transformed FPRG terminal stations. All FPRG terminal stations are created by referencing a normal terminal station. Transformed FPRG terminal stations are actually the referenced terminal stations transformed into FPRG terminal stations. Pure FPRG terminal stations are created anew with, of course, their own station name and upon creation are completely independent of the referenced terminal station. The FPRG for either a pure or a transformed FPRG terminal station is attached to the same device as the reference terminal station.

FPRG terminal stations require the presence of an AU. As a result of this, FPRG terminal stations may not exist without the presence of an AU. Whenever the AU disconnects from the comgroup, all FPRG terminal stations are automatically deactivated. This deactivation will cause transformed FPRG terminal stations to revert to deactivated normal terminal stations and cause pure FPRG terminal stations to be disconnected from the comgroup.

ADMINISTRATION

This section discusses in some detail the powers and responsibilities of a comgroup administrative user (AU).

A comgroup may have at most one AU at any given time. The AU is a DCB station having special powers to control the comgroup. In order to become the AU of a comgroup, a CP-6 user must open a DCB to that comgroup and request (via the AU parameter on M$OPEN) to become AU. The request is accepted provided that the comgroup does not presently have an AU and the user's account has AU access to the comgroup file. Note that an AU station may use any station name it wishes.

Upon joining the comgroup as AU, the DCB acquires certain powers and responsibilities. The powers take the form of special monitor services which change and control the comgroup. The responsibilities take the form of certain actions which the comgroup expects the AU to take in order to ensure proper operation of the comgroup.

It is important to note that a comgroup is not required to have an AU. The default comgroup control parameters are set up in such a way that a small and simple comgroup application need never have an AU connected to the comgroup. Furthermore, certain large and complex, but relatively stable applications may only require an AU to connect to the comgroup once (in order to establish the desired control parameters). Thereafter, the comgroup is capable of running without the aid of an AU. Only when the complexity of the environment in the comgroup exceeds the ability of the CP-6 system to control the comgroup's operations, is the constant presence of an AU required.

The AU controls the comgroup by means of special information provided him by the comgroup, and by means of the following special monitor services: M$CGCTL, M$ACTIVATE, M$DEACTIVATE, and M$FWCG.

M$CGCTL establishes the parameters which tell CP-6 file management how the comgroup is to be operated. The details may be found in the section describing this monitor service. Essentially, however, M$CGCTL describes the physical parameters of the comgroup, and allows a wide range of different 'types' of comgroups to be established. For example, the AU can supply a list of legal station names and message types for his comgroup, and further specify that any others are illegal. He can also provide a VLP_SETSTA which specifies defaults for, or restrictions on, DCB stations' read and write operations.

M$CGCTL specifies how much main memory is to be allocated to the active comgroup, and how that memory is to be divided between the data cache and everything else. A single comgroup is currently limited to 250 pages of main memory.

The special information provided to the AU, in order to aid in control of the comgroup, takes many forms. The AU may, via M$CGCTL, specify which types of such information he wishes to receive.

The M$ACTIVATE and M$DEACTIVATE services are used to perform control actions on specific stations. Their most important function is to activate, deactivate, and disconnect stations.

A station may not read and write messages unless it is 'activated'. The AU may activate or deactivate a station at any time. Furthermore, TERMINAL stations may be disconnected from the comgroup and from the CP-6 system by means of M$DEACTIVATE with DISCONNECT specified. A DCB station will receive a 'deactivated' error if it issues an M$READ or M$WRITE service and is not active. Note also that if a DCB station has a read request pending and is then deactivated by the AU, the request is terminated with the same error.

In addition to activation and deactivation, other actions (such as deleting a station's message queue and creating FPRG terminal stations) may be specified.

The M$FWCG service selects a subset of the messages within a comgroup, and then optionally produces a report concerning the messages found, as well as optionally deleting or changing them. The report feature is limited to users having AURD access to the comgroup. The delete or change feature is limited to the AU.

The M$CGINFO service provides a snapshot of the current state of various parts of the comgroup. Its use is not limited to the AU; any DCB station which has AURD access to the comgroup file may issue this service.

The information provided to the AU by the comgroup takes the form of special messages directed to his station. These messages are distinguished by their message type: '*AUEV'. Each such message has a code in it, indicating what event caused the message to be generated. The messages are:

MESSAGE	EXPLANATION
DCB open	A DCB has joined the comgroup
TERMINAL connect	A TERMINAL station has joined the comgroup
DCB close	A DCB station has left the comgroup
TERMINAL disconnect request	A TERMINAL station is attempting to leave the comgroup (see DSCREQ in VLP_CGCP).
TERMINAL disconnect	A TERMINAL station has left the comgroup
TERMINAL break	A TERMINAL station has submitted a break event. This can be an asynchronous terminal device BREAK key or CONTROL-Y, or a card reader sending input-ready.
UNLATCH	A DCB station has issued an M$UNLATCH service that caused a message or messages to be unlatched.
WARNING	The comgroup file has become low on disk space, or the amount of main memory available to the comgroup has become low.
FPRG TERMINAL exit	An FPRG TERMINAL station is now at jobstep due to issuing an M$EXIT. It will now accept an M$SETFP call.
FPRG TERMINAL abort	An FPRG TERMINAL station is now at jobstep due to aborting. The reason for the abort is reported in B$CGAURD.INFO in B$JIT.ERR form. It will now accept an M$SETFP call.
FPRG TERMINAL ldtrc complete	An FPRG TERMINAL station just successfully completed an M$LDTRC. The B$CGAURD structure may be followed immediately by a VLP_FPRG to obtain the new FPRG's description.

The structure of such messages is detailed in the discussion of the macros B$CGAURD and B$CGAUCI. In general, however, each message contains information pertinent to the event which has occurred. For example, the information accompanying the TERMINAL CONNECT message includes the station name with which the device has joined the comgroup, the name and account it specified in logging on to the CP-6 system, and information about the device's physical characteristics.

These special messages have the highest priority of any message in the comgroup. Therefore, if an AU reads through his AU DCB without specifying any STATION or MSGTYP restrictions, he will naturally receive AUEV messages in preference to any other messages which may be present. The comgroup, however, takes even stronger measures to ensure prompt delivery of these messages. The STATION and MSGTYP parameters specified on an M$READ through an AU DCB will be ignored if an AUEV message is present. In other words, an AU must be prepared to receive AUEV messages in response to an M$READ that attempts to read, for example, only messages of type 'XYZ'.

An important responsibility of an AU is the maintenance of available space in the comgroup. The AU's interaction with CP-6 file management to this end is rather complex.

First, and simplest, is the control of the comgroup file. The size of the file is established first by the IXTNSIZE parameter on M$OPEN. Any subsequent increase must take place via the M$EXTEND monitor service. In this sense a comgroup is treated like a RANDOM file: CP-6 file management will never automatically allocate more space to the file. The file can be viewed as consisting of three sections: the control area, the save area, and the message storage area. The control area is of concern only to CP-6 file management, and is about one granule in length. The save area is the section of the file to which the active comgroup is written when the comgroup becomes inactive. The size of this area is MAXPG granules, where MAXPG is the parameter specified in VLP_CGCP. Since this parameter can be changed at any time by the AU, CP-6 file management ensures that the save area will be available when needed by reserving the appropriate number of granules in the file for that purpose. Note then, that an M$CGCTL service which attempts to increase MAXPG may be rejected if the file contains insufficient free space for the necessary increase to the save area.

The rest of the file is dedicated to message storage. It is this section of the file that is cached in the data cache tree. Data cache granules are written to this section of the file as required. Note, however, that every data block, and hence every data cache page, logically belongs on disk and thus causes space to be allocated from the message storage area of the file. Thus, no matter how large the data cache may be, the message storage area of the file must be sufficiently large to hold the data blocks of all the messages in the comgroup. In order to allow the file to grow if necessary, the comgroup will send a special warning message to the AU when the amount of free space in the message storage area of the file sinks below a preselected level. The AU is then expected to react by M$EXTENDing the file. Should the message storage area become full, the comgroup is considered to be full (see below).

The second interaction concerning availability of space has to do with the active comgroup. The comgroup must have sufficient main memory to hold the station and message type trees, as well as a minimum of one MBLK per message queue. Full details are given in the subsection on performance considerations. In brief, however, the amount of memory consistent with efficient operation fluctuates with the level of activity in the comgroup, since the CP-6 system will attempt to acquire memory for the above items and will resort to spilling MBLKs to disk if memory is not available. This has limited effect in the sense that the station and message type trees cannot be so spilled, and thus a situation may arise where the comgroup cannot obtain sufficient memory to operate; the comgroup is then considered to be full. In order to prevent this situation, and to allow the AU to ensure efficient operation of the comgroup if he so desires, a special warning message will be sent to the AU should the amount of potentially-available memory sink below a preselected level. The AU can then react by increasing the MAXPG parameter.

When a comgroup becomes full as a result of insufficient disk space or main memory, the following specific actions are taken. First, all TERMINAL stations are disassociated from the comgroup as though the comgroup had become inactive through the disconnection of its last DCB. Next, the comgroup control parameters are altered to disallow activity in the comgroup; specifically:

TERMCONLGL	is set to NO
DCBCONLGL	is set to NO
DCBCONAU	is set to NO
DCBCONNAU	is set to NO
TERMCONAU	is set to NO
TERMCONNAU	is set to NO
AUCONDCB	is set to DEACTIVATE
AUDCONDCB	is set to DEACTIVATE
AUCONTERM	is set to DEACTIVATE
AUDCONTERM	is set to DEACTIVATE

(see VLP_CGCP for the meanings of these parameters). Last, all DCB stations are deactivated, INCLUDING the AU's station. Thus, the AU is informed that the comgroup became full by his being deactivated.

When faced with this situation, the AU must first rectify the cause; that is, he must allocate more space in the comgroup file and/or increase the value of MAXPG. This is possible since neither M$EXTEND nor M$CGCTL requires that his AU station be activated. Then, DCB stations may be selectively re-activated, including the AU's own station (ncte that M$ACTIVATE also may be issued when the AU is deactivated). Finally, TERMINAL stations which are in 'limbo' may be re-connected to the comgroup by issuing M$CGCTL setting TERMCONLGL to YES again. Naturally, if the comgroup has no AU it simply stops operation until such time as an AU joins and takes appropriate action.

PERFORMANCE CONSIDERATIONS

The performance level of a comgroup is indirectly controlled by the AU, or more exactly by the control parameters he establishes. The main factor here is the amount of main memory available to the comgroup.

The system manager, via the PIG processor, establishes for each file management account a limit on the total main memory available to all comgroups residing in that account. Thus the comgroups in a file management account 'draw' their main memory from a common well. Next, each comgroup has a limit on the main memory it may acquire when active. This limit is established by the AU (MAXPG in VLP_CGCP). A comgroup will normally acquire memory only as needed, up to that limit. However, an AU may set a lower limit (MINPG in VLP_CGCP) on his comgroup's main memory utilization, thus forcing CP-6 file management to allocate that amount to the comgroup the instant it becomes active. This guarantees that the comgroup will have at least that much memory available. This feature is useful because of course there is no guarantee that a given comgroup will be able to acquire memory when it needs to (due to the account limit).

A comgroup's requirements for memory come from three sources: data cache, MBLK storage, and 'permanent' storage (the station and message-type trees). Memory is divided into two 'buckets':
data cache and everything else. The size of each of these 'buckets' is controlled by the AU through the MAXPG, MINPG, and DATAPGS parameters (see VLP_CGCP). The 'permanent' storage requirement is a fairly static one, since message types are rarely redefined and the number of stations is in most cases almost constant. The two former sources are different, however. The amount of memory required here varies according to the number and size of the messages in the comgroup, which can vary rapidly and widely. Thus the problem of managing the comgroup might seem to be an intractable one. However, the CP-6 system makes a great effort to smooth out such fluctuations.

In the first place, the data cache automatically spills to disk the data parts of messages not immediately required, and does so in an efficient manner. The CP-6 system permits the data cache to grow to the maximum size set by the AU, but releases memory from the cache when it is not required. Clearly, then, a comgroup will operate with a very small data cache, but will benefit from a larger one. The AU must balance the cost of additional disk I/O against the cost of memory residency in deciding what maximum he will establish for the data cache size. Note that cache I/O activity, as well as all other 'hidden' overhead in the operation of the comgroup, is charged to the comgroup file's creator.

In the second place, the CP-6 system will spill to disk the MBLK parts of messages if the comgroup cannot fit them into memory. The amount of space available for MBLK storage is the effective maximum size of the comgroup minus the size of the data cache, minus the space required for the 'permanent' items. A station tree node and message type tree node each require on the order of 40 words of memory, whereas an MBLK requires on the order of 16 words of memory. Thus the AU should be able to roughly calculate how many messages his comgroup will hold without having to resort to MBLK spillage. This may be important, since such spillage is a relatively expensive process, except in certain special cases. For example, it is inexpensive to spill to disk a queue of 10000 messages all destined for a station which is not connected to the comgroup. On the other hand, a connected DCB station which is reading its message queue and specifying a combination of STATION and MSGTYP will cause great overhead if its message queue is on disk, since every M$READ will cause multiple disk accesses. As a rough guide then, it may be stated that, if efficiency is desired, the comgroup should possess sufficient memory to hold all the messages queues that are being accessed frequently. Particular performance gains will be achieved by having sufficient memory to hold all the queues that are being accessed out of order, that is by MSGTYP and/or STATION as opposed to priority and time of arrival. This is because message queues are ordered in the latter way, thus that type of access simply involves picking the message at the head of the queue. When the CP-6 system is forced to resort to MBLK spillage, the first candidates for spillage are queues which are unlikely to be accessed, for example those for absent stations.

REBUILDING A COMGROUP

Given certain 'backed-up' information, an AU may in a certain sense, rebuild his comgroup. Beginning with a comgroup opened in create mode, that is to say an essentially empty comgroup, an AU may turn that comgroup into an almost perfect image of some other comgroup, with as many of the original comgroup's messages and attributes as he desires.

In order to accomplish this, two classes of 'backed-up' information are required. First, the AU must know all the AU-controlled parameters of the comgroup he wishes to recreate. Since all of this information in the original comgroup was supplied by the AU (for example the control parameters and list of legal message types), CP-6 file management makes no attempt to aid the AU in remembering this class of data. The second class of information is of course the actual messages present in the original comgroup. As may be expected, the problem of preserving this information breaks down into two parts, that of preserving the data, and the attributes, of these messages. CP-6 file management provides special capabilities which aid the AU in remembering and restoring this information.

We will consider first the problem of preserving the messages. For those messages which are to be preserved, the data and attributes must be accessed from the comgroup, and written to some private storage medium from which they will be restored when the comgroup is rebuilt. There are essentially two methods of approaching this problem. The first, referred to as journaling, involves saving each message as it appears for the first time, in other words when it is written into the comgroup. Since the message is not normally visible to the AU at this time, CP-6 file management makes it possible for the AU to intervene and 'see' the message before it is actually written to its destination. A detailed discussion of this process is given below. The second approach is that of deciding at some particular time that the contents of the comgroup are to be saved, and at that time accessing all of the selected messages and saving them. This process may be performed by use of the M$FWCG service. In brief, it involves using M$FWCG to re-direct the selected messages to the AU, who will then perform the 'journal' procedure described below, and subsequently re-direct the messages to their original destination. The re-direction is required since M$FWCG, while it can inform the caller of message attributes, cannot access message data. One of the prime features of this approach is that unlike the first method, it perturbs the operation of the comgroup in a way that is 'visible' to, and may have an effect on, other member stations. This is because messages are removed from their proper queues, if only momentarily. The first method does not have this effect, since the essential disjunction between member stations makes the 'window' during which a message is journaled (after being written but before being delivered to its destination) invisible to the stations involved.

The process of journaling operates in the following manner. Each message type has an attribute called JOURNAL, controlled by the AU. If JOURNAL is set to YES, then every message written with that type is to be journaled. This means that instead of being sent to its destination, the message will be directed to the station given by the AU as JRNLSTA in M$CGCTL. Note, however, that the destination attribute of the message is intact, just as for messages written to HOLDSTA as a result of M$UNLATCH HOLD. The designated station then reads the message with LATCH, and as a result obtains the message data. The message attributes are obtained by issuing an M$FWCG service specifying ONEATTR, and NIL for CRITERIA (this selects the currently-latched input message). The data and attributes together are then saved appropriately, whereupon the message is re-directed to its proper destination via M$UNLATCH RERUN. When the message is finally deleted from the comgroup, a JOURNAL DELETE message is generated and sent to the journal station. This message informs the journal station of the MSGID of the message that has just been deleted. Note that none of the steps of this process require the journal station to be the AU; it need only have AURD access to the comgroup.

We arrive now at the problem of re-structuring the newly-opened comgroup to be a copy of the original. First, the comgroup attributes and control parameters must be re-established via M$CGCTL. The information required here may have been saved in a variety of ways, according to the whim of the particular AU. Next and last, the saved messages must be restored into the comgroup. This process MUST be performed by the AU, since it involves using the operational form of M$FWCG. There are in fact a variety of ways to do this, but the simplest is given here. Each saved message is retrieved, both data and attributes. The AU then writes the message data into the comgroup, specifying any message type he desires, and himself as destination. The completion of this write operation will return the MSGID, say X, in the new comgroup of the message just created. The AU then issues an M$FWCG call specifying the following criteria:

```
DEST=himself
QSELECT=DIRECT
MSGID=X
```

The REATTR parameter frames the saved message attributes, and SRA frames a VLP_SCGMA with all the options set to YES. The effect of this is to restore to the message all of its original attributes, including MSGID and destination. After performing this process on all of the saved messages, the new comgroup will contain all of the saved messages, each in its original state.

A few caveats are in order. First, in order to avoid duplicate MSGIDs in the new comgroup, the AU should ensure that he is the only station connected to the comgroup while it is being rebuilt (see also MSGID in VLP_SCGMA). Second, note that if the saved messages were obtained by the journal method, then the special AU event messages will not be saved, inasmuch as their message type, '*AUEV', never appears in the message type list and thus cannot have the JOURNAL attribute attached to it.

B$CGAURD

The B$CGAURD macro may be used to generate a based structure to receive special event messages given to the administrative user (AU). Special events occurring within a comgroup are reported when the AU reads through the AU DCB. Regardless of the qualifying parameters specified on the call to M$READ, the read will be satisfied by one of these special 'event' messages if there is one. Such a message is identified by the message type, which is '*AUEV'. This structure is used to frame the data returned into the buffer specified on M$READ.

EVENT contains the code identifying which type of event this is. See %CG_XXXX# defined by the macro EQU_CG.

INFO is one word of information specific to this event. For example, it may contain the type of break (control-Y, BREAK, input ready) for the CG_TBRK# event. Some events supply more information. In these cases, the extra information follows INFO in the user's buffer. The format of this information will be specific to the type of event. By setting LAST="," and supplying appropriate following structure, B$CGAURD may be made to frame the entire buffer.

Following is a list of the event codes and corresponding to each the structure which frames the remainder of the message:

CG_TCON#	B$CGAUCI
CG_DOPN#	B$CGAUCI
CG_UNLR#	VLP_STATION
CG_UNLH#	VLP_STATION
CG_UNLD#	VLP_STATION
	(for these, the VLP_STATION describes the UNLATCHed message in the same manner as the VLP_STATION in a

	DCB following an M$READ describes the message just read).
CG_FWCG#	See INFORM in FPT_FWCG
CG_FPLDTRC#	VLP_FPRG

MID is meaningful only when EVENT is CG_JNLD#, i.e. when the event is 'journal delete'. MID contains the message ID of the message whose deletion is being reported.

MID.MSGID = VALUE-DEC (1-?) is the primary part of the message ID.

MID.MSGIDXT = VALUE-DEC (1-?) is the extension part of the message ID.

STATION = VALUE-CHAR(8) is the name of the station (if any) associated with this event.

EQU_CG

The EQU_CG macro generates EQUs which are of use in dealing with comgroups, particularly for an AU. A list of the symbols thus defined follows.

The codes beginning with simply CG_ are for the AU event messages generated by the comgroup. These codes are reported in B$CGAURD.EVENT, and are:

CG_DOPN#	DCB open
CG_DCLS#	DCB close
CG_TCON#	TERMINAL connect
CG_TDRQ#	TERMINAL disconnect request
CG_TDSC#	TERMINAL disconnect
CG_TBRK#	TERMINAL BREAK
CG_UNLR#	UNLATCH 'rerun' performed
CG_UNLH#	UNLATCH 'hold' performed
CG_UNLD#	UNLATCH 'delete' performed
CG_WARN#	Comgroup is low on space
CG_FWCG#	Message generated by M$FWCG
CG_JNLD#	Journal delete message
CG_FPOPERR#	Error on FPRG activate
CG_FPEXIT#	FPRG issued an M$EXIT
CG_FPABORT#	FPRG aborted
CG_FPLDTRC#	FPRG issued an M$LDTRC successfully

The codes beginning with CG_WARN_ identify variations of the CG_WARN event message, and are reported in B$CGAURD.INFO :

CG_WARN_MEM#	Available memory is low
CG_WARN_DISK#	Disk free space is low

The codes beginning with CG_TBRK_ identify variations of the CG_TBRK event message, and are reported in B$CGAURD.INFO :

CG_TBRK_BRK#	BREAK
CG_TBRK_EC#	Control-Y
CG_TBRK_RDY#	Input ready (card reader)

The codes beginning with CG_UNL_ identify variations of the CG_UNLX event messages, and are reported in B$CGAURD.INFO :

CG_UNL_USER#	UNLATCH performed by user
CG_UNL_RCVR#	UNLATCH performed by comgroup recovery after system crash.

The codes beginning with CG_CTLFC_ are the options for TYPFC and STAFC of M$CGCTL :

CG_CTLFC_CHGATTR#	Change attributes
CG_CTLFC_ADD#	Add
CG_CTLFC_DELETE#	Delete
CG_CTLFC_SETLIST#	Set list

The codes beginning with CG_INFFC_ are the options for TYPFC and STAFC of M$CGINFO :

CG_INFFC_SELECT#	Select
CG_INFFC_DUMP#	Dump
CG_INFFC_WILD#	Wild

The codes beginning with CG_QSELECT_ are the options for QSELECT of M$FWCG:

CG_QSELECT_ALL#	ALL
CG_QSELECT_DIRECT#	DIRECT
CG_QSELECT_ANONYMOUS#	ANONYMOUS

B$CGAUCI

The B$CGAUCI macro may be used to generate a based structure to receive information regarding DCB and TERMINAL connects for a comgroup administrative user (AU) (for the CG_TCON# and CG_DOPN# events). This information follows B$CGAURD in the buffer, and is framed by B$CGAUCI.

ATTR is the mask of device attribute bits (TERMs only).

FLG contains various flags.

FLG.INPUT = VALUE-BIT(1) indicates input is legal on this device (TERMs only).

FLG.OUTPUT = VALUE-BIT(1) indicates output is legal on this device (TERMs only).

LENGTH = VALUE-DEC(0-?) is the logical page length for output (TERMs only).

NWINFO = VALUE-DEC(0-511) is the number of of words of extra INFO following B$CGAUCI. There may also be extra words of information specific to the individual comgroup. NWINFO contains the number of words present, and the individual AU may attach his own structure to frame the information. The structure B$CGAUINFO may be used directly or as a template for such a structure.

SYSID is the SYSID of the user who owns the connecting DCB (DCBs only).

WHO provides identification of the connecting entity.

WHO.ACCT = VALUE-CHAR(8) is the logon ACCOUNT for DCBs and TERMS.

WHO.NAME = VALUE-CHAR(12) is the logon NAME for DCBs and TERMs.

WIDTH = VALUE-DEC(0-?) is the logical line width for output (TERMs only).

B$CGAUINFO

The B$CGAUINFO macro may be used to generate a based structure to receive extra information passed to the administrative user (AU) of a comgroup when a TERMINAL connects. The system manager may supply extra TERMINAL connect information to SUPER. This is its default structure.

CMD = VALUE-CHAR(80) is the text of the command described in UCMDCNT.

UCMDCNT = VALUE-DEC(0-80) is the length in bytes of a command supplied for this device by the system manager using SUPER. If the command is present (UCMDCNT~=0) it is only meaningful to the receiving program.

USER1 = VALUE-DEC(0-?) is a word of information supplied by the system manager to SUPER when the device is authorized. Its value is meaningful only to the receiving program.

USER2 = VALUE-DEC(0-?) (See USER1.)

USER3 = VALUE-DEC(0-?) (See USER1.)

USER4 = VALUE-DEC(0-?) (See USER1.)

B$CGAUCRD

The B$CGAUCRD macro provides a complete structure for comgroup administrative user (AU) connect events. B$CGAURD, B$CGAUCI, and B$CGAUINFO are invoked by this one macro.

M$ACTIVATE - Activate a Comgroup Station

The M$ACTIVATE service is used by the administrative user (AU) of a comgroup to allow other users and terminals access to his comgroup. He can also set and/or change various parameters controlling their access to the comgroup. FPRG terminal stations may be created using the MAKEFPRG option.

Automatic activation of a station by the comgroup, for example at AU connect (see AUCONDCB in VLP_CGCP), acts like M$ACTIVATE with the following options:

```
ABTREAD    = NONE
ABTET      = NO
FLUSH      = NO
```

except that the station's DVBYTE is not changed.

The M$ACTIVATE procedure call is of the form:

CALL M$ACTIVATE(FPT_ACTIVATE) [ALTRET (label)];

The parameters for this service are explained in the discussion of M$DEACTIVATE.

M$DEACTIVATE - Deactivate a Comgroup Station

The M$DEACTIVATE service is used by the administrative user (AU) of a comgroup to disconnect its attached terminals or FPRG terminals or to remove access from other users or terminals. In the case of the latter, various parameters controlling their access to the comgroup when they are M$ACTIVATEd again may be set or changed.

Automatic deactivation of a station by the comgroup, for example at AU connect (see AUCONDCB in VLP_CGCP), acts like M$DEACTIVATE with the following options:

```
FLUSH      = NO
DISCONNECT = NO
DSCGO      = NO
```

The form of the call for this service is:

CALL M$DEACTIVATE (FPT_ACTIVATE)   [ALTRET (label)];

The parameters are as follows:

ABTET = {YES|NO} specifies whether (YES) or not (NO) the TYC value specified in ABTREAD is to be permanent. If YES, then the target station will receive that error on every read until it is re-activated. If NO, then the error will be passed only on the next or pending read; subsequent reads will receive no error.

ABTET applies only when ABTREAD is set. Note that M$DEACTIVATE, or automatic activation or deactivation by the comgroup, will clear the ABTET set by M$ACTIVATE.

The default is NO.

ABTREAD = VALUE-BIT(36) applies only to M$ACTIVATE when directed at a DCB. The default is none. If ABTREAD is set, then the target DCB's next or pending read is aborted with the TYC value supplied as the ABTREAD value.

Note that if ABTREAD is not set, then M$ACTIVATE will clear any previous ABTREAD value which has not yet been honored, that is to say if a station is M$ACTIVATEd with ABTREAD, and has no pending read, then if it is M$ACTIVATEd with ABTREAD not set before it issues its next read, the first ABTREAD has no effect. Furthermore, M$DEACTIVATE, or automatic activation or deactivation by the comgroup, will reset the station's ABTREAD.

See also ABTET. The default is none.

ATTR = VALUE-BIT(18) is a mask of required attributes for the resource that is to be connected to the comgroup. The default is '0'B.

BIN = {YES|NO} BIN = YES specifies that the data is to be read as a string of bits rather than one character per 9-bit byte. The FPT field name for this parameter is FPT_ACTIVATE.V.DVBYTE.BIN#. The Default is NO.

CONNECTRES = {YES|NO} applies only to M$ACTIVATE. If CONNECTRES is set, the resource specified by RES and ATTR in this FPT will be connected to the comgroup with the station name specified in the VLP_STATION. Default is NO.

DCB = DCBNAME specifies the DCB associated with the comgroup.

DISCONNECT = {YES|NO} specifies whether (YES) or not (NO) M$DEACTIVATE is to disconnect the target station. DISCONNECT applies only if the target station is a TERMINAL. DISCONNECT is forced to YES for M$DEACTIVATEs of pure FPRG terminal stations. Deactivation of transformed FPRG terminal stations with DISCONNECT=NO causes the station to revert to a deactivated normal terminal station.

The default is NO.

DSCGO = {YES|NO} indicates whether (YES) or not (NO) disconnect permission is being given. DSCGO is only meaningful for M$DEACTIVATE directed at a TERMINAL station, and then only when the station has requested permission to disconnect (see DSCREQ in VLP_CGCP). DSCGO is only honored if DISCONNECT is NO.

The default is NO.

DVBYTE is a collection of bits controlling the manner in which TERMINALs are read. That is, they supply parameters controlling the way in which TERMINALs supply data to the comgroup. Each bit is specified separately. DVBYTE is meaningful only for M$ACTIVATE directed at a TERMINAL. Note that when DVBYTE is meaningful, it always take effect, in other words it is not possible to M$ACTIVATE a TERMINAL station without setting its DVBYTE.

FLUSH = {YES|NO} specifies whether or not to flush the list of messages waiting to go to the target station. Applies to M$ACTIVATE and M$DEACTIVATE. Note that if the target station is not connected, then M$ACTIVATE will return an error, but M$DEACTIVATE will not. Thus, messages waiting for a non-connected station may be FLUSHed by M$DEACTIVATE. The default is NO.

FPRGSTA = VARIABLE locates a VLP_STATION area which supplies the station name to be given a pure FPRG terminal station. FPRGSTA applies only to MAKEFPRG=YES M$ACTIVATEs. See the MAKEFPRG option for additional information.

The default is NIL.

MAKEFPRG = {YES|NO} applies only to M$ACTIVATE when directed at non-FPRG terminal stations. If MAKEFPRG is set, a single non-FPRG terminal station is specified and FPRGSTA specifies a non-blank station name, then a new pure FPRG terminal station is created obtaining its station name from FPRGSTA. If, on the other hand, MAKEFPRG is set but the rest of the above is not true, then every specified non-FPRG terminal station will be transformed into a transformed FPRG terminal station.

The default is NO.

REREAD = {YES|NO} REREAD = YES specifies that the previous input line is to be echoed and set to be the current input line. The FPT field name for this parameter is FPT_ACTIVATE.V.DVBYTE.REREAD#. The Default is NO.

RES = VALUE-CHAR(4) specifies the four character resource name that is to be connected to the comgroup. The resource had to be acquired previously. The default is blanks.

STATION = VARIABLE locates a VLP_STATION area which specifies the target station(s), and must be supplied.

The station name may be wild-carded in the way described under STATION in VLP_STATION. The action(s) specified are performed on all stations whose names qualify under the wild-card, subject to the restrictions specified by ALLABSENT, ALLDCBS, and ALLTERMS.

M$ACTIVATE ignores any station that is not connected. M$DEACTIVATE may (subject to the above-mentioned control parameters) process absent stations, in which case no deactivation is performed but options such as FLUSH will take effect.

If no appropriate station(s) is(are) found, then an error is returned.

The default is NIL.

TRANS = {YES|NO} TRANS = YES specifies transparency, (i.e., that data is to be read without translation). This option is useful primarily for devices which normally perform translation (for instance, asynchronous terminals). The FPT field name for this parameter is FPT_ACTIVATE.V.DVBYTE.TRANS#. The default is NO.

M$CGCTL - Comgroup Control

The M$CGCTL service provides a comgroup administrative user (AU) with the capability to change the control parameters of his comgroup. When a comgroup is created, all of the parameters changeable via M$CGCTL are set to default values. At any time thereafter, the AU may change these by issuing M$CGCTL. In general, changes requested through this service take effect immediately and will be visible to stations connected to the comgroup at the time M$CGCTL is issued.

The various parameters are processed in the following order:
CGCP, TYPLIST, STALIST, SETSTA. This can be important since there is some interaction between the options (e.g. one cannot delete a default message type; if a single service request sets CGCP and specifies a list of message types to be deleted, the default types specified in the former cannot be specified in the latter).

The form of the call for this service is:

CALL M$CGCTL (FPT_CGCTL) [ALTRET (label)];

The parameters for this service are as follows:

ATTRS specifies the attributes to be changed by CHGATTR.

ATTRS.AUP = {YES|NO} specifies whether or not the AUP attributes of the stations are to be changed. The values will be taken from LIST.AUP# in VLP_CGSTAL. The default is NO.

ATTRS.JOURNAL = {YES|NO} specifies whether or not the JOURNAL attributes of the message types are to be changed. The values will be taken from LIST.JOURNAL# in VLP_CGTYPL. The default is NO.

ATTRS.MXACT = {YES|NO} specifies whether or not the maximum-active limits for the queue nodes are to be changed. The values will be taken from LIST.MXACT# in VLP_CGQL. The default is NO.

ATTRS.ONEREPORT = {YES|NO} specifies whether or not the ONEREPORT attributes of the message types are to be changed. The values will be taken from LIST.ONEREPORT# in VLP_CGTYPL. The default is NO.

ATTRS.PERM = {YES|NO} specifies whether or not the PERM attributes of the message types and/or stations are to be changed. The values will be taken from LIST.PERM# in VLP_CGTYPL and/or VLP_CGSTAL. The default is NO.

ATTRS.PRIO = {YES|NO} specifies whether or not the priorities of the message types and/or stations are to be changed. The values will be taken from LIST.PRIO# in VLP_CGTYPL and/or VLP_CGSTAL. The default is NO.

CGCP = VARIABLE locates a VLP_CGCP area. All of the comgroup control parameters will be changed to reflect the values contained in the VLP_CGCP area.

DCB = DCBNAME specifies the DCB associated with the comgroup.

HOLDSTA = VARIABLE locates a VLP_STATION area which names the DCB station to which messages placed in 'hold' status via M$UNLATCH will be written.

The station designated here will become a 'special' station. A special station has the following attributes:

- It is created by the M$CGCTL service call which defines it.
- Once a special station has been defined for a special purpose such as HOLDSTA, it remains defined until changed by M$CGCTL. At that time, the old special station has its 'special' attribute removed. Until then, however, the special station may not be deleted by means of M$CGCTL SETLIST or DELETE. Furthermore, special stations are treated as though their PERM attributes were YES (see VLP_CGSTAL).
- A terminal may not join a comgroup under a special station name.
- Once a special station has been defined, issuing M$CGCTL and specifying NIL for the corresponding parameter, or specifying the parameter with a STATION# of blanks, have no effect.
- Once a special station has been defined, issuing M$CGCTL and specifying ERASE for the corresponding parameter will cause the comgroup to no longer possess the special attribute, and the actual station previously defined will lose its special status.
- When first created, a comgroup has no special stations of any kind.

M$UNLATCH HOLD services cause the unlatched message to be written to the station specified by HOLDSTA without any of their original parameters being changed, and may subsequently by read by that station.

If no HOLDSTA has ever been specified, or after it has been ERASEd, messages placed in 'hold' are simply deleted.

The default is NIL.

JRNLFID = VARIABLE locates a VLP_JRNLFID area.

A comgroup may or may not possess a journal FID which specifies the file to which journal records are to be written by the journal station. When a comgroup is first created, it has none.

Issuing M$CGCTL with JRNLFID framing a VLP_JRNLFID will cause the comgroup's journal FID to be set to the one specified.

Issuing M$CGCTL with JRNLFID = ERASE causes the comgroup to have no journal FID.

Issuing M$CGCTL with JRNLFID = NIL will cause no change.

The M$CGINFO service may be used to retrieve the comgroup's journal FID. If the comgroup has none, then VLP_JRNLFID.L# is returned as zero. This should be construed by the journaling process to mean that the comgroup is not to be journaled.

If the comgroup is being journaled by the system JOURNAL GHOST (see GHSTACS in M$DCB), then the ghost will be informed whenever the comgroup's journal FID is changed or ERASEd. If the FID is being changed, then the ghost will attempt to start journaling to the newly-specified file. Any error encountered by the ghost in attempting to open the new journal file will be passed back as an error to the M$CGCTL service caller, and journaling will stop. Note that in this case the comgroup's journal FID will be left set the to newly-specified value.

The default for JRNLFID is NIL.

JRNLSTA = VARIABLE locates a VLP_STATION area which names the DCB station to which messages whose type has the JOURNAL attribute are to be written, and to which journal delete messages are to be written. See JOURNAL in VLP_CGTYPL.

The station designated here will become a 'special' station (see HOLDSTA).

This writing of the message to the station specified here will occur at the time when the message would otherwise be sent to its destination, and takes place with the message's destination attribute intact. Thus if the station named here reads such a message with LATCH, and then performs an M$UNLATCH RERUN, the message will be sent to its intended destination, with neither the originator nor the reader being aware of the detour. The default for JRNLSTA is NIL, and indicates simply that the 'journal detour' is not to be performed regardless of the JOURNAL attribute of any message type.

Should the station designated here be present on the comgroup and then disconnect, then independent of WAS (see VLP_CGCP), the journal detour will be performed despite the absence of the journal station, with the to-be-journaled messages awaiting the arrival of the journal station. Messages waiting at the journal station are retained independent of SMD (see VLP_CGCP).

If the comgroup is to be journaled by the system JOURNAL GHOST (see GHSTACS in M$DCB), then the name of the journal station MUST be JRNLSTA (this is required in order that the ghost know what station name to open with). When the ghost attempts to open the comgroup and asks to be station 'JRNLSTA', then the open will fail unless the comgroup has a special journal station of that name. Note that the JOURNAL GHOST will be informed if JRNLSTA is changed.

The JRNLFID parameter specifies which journal file is to be used.

The default is NIL.

QLIST = VARIABLE locates a VLP_CGQL area. This contains a list of attributes for the anonymous queue nodes. This list is parallel to the VLP_CGTYPL or VLP_CGSTAL list (depending upon which way the queue is keyed), and is used by the ADD and CHGATTR functions.

SETSTA = VARIABLE locates a VLP_SETSTA area which is to be supplied to any DCB which opens to the comgroup. Note that the administrative user (AU) will receive this same VLP_SETSTA when opening his DCB to the comgroup, but that he is exempt from lock violation errors that may result, whereas other users are not.

STAFC = OPTION indicates the function of the station list (if present). The comgroup maintains a list of known stations, each with associated control parameters. This list, if XSTALGL is reset (see VLP_CGCP), constitutes the totality of legal stations. On the other hand, if XSTALGL is set, it simply provides a way for the AU to attach different attributes to different stations. When XSTALGL is set and a previously unknown station is written to, an entry for that station is created with the default station priority (see VLP_CGCP. STAPRIO). The STAFC options are as follows:

STAFC = ADD requests that any previously unknown stations listed in VLP_CGTYPL, are to be added to the comgroup's list. All stations listed will have all of their attributes changed. The attribute values are taken from the fields specified in VLP_CGSTAL and, if the queue is keyed by station, from VLP_CGQL. Note that if QLIST is not specified and the queue is keyed by station, then default values are used.

STAFC = DELETE requests that each station listed is to be deleted from the comgroup's list. Note that if the queue is keyed by station, then queued messages from the listed stations will be deleted. Messages written to the listed stations will also be deleted.

STAFC = CHGATTR requests that each station listed have the attributes specified in ATTRS set as provided. Note that some of these attributes apply only if the queue is keyed by station, and are to be provided in QLIST (which locates a VLP_CGQL).

STAFC = SETLIST requests that the list of legal stations be set to the one provided. That is, the given stations are ADDed, then any not present in the given list are DELETEd.

The default is ADD.

STALIST = VARIABLE locates a VLP_CGSTAL area. This contains a list of stations to be used according to the value of STAFC. STALIST must frame the exact number of message types passed in via VLP_CGSTAL; VLP_CGSTAL.FOUND# is not used for M$CGCTL.

TYPFC = OPTION indicates the function of the message type list (if present). The comgroup maintains a list of known message types, each with associated control parameters. This list, if XTYPLGL is reset (see VLP_CGCP), constitutes the totality of legal message types. On the other hand, if XTYPLGL is set, it simply provides a way for the AU to attach different attributes to different message types. When XTYPLGL is set and a previously unknown message type is used, an entry for that type is created with the default message type priority (see VLP_CGCP.MTPRIO). The TYPFC options are as follows:

TYPFC = ADD requests that any previously unknown message types listed in VLP_CGTYPL, are to be added to the comgroup's list. All listed types are to have all of their attributes changed. The attribute values are taken from the fields specified in VLP_CGTYPL and, if the queue is keyed by message type, from VLP_CGQL. Note that if QLIST is not specified and the queue is keyed by type, then default values are used.

TYPFC = DELETE requests that each type listed in VLP_CGTYPL is to be deleted from the comgroup's list. Note that if the queue is keyed by message type, then queued messages of the listed types will be deleted.

TYPFC = CHGATTR requests that each type listed have the attributes specified in ATTRS set as provided. Note that some of these attributes apply only if the queue is keyed by message type, and are to be provided in QLIST (which locates a VLP_CGQL).

TYPFC = SETLIST requests that the list of legal types be set to the one provided. That is, the given types are ADDed, then any not present in the given list are DELETEd.

The default is ADD.

TYPLIST = VARIABLE locates a VLP_CGTYPL area. This contains a list of message types to be used according to the value of TYPFC. TYPLIST must frame the exact number of message types passed in via VLP_CGTYPL; VLP_CGTYPL.FOUND# is not used for M$CGCTL.

M$CGINFO - Comgroup Information

The M$CGINFO service provides a comgroup administrative user (AU) with the capability to gather information about the status of his comgroup.

The form of the call for this service is:

CALL M$CGINFO (FPT_CGINFO) [ALTRET (label)];

The parameters for this service are as follows:

CGCP = VARIABLE locates a VLP_CGCP area which will receive the current values of the comgroup control parameters.

The default is NIL.

DCB = DCBNAME specifies the DCB associated with the comgroup.

HOLDSTA = VARIABLE locates a VLP_STATION area which is to receive the current value of the comgroup's HOLDSTA (see FPT_CGCTL).

If the comgroup does not possess a defined HOLDSTA then the value returned will be the same as that generated by the default VLP_STATION (in particular the STATION part of the VLP_STATION will be the default value for STATION in VLP_STATION).

The default is NIL.

JRNLFID = VARIABLE locates a VLP_JRNLFID area which is to receive the comgroup's current journal FID.

If the comgroup does not possess a journal FID, then the L field of the framed VLP_JRNLFID will be set to zero.

The default is NIL.

JRNLSTA = VARIABLE locates a VLP_STATION area which is to receive the current value of the comgroup's JRNLSTA (see FPT_CGCTL).

If the comgroup does not possess a defined JRNLSTA then the value returned will be the same as that generated by the default VLP_STATION (in particular the STATION part of the VLP_STATION will be the default value for STATION in VLP_STATION).

The default is NIL.

QLIST = VARIABLE locates a VLP_CGQL area. This area receives information about the comgroup queue nodes. This list is parallel to the VLP_CGTYPL or VLP_CGSTAL list (depending upon which way the queue is keyed).

The default is NIL.

SETSTA=VARIABLE. Locates a VLP_SETSTA area. This area will receive the current VLP_SETSTA established by M$CGCTL.

The default is NIL.

STAFC = OPTION indicates the function of the station list (if present).

The service will always return a list of known stations, as well as parameters and statistics about each station. Also, if the queue is keyed by station, and QLIST is specified, then queue specific information about each station will be returned in the corresponding VLP_CGCL entries. STAFC selects how this is to be done.

STAFC = DUMP requests that all known stations are to be returned. In this case, if the number of stations exceeds the length of VLP_CGSTAL, then an error will be returned after filling all the available area. In any case, VLP_CGSTAL.FOUND will be set to the number of stations returned.

STAFC = SELECT indicates that VLP_CGSTAL contains a selected list of stations, which are named in VLP_CGSTAL.LIST.NAME. Only information about these stations is to be returned. If one of the selected stations does not exist, an error is returned.

STAFC = WILD indicates that VLP_CGSTAL contains a single station, which is named VLP_CGSTAL.LIST.NAME(0). This NAME contains a trailing '?' character. Only information about stations matching this wildcarded input station is to be returned. If the number of stations exceeds the length of VLP_CGSTAL, then an error will be returned after filling all the available area. The default is SELECT.

STALIST = VARIABLE locates a VLP_CGSTAL area which will receive information about the currently known stations. (See STAFC.)

The default is NIL.

STATS = VARIABLE locates a VLR_CGSTATS area. This area receives global comgroup statistics. (See VLR_CGSTATS.)

The default is NIL.

TYPFC = OPTION indicates the function of the message type list (if present).

The service will always return a list of known message types, as well as parameters and statistics about each station. Also, if the queue is keyed by message type, and QLIST is specified, then queue specific information about each type will be returned in the corresponding VLP_CGQL entries. TYPFC selects how this is to be done.

TYPFC = DUMP requests that all known message types are to be returned. In this case, if the number of types exceeds the length of VLP_CGTYPL, then an error will be returned after filling all the available area. In any case, VLP_CGTYPL.FOUND will be set to the number of types returned.

TYPFC = SELECT indicates that VLP_CGTYPL contains a selected list of types, which are named in VLP_CGTYPL.LIST.NAME. Only information about these types is to be returned. If one of the selected types does not exist, an error is returned.

TYPFC = WILD indicates that VLP_CGTYPL contains a single type, which is named VLP_CGTYPL.LIST.NAME(0). This NAME contains a trailing '?' character. Only information about types matching this wildcarded input type is to be returned. If the number of types exceeds the length of VLP_CGTYPL, then an error will be returned after filling all the available area. The default is SELECT.

TYPLIST = VARIABLE locates a VLP_CGTYPL area which will receive information about the currently known message types. (See TYPFC.)

The default is NIL.

M$UNLATCH - Unlatch Comgroup Messages

The M$UNLATCH service is used to control the status of latched messages read or written with LATCH=YES through a comgroup DCB. The last output message may be deleted or sent to its destination. An input message may be deleted, put back into the queue to be re-read, or put back in the queue in 'hold' status such that it is inaccessible until action by the administrative user.

The form of the call for this service is:

CALL M$UNLATCH (FPT_UNLATCH) [ALTRET (label)];

The parameters are as follows:

DCB = DCBNAME specifies the DCB associated with the comgroup.

INPUT = OPTION specifies the action to be taken in respect to an input latched to this DCB. The options are:

- o DELETE - Processing complete, delete it.
- o RERUN - Put it back in the queue to be re-read and re-processed.
- o HOLD - Some error occurred; put it back in the queue inactive to be examined later by the administrative user (AU).

If INPUT is not specified, the OUTPUT option operates only on the last message written through this DCB.

JOURNALED = {YES|NO} indicates, if RERUN is specified, whether (YES) or not (NO), the input message is to be released for processing because it has been journaled. If JOURNALED is YES then the message will go to its original intended destination. If NO, the message will be requeued for the journal station.

JOURNALED is only meaningful when M$UNLATCH is being issued by the comgroup journal station.

The default is NO.

OUTPUT = OPTION specifies the action to be taken with respect to output(s) latched to this DCB. If the INPUT option is specified, the OUTPUT option refers to all outputs latched to the input. Otherwise it refers to the last message written through this DCB. The options are:

- o DELETE - An error occurred, delete the output(s).
- o SEND - Everything ok, send the output(s) to destination.

The default is DELETE.

TELLAU = {YES|NO} specifies whether or not to send a special unlatch message to the administrative user of the comgroup. If this option is specified then it is always honored. If not specified then the default depends on the type of unlatch being performed. For HOLD the default is YES otherwise the default is NO.

NOTE: The administrative user (AU) can choose to not receive any such messages by specifying UNLAMSG = NO in VLP_CGCP on M$CGCTL.

TELLOS = {YES|NO} specifies whether or not to send a response message to the station which originated the message being unlatched. This message will have message type '*RESPX' and no content. For HOLD the default is YES otherwise the default is NO.

VLP_CGCP

The VLP_CGCP macro creates an area containing the control parameters for a comgroup.

AUCONDCB = OPTION specifies the action to be taken for all connected DCBs when the AU connects. The applicable options are NC, ACTIVATE and DEACTIVATE which are described under AUCONTERM. If DISCONNECT is specified, it is treated like DEACTIVATE. The default is NC.

AUCONTERM = OPTION specifies the action to be taken for all connected terminals when the AU connects.

ACTIVATE	Activate all connected terminals that are not already activated.
DEACTIVATE	Deactivate all terminals that are currently activated.
DISCONNECT	Disconnect all connected terminals.
NC	No change - leave in current state.

The default is NC.

AUDCONDCB = OPTION specifies the action to be taken for all connected DCBs when the AU disconnects. The applicable options are NC, ACTIVATE and DEACTIVATE which are described under AUCONTERM. If DISCONNECT is specified, it is treated like DEACTIVATE. The default is NC.

AUDCONTERM = OPTION specifies the action to be taken for all connected terminals when the AU disconnects. The applicable options are described under AUCONTERM. The default is NC.

AUTORCVR = {YES|NO} establishes the default AUTORCVR value for queue nodes not established by M$CGCTL. (See LIST.AUTORCVR in VLP_CGQL.) The default is YES.

BIGMXT = {YES|NO} indicates whether (YES) or not (NO), message ID extensions are to take the 'large' form. BIGMXT = YES is used in a comgroup wherein DCB stations need to latch more that 62 output messages to a single input message, or more that 6 levels of spawning of input messages are required. Refer to the discussion of MSGID and MSGIDXT in VLP_STATION. The default is NO.

BIN = {YES|NO} BIN = YES specifies that the data is to be read as a string of bits rather than one character per 9-bit byte. The FPT field name for this parameter is VLP_CGCP.DVBYTE.BIN#. The default is NO.

CARRYOSTA = {YES|NO}. YES specifies that the origin station of the current latched read be used as the origin station for all latched DCB writes into the anonymous queue. The contribution to priority by a station comes from the current latched read also. CARRYOSTA = NO specifies that the origin station name and the station's contribution to priority come from the current writer's station. The default is NO.

CONMSG = {YES|NO} CONMSG = YES indicates that the administrative user (AU) is to receive messages when stations connect and disconnect.

When an AU joins a comgroup in which CONMSG is YES, a series of connect event messages is generated and sent to him, one for each station that is currently connected.

Note that if CONMSG is NO, and is set to YES via M$CGCTL, such a series of messages is not generated when the M$CGCTL is issued.

If CONMSG is NO, then DSCREQ and DCBCONWA are forced to NO.

The default is YES.

DATAPGS = VALUE-DEC(1-99) is the percent of physical memory pages (MAXPG,MINPG) that are to be allocated to the data cache. The range of values that can be produced for the maximum on data pages is 2 through MAXPG-1. Similarly, the range of values that can be produced for the minimum on data pages is 0 through MAXPG-1. The default is 50.

DCBCONAU = {YES|NO} DCBCONAU = YES indicates that DCBs are to be activated upon connection if the AU is connected. The default is YES.

DCBCONLGL = {YES|NO} specifies whether or not DCBs are allowed to open to this comgroup. If NO, the AU will be exempted from this restriction. The default is YES.

DCBCONNAU = {YES|NO} DCBCONNAU = YES indicates that DCBs are to be activated upon connection if the AU is not connected. The default is YES.

DCBCONWA = {YES|NO} indicates whether (YES) or not (NO), DCB stations are to wait for activation upon joining the comgroup.

If YES, then the M$OPEN process involves waiting for the AU to read the DCB connect event message and respond with M$ACTIVATE or M$DEACTIVATE. If the response is ACTIVATE, then the M$OPEN proceeds and the DCB station joins the comgroup.

If the response is DEACTIVATE, then the M$OPEN fails with an appropriate error. Thus from the point of the user the M$OPEN has failed; from the point of the comgroup, however, the DCB station joins the comgroup and then immediately disconnects. This means that the AU will see the DCB connect, respond by deactivating the station, and then see the DCB disconnect.

DCBCONWA = YES is used by an AU who wishes to to be able to prevent DCBs from opening to his comgroup. It is also used to prevent user programs from opening to the comgroup and receiving the 'deactivated' error when they read or write before the AU can respond to the DCB open message.

DCBCONWA is only meaningful when DCBCONAU is NO, and is only acted upon when the AU is connected to the comgroup. In particular, a user awaiting AU response as a result of DCBCONWA being YES will proceed with his M$OPEN if the AU disconnects from the comgroup. This works as follows. If AUDCONDCB is ACTIVATE or NC, the M$OPEN succeeds; if AUDCONDCB is DEACTIVATE, the M$OPEN fails just as though the AU had M$DEACTIVATEd the station.

Note that the wait in M$OPEN is terminated by ANY activation or deactivation. In particular, then, if the comgroup becomes full, the automatic deactivation that results will terminate the M$OPEN abnormally.

The wait in M$OPEN is bypassed under any of the following circumstances:

- o The DCB is being opened by the user who owns the AU DCB. In this case the joining station will NOT be activated (since the AU is present and DCBCONAU, the relevant parameter, is NO).

- o The DCB is being opened by the system JOURNAL ghost job, under the station name specified in JRNLSTA (see FPT_CGCTL). In this case the joining station will be activated unconditionally.

DCBCONWA is forced NO if CONMSG is NO.

The default is NO.

DEFERBLKS = VALUE-DEC(5-200) is the number of defer blocks to allocate. Defer blocks are required when messages not in memory are needed. The number of blocks required depends on the comgroup activity level. Statistics available to the AU will reveal whether the number supplied is too small or too large. The minimum number is 5, and the maximum is 200. The default is 20.

DISKWARN = VALUE-DEC(0-18000) applies when the number of free granules in the comgroup file falls below this value. The AU will receive a warning message when this occurs. (See CG_WARN under EQU_CG) The default is 0 (no message ever).

DMTYP = VALUE-CHAR(8) is the default message type for messages written from DCBs.

DRML = {YES|NO} indicates whether (YES) or not (NO) DCB stations will most often read their own message queues with LATCH. The value given to DRML affects the way in which the comgroup processes read and write operations in order to produce minimum overhead. Note that if all or most DCB stations read with DIRONLY = NO (see VLP_STATION), then DRML and QRML should be set identically. The default is NO.

DSCREQ = {YES|NO} indicates whether (YES) or not (NO), disconnecting TERMINAL stations are to request permission to do so from the AU.

When DSCREQ is NO, a disconnecting TERMINAL simply sends the %CG_TDSC# message to the AU and disconnects. Since this disconnection is asynchronous with respect to the disconnect message being delivered to the AU, another (or perhaps the same) TERMINAL can join the comgroup under the same station name before the AU is aware of the disconnect. Thus, the AU can never be sure, for example, what particular incarnation of a given station name will be affected by, say, an M$ACTIVATE.

If necessary, this type of problem can be eliminated by specifying DSCREQ as YES. When this is done, TERMINAL stations wishing to disconnect send a %CG_TDRQ (disconnect request) event message to the AU and then wait for him to respond with M$DEACTIVATE DSCGO. Whilst such a station is waiting, any attempt to M$ACTIVATE it will produce a 'disconnected' error, as will an M$DEACTIVATE that does not specify DSCGO. Furthermore, the station is treated as absent if it is being written to. When the M$DEACTIVATE DSCGO is done, the station disconnects and the normal %CG_TDSC event message is sent to the AU.

If the AU disconnects from the comgroup, then all waiting TERMINAL stations are allowed to disconnect automatically.

DSCREQ is forced NO if CONMSG is NO.

The default is NO.

DVBYTE is a collection of bits controlling the manner in which TERMINALs are read. That is, they supply parameters controlling the way in which TERMINALs supply data to the comgroup. Each bit is specified separately.

INPUT = {YES|NO} specifies whether or not terminals connected to this comgroup are capable of inputting data to the comgroup (i.e., generating messages). The default is YES.

JOURNAL = {YES|NO} indicates whether (YES) or not (NO) journaling is to be performed in the comgroup. If NO, then this overrides the presence of JRNLSTA and a JRNLFID. If YES, then the comgroup must possess a JRNLSTA (see FPT_CGCTL).

If the comgroup is being journaled by the system JOURNAL GHOST (see GHSTACS in M$DCB), then the ghost is informed whenever the value of JOURNAL is changed.

The default is NO.

MAXMC = VALUE-DEC(0-2048) specifies maximum byte count permitted for a single write into the comgroup. The default is 1024.

MAXPG = VALUE-DEC(3-1000) is the maximum number of memory pages to use. The default is 10.

MEMWARN = VALUE-DEC (0-18000) specifies the level at which a warning message will be issued to the AU (see CG_WARN# in EQU_CG).

When the comgroup is acquiring more memory in order to meet demand, the AU may wish to know if the amount of memory currently in use is approaching the maximum possible. Note that this maximum is derived from the interaction between MAXPG and DATAPGS.

MEMWARN establishes the value for maximum minus current at which the warning will be issued.

If MEMWARNP is NO, then 'current' is construed to mean all context memory. This includes station- and type-tree nodes, as well as MBLK storage and miscellaneous other items. If MEMWARNP is YES, then 'current' is construed to mean only 'permanent' storage, that is station- and type-tree nodes.

For example, suppose MAXPG is 125 and DATAPGS is 20. This means that the maximum for all context is 100 pages. Suppose further that MEMWARN is 20 and MEMWARNP is NO. Then the warning will be issued whenever the number of current context pages goes from 79 to 80. If, on the other hand, MEMWARNP is YES, then the warning will not be issued until the number of pages allocated to station- and type- tree nodes goes from 79 to 80. Note that before the warning is issued, the actual number of context pages may climb to 100, due to demand for MBLK storage. Note further that in this case the comgroup may be resorting to spilling MBLKs to disk, without the AU receiving any warning.

The setting of MEMWARNP essentially depends on what the AU wishes to prevent. If the AU wishes to prevent MBLK spillage, then MEMWARNP should be specified as NO. On the other hand, if MBLK spillage is deemed acceptable (or perhaps even desirable), then MEMWARNP should be specified as YES, and then the warning will only be issued when the amount of non-spillable context reaches the MEMWARN limit.

The default for MEMWARN is zero (no warning will ever be issued).

MEMWARNP = {YES|NO} indicates whether (YES) or not (NO), the MEMWARN parameter is to apply only to 'permanent' context.

See MEMWARN. The default is NO.

MINPG = VALUE-DEC(2-1000) is the minimum of number of physical memory pages to maintain. The default is 2.

MXACT = VALUE-DEC(0-?) is the default value for the number of messages of a given Q-tree type that can be 'active' at one time. An 'active' message is one which has been read from the anonymous queue by a DCB specifying LATCH. The message becomes inactive when it is unlatched. Note, then, a message directed to a station is never 'active' in this sense, regardless of its MSGTYP or origin station. The default is 9999999.

OUTPUT = {YES|NO} specifies whether or not terminals connected to this comgroup are capable of outputting data from the comgroup (i.e. receiving messages). The default is YES.

QRML = {YES|NO} indicates whether (YES) or not (NO) the anonymous queue will most often be read with LATCH. The value given to QRML affects the way in which the comgroup processes read and write operations in order to produce minimum overhead. Assigning the correct value to QRML is only important in a large comgroup with many simultaneous outstanding read requests against the anonymous queue. The default is NO.

RAS = {YES|NO} If YES, reading from an absent or deactivated station is legal. If NO, such action is an error.

When RAS is NO, an M$READ which specifies (either in VLP_STATION or VLP_SETSTA) a STATION other than '?' (don't-care) will receive a 'station-disconnected' error if there is no such station currently connected and active. Furthermore, if such a read request is pending, and the last or only such station disconnects or is deactivated, then the request will be terminated with that error.

One special exception is noteworthy. Even if RAS is YES, such a read request will be errored/terminated as described above if it specified a MSGTYP beginning with '*'.

The default is NO.

REDUNDANT = {YES|NO} specifies whether or not the comgroup is to have data redundancy. Data redundancy increases by about 64 bytes the amount of space required to store a message (either in memory or on disk), carries with it no penalty in terms of execution time, and makes it possible to reconstruct the comgroup from the disk overflow area in the event of a disastrous system failure (i.e., one in which files are not closed). The default is NO.

REREAD = {YES|NO} REREAD = YES specifies that the previous input line is to be echoed and set to be the current input line. The FPT field name for this parameter is VLP_CGCP.DVBYTE.REREAD#. The default is NO.

SECURE = {YES|NO} specifies the mode of operation with respect to guarantee of message delivery. If YES, the specifications for SMD and WAS are overridden and set to YES, and messages read by TERMINALs are not deleted until the entire message has been output. This has the effect of ensuring delivery of any message which has been successfully written into the comgroup, and is destined for a TERMINAL. Note that this does not apply to messages read by DCBs, but that a similar mode of operation may be enforced by supplying a VLP_SETSTA on M$CGCTL that has LATCH=YES, thus forcing all DCBs to read with LATCH and ensuring message delivery.

SECURE = NO permits TERMINAL-directed messages to be deleted after the message has been delivered but (possibly) before it has actually been output, and has no effect on SMD or WAS.

The default is NO.

SMD = {YES|NO} If YES, messages written to a station will be saved when it disconnects, to be passed to it when it reconnects. If NO, such messages are deleted upon disconnect. NO forces WAS to NO also (see SECURE, also). The default is NO.

STAPRIO = VALUE-DEC(1-500) is the default value for the part of message priority that comes from the origin station. The default is 1.

TERMCONAU = {YES|NO} TERMCONAU = YES indicates that terminals are to be activated upon connection if the AU is connected. The default is YES.

TERMCONLGL = {YES|NO} specifies whether or not terminals are allowed to connect to this comgroup. If NO, this will prevent connection despite the system manager having authorized logons which specify connection to this comgroup. The default is YES.

TERMCONNAU = {YES|NO} TERMCONNAU = YES indicates that terminals are to be activated upon connection if the AU is not connected. The default is YES.

TMTYP = VALUE-CHAR(8) is the default message type for messages written from terminals.

TRANS = {YES|NO} TRANS = YES specifies transparency, (i.e., that data is to be read without translation). This option is useful primarily for devices which normally perform translation (for instance, asynchronous terminals). The FPT field name for this parameter is VLP_CGCP.DVBYTE.TRANS#. The default is NO.

TRMRDSIZ = VALUE-DEC(1-MAXMC) specifies the size of the reads done to terminals by the comgroup. TRMRDSIZ thus specifies the largest message that can be input from a terminal and the point at which asynchronous terminals will activate on byte count.

The minimum is 1; the maximum is the value specified for MAXMC. The default is 140.

TYPPRIO = VALUE-DEC(1-500) is the default value for the part of message priority that comes from the message type. The default is 1.

UNLAMSG = {YES|NO} specifies whether or not the AU wishes to receive messages automatically generated by UNLATCH. (See M$UNLATCH). If NO, such messages will not be generated. The default is YES.

WAS = {YES|NO} WAS = YES specifies that writing to an absent or deactivated station is legal. If NO, such action is an error (see SMD and SECURE, also). The default is NO.

WRITETIME = VALUE-DEC(0-511) specifies the time in minutes that data pages are to be allowed to remain in memory without being written to disk. The default is zero, which indicates that data pages are never to be written for this reason.

XSTALGL = {YES|NO} indicates whether (YES) or not (NO) any station name is legal within the comgroup. This applies to joining the comgroup, and writing to specific stations. If YES, then DCBs and TERMINALs may join the comgroup under any name they chose, which may require the comgroup to build a new node in the station tree when the join takes place. Also, stations may write to any station they chose, which, if the destination station is not already represented in the station tree, requires building a new node. If XSTALGL is NO, then one may only join under, or write to, a station name which is already present in the tree. XSTALGL = NO is usually used in conjunction with the STAFC = SETLIST option of M$CGCTL, which defines the exact list of legal station names, and inserts them in the station tree. See also PERM in VLP_CGSTAL.

The default is YES.

XTYPLGL = {YES|NO} indicates whether (YES) or not (NO) any message type is legal within the comgroup. This applies to write operations, and also to reading with a specific message type. If YES, then stations may write messages with any type they chose, which, if the type is not already represented in the message type tree, requires building and entering a new node. If XTYPLGL is NO, then one may only write a message with a type which is already present in the tree. XTYPLGL = NO is usually used in conjunction with the TYPFC = SETLIST option of M$CGCTL, which defines the exact list of legal message types, and inserts them in the message type tree. See also PERM in VLP_CGTYPL.

The default is YES.

VLP_CGQL

The VLP_CGQL macro generates an area for use with the M$CGCTL and M$CGINFO services. It generates an array of parameters and statistics, with one entry for each anonymous queue node. The array is parallel to the array in VLP_CGSTAL or VLP_CGTYPL, depending upon how the queue is keyed. All items are returned by the M$CGINFO service while some can be provided to the M$CGCTL service. These are distinguished by the fact that they have default values.

Note that if more than one queue node is being generated to be supplied to M$CGCTL, then in general the values for the different queue nodes must be assigned at execution time. This is because when the macro is invoked at compile time, only one value may be specified for the various attributes. The values specified for the various attributes will initialize ALL the array elements, thus at least providing 'default' values.

LIST is the array of queue nodes.

LIST.AUTORCVR = {YES|NO} selects the action to be taken when messages from this queue node are active (LATCHed) at the time of a system recovery. YES requests that an UNLATCH RERUN be performed. NO causes an UNLATCH HOLD to be performed, and also causes MXACT for the queue node to be set to zero. Specifying NO permits the AU user to perform explicit recovery for transactions that are active at the time of system recovery. The default is YES.

LIST.CNACT = VALUE-DEC(0-?) is the current number of active messages from this queue node.

LIST.MSGCNT = VALUE-DEC(0-?) is the count of messages from this queue node which have not yet been read.

LIST.MXACT = VALUE-DEC(0-?) is the maximum number of messages from this queue node which may be active (read with LATCH) at any time. The default is 9999999.

NLIST = VALUE-DEC(1-n) is the number of queue node slots to be generated. The default is 1.

VLP_CGSTAL

This macro generates an area for use with the M$CGCTL and M$CGINFO services. It generates an array of stations, each with a name and a list of attributes. All information in the array is returned by the M$CGINFO service. Parameters that can be provided to the M$CGCTL service are distinguished by the fact that they have default values.

Note that if more than one station is being generated to be supplied to M$CGCTL, then in general the values for the different stations must be assigned at execution time. This is because when the macro is invoked at compile time, only one value may be specified for NAME and the various attributes. The value specified for NAME will initialize NAME# only in the first array element. The values specified for the various attributes will initialize ALL the array elements, thus at least providing 'default' values.

FOUND = VALUE-DEC(0-n) is set to the number of stations found by M$CGINFO DUMP. Note that FOUND is not used by M$CGCTL.

LIST is the array of stations.

LIST.ACTIVE = VALUE-BIT(1)is set if the station is activated.

LIST.AUP = {YES|NO} specifies whether (YES) or not (NO) this station is administrative user proprietary. Such a station has two special features. First, it cannot be written to by a normal write operation. If one is attempted, then the writer perceives the destination station as being absent (independent of WAS in his VLP_STATION, and independent of the comgroup control parameter WAS), and is given an error. The following is a list of the only classes of messages which may be sent to an AUP station:

- o HELD messages if the station is HOLDSTA (see FPT_CGCTL).
- o JOURNALed messages if the station is JRNLSTA (see FPT_CGCTL).
- o Any message whose type begins with '*'. This of course includes all AU event messages.

Second, no terminal is permitted to join the comgroup under that station name, and no DCB is permitted to join the comgroup under that name unless the DCB owner already has an AU DCB open to the comgroup.

Thus, the AUP attribute is even more restrictive that the 'special' attribute discussed under HOLDSTA in FPT_CGCTL. A typical usage for AUP is its application to the station designated as HOLDSTA (see FPT_CGCTL).

The default is NO.

LIST.CON = VALUE-BIT(1) is set if the station is connected.

LIST.MSGCNT = VALUE-DEC(0-?) is the count of messages written specifically to this station, and waiting to be read.

LIST.NAME = VALUE-CHAR(8) is the name of the station. This must be supplied to M$CGCTL. Station names may be composed of any characters, with the exception of the first character which must belong to the list given under NAME# in VLP_CGTYPL. Also, the first character may not be the character '&'.

LIST.PERM = {YES|NO} indicates whether (YES) or not (NO) this station is permanent and not to be deleted. Note that 'deleted' refers to the station node itself, not to messages written to the station. In particular, a station node is only considered for deletion when the station has left the comgroup and there are no messages in its message queue (and, if it is a Q-tree node also, there are no messages in its Q-tree queue). Thus the PERM parameter only takes effect in that situation, and may be set NO to prevent such station nodes from cluttering up the comgroup.

The PERM attribute interacts with XSTALGL (see VLP_CGCP). If XSTALGL is YES, then PERM for a station only affects its deletion as discussed above. If, however, XSTALGL is NO, then only those stations designated as PERM (or SPECIAL - see HOLDSTA in FPT_CGCTL) are guaranteed to be always legal. This is because a station with PERM = NO may be deleted from the comgroup under the conditions detailed above, and when this happens, the station name becomes illegal to write to or join under, by virtue of XSTALGL being NO. Note that changing XSTALGL from YES to NO does not make non-PERM stations instantly illegal. Thus, creating non-PERM stations via M$CGCTL, in a comgroup with XSTALGL = NO, causes certain anomalies.

The default is YES.

LIST.PRIO = VALUE-DEC(1-500) is the contribution to total priority of messages written from this station. The default is 1.

LIST.READS = VALUE-DEC(0-?) is the number of messages read by the station. This value is reset to zero each time the station joins the comgroup (but not when it disconnects).

LIST.WRITES = VALUE-DEC(0-?) is the number of messages written into the comgroup by the station. This value is reset to zero each time the station joins the comgroup (but not when it disconnects).

NSTA = VALUE-DEC(0-n) is the number of station slots to be generated. The default is 1.

VLP_CGTYPL

This macro generates an area for use with the M$CGCTL and M$CGINFO services. It generates an array of message types, each with a name and a list of attributes. All information in the array is returned by the M$CGINFO service. Parameters that can be provided to the M$CGCTL service are distinguished by the fact that they have default values.

Note that if more than one station is being generated to be supplied to M$CGCTL, then in general the values for the different stations must be assigned at execution time. This is because when the macro is invoked at compile time, only one value may be specified for NAME and the various attributes. The value specified for NAME will initialize NAME# only in the first array element. The values specified for the various attributes will initialize ALL the array elements, thus at least providing 'default' values.

FOUND = VALUE-DEC(0-n) is set to the number of message types found by M$CGINFO DUMP. Note that FOUND is not used by M$CGCTL.

LIST is the array of message types.

LIST.JOURNAL = {YES|NO} indicates whether (YES) or not (NO) messages of this type are to be journaled. If YES, then two specific processes are invoked. First, when a message of this type is created (i.e. written into the comgroup), it will be sent to the station specified by JRNLSTA (see FPT_CGCTL) instead of to its proper destination. Second, when a message of this type is finally deleted from the comgroup, a special journal delete message will be sent to the station specified by JRNLSTA, provided that it has been M$UNLATCHed with the RERUN and JOURNALED options by the journal station.

The journal delete message has MSGTYP '*AUEV'. Like all other AU-event messages, the data portion is framed by B$CGAURD, in which B$CGAURD.EVENT contains the value CG_JNLD#, and B$CGAURD.MID contains the MSGID of the message just deleted. The origin station of this message (reported in B$CGAURD.STATION and VLP_STATION.STATION) is undefined.

The default is NO.

LIST.NAME = VALUE-CHAR(8) is the name of the message type. This must be supplied when calling M$CGCTL. Message type names may be composed of any characters, with the exception of the first character. The first character must be one of the following:

```
    *ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_
    :$&-abcdefghijklmnopqrstuvwxyz
```

Note that although the character '*' is present in this list, CP-6 file management reserves the '*AUEV' message type for its own use.

LIST.ONEREPORT = {YES|NO} indicate whether (YES) or not (NO), latched outputs of this message type are to be grouped as single reports.

When ONEREPORT is YES, all LATCHed M$WRITEs through a given DCB are treated as though CONT was specified as YES on the M$WRITE. The effect of this is that the M$WRITEs, instead of creating multiple messages, create a single continued message. The message will automatically be ended when unlatch is performed.

ONEREPORT is intended for use in cases where the destination of the message is a TERMINAL station and many records are being written which are logically a single 'report'. It has the desired effect (that is, the target station reads the message in 'chunks' with each chunk corresponding to a single M$WRITE) because TERMINAL stations, unlike DCB stations, 'see' the divisions between pieces of a CONTinued message.

Since such a message may only be ended by an unlatch operation, ONEREPORT is of limited utility for normal user programs inasmuch as it limits them to a single output message per latched input. However, for TP users only, output DCBs may, if they are opened in CREATE mode to the comgroup, be used to create ONEREPORT reports, one per DCB. Thus the TP user will have one output DCB for each report of this type that is to be produced for the input 'transaction'.

The actual operation of ONEREPORT (i.e. the forcing of CONT on the M$WRITE) may change in future releases, but the overall effect will be the same, at least for target TERMINAL stations. The result of using ONEREPORT to write to a DCB station is undefined.

The default is NO.

LIST.PERM = {YES|NO} indicates whether (YES) or not (NO) this type is permanent and not to be deleted. Note that 'deleted' refers to the type node itself, not to messages written of that type. In particular, a type node may be deleted at any time if PERM is NO and (if the message type tree is the Q-tree) the node's Q-tree queue is empty. The PERM parameter may be set to NO to prevent such nodes from cluttering up the comgroup.

The PERM attribute interacts with XTYPLGL (see VLP_CGCP). If XTYPLGL is YES, then PERM for a type only affects its deletion as discussed above. If, however, XTYPLGL is NO, then only those types designated as PERM are guaranteed to be always legal. This is because a type with PERM = NO may be deleted from the comgroup under the conditions detailed above, and when this happens the type name becomes illegal to write with, by virtue of XTYPLGL being NO. Note that changing XTYPLGL from YES to NO does not make non-PERM types instantly illegal. Thus, creating non-PERM types via M$CGCTL, in a comgroup with XTYPLGL = NO, causes certain anomalies.

The default is YES.

LIST.PRIO = VALUE-DEC(1-500) is the contribution to total priority of messages written with this type. The default is 1.

NTYP = VALUE-DEC(1-n) is the number of message type slots to be generated. The default is 1.

VLP_JRNLFID

The VLP_JRNLFID macro is used to generate the FID for a journal file. It generates a length halfword and a character string of fixed length. See also VLP_JRNLFIDV.

The maximum length of the text string is 2048 characters.

JRNLFID = VALUE-CHAR. Specifies a FID. The length of the CHAR field generated is the length of JRNLFID (31 if JRNLFID is not specified). The LEN parameter can be specified to override the length (e.g. LEN=5). See M$DCB for the character set for file names. If STCLASS=BASED is used, LEN must be specified. See also VLP_JRNLFIDV for a different type of based structure.

L is the length halfword. It is set for the user to the length of JRNLFID, or to 31 if JRNLFID is not specified.

LEN = VALUE-DEC(0-?) specifies the number of bytes to be provided for the FID.

VLP_JRNLFIDV

The VLP_JRNLFIDV macro generates a based version of the VLP_JRNLFID macro with the length of the character string being the value currently in the length halfword.

VLR_CGSTATS

This macro generates an area which receives statistics global to the entire comgroup.

CCMQ = VALUE-DEC(0-?) is the current count of messages in the anonymous QUEUE.

CCMS = VALUE-DEC(0-?) is the current count of messages waiting at specific stations.

DISCRDS = VALUE-DEC(0-?) is the number of disk reads since open.

DISCWRS = VALUE-DEC(0-?) is the number of disk writes since open.

GARBCOLLS = VALUE-DEC(0-?) is the number of times the memory "garbage collector" was forced to run.

GETS = VALUE-DEC(0-?) is the count of TERMINAL READs since opening of the comgroup.

NODFRS = VALUE-DEC(0-?) is the number of times a defer block was required, but there were none available.

NUMDCBS = VALUE-DEC(0-?) is the number of DCBs open to the comgroup.

NUMTERMS = VALUE-DEC(0-?) is the number of connected terminals.

PGSI = VALUE-DEC(0-?) is the page storage integral which is calculated by multiplying (number of memory pages) times (time allocated in 10 millisecond units), stored as a floating point number.

PUTS = VALUE-DEC(0-?) is the count of TERMINAL WRITEs since opening of the comgroup.

READS = VALUE-DEC(0-?) is the count of DCB READs since opening of the comgroup.

WRITES = VALUE-DEC(0-?) is the count of DCB WRITEs since opening of the comgroup.

M$FWCG - Find Messages within a Comgroup

The M$FWCG service is used to find some subset of the messages within a comgroup, and then optionally produce a report about them as well as optionally deleting or changing them. Usage of the informational (reporting) aspect of this service is restricted to users having AURD access to the comgroup. Usage of the operational (delete or change) aspect of this service is limited to the AU of the comgroup.

The desired subset of messages is defined by the SCA, CRITERIA and QSELECT parameters. If CRITERIA is NIL then one message is being selected, namely the currently-latched input message of the service call issuer. In this case DELETE and REATTR are ignored, that is the service becomes informational only. Note that such latched messages may only be viewed via M$FWCG in this special way, that is to say the service call issuer can only 'see' his own latched input message by this special method, and can never see any other currently-latched messages.

Otherwise, that is when a VLP_CGMA is supplied as CRITERIA, QSELECT defines the class(es) of messages being selected. The two classes are directed messages (i.e. those written to a specific station) and anonymous messages (i.e. those in the anonymous queue). QSELECT may be used to select either or both of these classes.

The VLP_CGMA framed by CRITERIA contains certain specifications for the attributes of the desired messages. The SCA parameter, framing a VLP_SCGMA, defines which of these attributes are to be used in defining the desired subset. Thus, for example, to select messages written to stations whose names begin with 'ABC', one specifies QSELECT=DIRECT, VLP_CGMA.DEST='ABC?', and VLP_SCGMA.DEST=YES, with all other options in VLP_SCGMA set to NO. To further qualify the subset and select only messages of type 'XYZ', one would also set VLP_SCGMA.MSGTYP to YES, and VLP_CGMA.MSGTYP to 'XYZ'. Note that for the criteria MSGTYP, DEST, and ORIGIN, one may indicate don't-care (i.e. no restriction) in two ways, namely by setting the VLP_CGMA attribute to '?' and the VLP_SCGMA option to YES, or simply by setting the VLP_SCGMA option to NO.

Once the desired set of messages has been defined as described above, the function of the service must be specified. There are two aspects to this service: informational and operational.

The informational function returns the attributes of the messages found. This function takes two forms. First, the ONEATTR parameter, if specified, frames a VLP_CGMA area which is to receive the attributes of the first message found. This is most commonly used when a single message is expected, particularly in the case where CRITERIA is NIL (see above). The second form of information return, which may be specified in conjunction with ONEATTR, is INFORM. Setting INFORM to YES requests that the service build a special message containing a list with one entry for each message found, with each entry consisting of the attributes of the corresponding message. This special message will be sent to the service call issuer at the conclusion of the service call, and may subsequently be read at the issuer's discretion. Its message ID is returned into MSGID in VLR_FWCG. See the discussion of the INFORM option for the specific structure of this message. Note that the size of the data of this special message depends on the number of messages found by the service, which is to say it may be very large. The message may be read safely irrespective of size by specifying CONT on the M$READ. Note that INFORM requires the comgroup to have a value of at least 2 for the minimum number of data cache pages. If the minimum is less than 2, then M$FWCG will return an error.

The operational function allows for deleting the selected messages or changing their attributes (called REATTR). These two choices are mutually exclusive. Specifically, the DELETE option is interrogated first; if DELETE is YES then the selected messages are deleted. If DELETE is NO, then the selected messages are changed if REATTR is specified. Thus requesting the informational-only form of this service requires setting DELETE to NO and REATTR to NIL.

Changing the attributes of the selected messages is limited by the fact that only one set of new attributes may be supplied. In other words, when REATTR is performed, all selected messages have their attributes changed in the same way. The set of attributes to be changed is defined by SRA which frames a VLP_SCGMA. The new values for these attributes are defined in REATTR which frames a VLP_CGMA. Thus, for example, to redirect messages to a specific station, say 'ABC', one specifies YES for VLP_SCGMA.DEST (in SRA), and 'ABC' for VLP_CGMA.DEST (in REATTR).

Once the attributes of a message have been changed, the message may have to be moved as a result. For example, if PRIO or UTS is changed then the message will need to be moved to a different place in its original message queue. If DEST is changed then the message will have to be moved to a different queue. Note in particular that if a non-blank DEST is assigned to a message originally found in the anonymous queue, then the message leaves the anonymous queue and is placed in the queue of messages destined for the station whose name is in DEST. Note further that the REATTR operation is limited by the way in which the comgroup is defined. Thus assigning a new DEST of, say, 'ABC' will cause the message to be deleted if station 'ABC' is not present and the comgroup does not allow writing to absent stations.

AU event messages (message type '*AUEV') may be deleted but may not have their attributes changed. When a REATTR operation is being performed, the count of messages found and the INFORM result (if any) will include the AU event messages, but they will be passed over during the REATTR phase.

Particular care must be exercised when changing the MSGID. Under normal circumstances it is guaranteed that each message in the comgroup has a unique MSGID-MSGIDXT pair. CP-6 file management makes little attempt to ensure that this will remain true when M$FWCG is used to change the MSGID of a message. For further information, refer to MSGID in VLP_SCGMA, and to the section on 'REBUILDING A COMGROUP' in the general discussion of comgroups.

The form of the call for this service is:

CALL M$FWCG (FPT_FWCG) [ALTRET (label)];

The parameters are as follows:

CRITERIA = VARIABLE locates a VLP_CGMA area containing the selection criteria describing the desired messages. See also SCA.

If CRITERIA is NIL, then the service is assumed to be selecting one specific message, namely the currently-latched input message associated with the DCB supplied with the service call. In this case, QSELECT is ignored; furthermore, DELETE and REATTR are ignored, that is, the selected message is not affected in any way. Note that this is the only way in which such a latched message may be 'seen' via M$FWCG.

The default is NIL.

DCB = DCBNAME specifies the DCB associated with the comgroup.

DELETE = {YES|NO} specifies whether (YES) or not (NO) the selected messages are to be deleted from the comgroup. If NO, then the REATTR option is honored if it is specified. The default is NO.

INFORM = {YES|NO} specifies whether (YES) or not (NO) a report is to be generated concerning the selected messages. If YES, then a special message will be created and sent to the station associated with the DCB supplied on the service call. This message has message type '*AUEV', and the first portion of the message data is structured as are all such messages, that is, it is framed by the structure B$CGAURD. In particular, the B$CGAURD.EVENT word contains the value CG_FWCG#. The remainder of the message data consists of a series of VLP_CGMA frames, one for each message found by the service, containing the attributes of those messages. The MSGID of this special message is returned in MSGID of VLR_FWCG (see RESULTS). The default is NO.

ONEATTR = VARIABLE locates a VLP_CGMA area into which the attributes of the first message found are to be returned. Specification of ONEATTR has no effect on the action of INFORM. The default is NIL.

QSELECT = OPTION is used in addition to CRITERIA to restrict the set of messages selected. DIRECT specifies that only those messages destined for specific stations are to be considered. ANONYMOUS specifies that only messages in the anonymous queue are to be considered. ALL specifies that both classes of messages are to be considered.

QSELECT is only meaningful if CRITERIA is specified. The default is ALL.

REATTR = VARIABLE locates a VLP_CGMA area containing the new attributes to be given to the selected messages. REATTR applies only if DELETE is NO, and if specified, indicates that the REATTR function is to be performed. See also SRA. The default is NIL.

RESULTS = VARIABLE locates a VLR_FWCG area into which certain result information will be returned. The default is NIL.

SCA = VARIABLE locates VLP_SCGMA area containing a list specifying which of the attributes supplied in CRITERIA are actually to be used in selecting the desired messages. SCA must be specified if CRITERIA is specified. The default is NIL.

SRA = VARIABLE locates a VLP_SCGMA area containing a list specifying which of the attributes supplied in REATTR are actually to to be used in changing the selected messages. SRA must be specified if REATTR is specified. The default is NIL.

VLR_FWCG

This macro generates an area which receives the results of M$FWCG.

FOUND = VALUE-DEC(0-n) is set to the number of messages found by the service.

MSGID = VALUE-DEC(1-n) is set to the MSGID of the message generated when INFORM is specified as YES.

VLP_SCGMA

This macro generates an area for use with M$FWCG. It contains a list of attribute flags used to select a subset of the attributes present in a VLP_CGMA area.

All of the flags are honored for SRA in M$FWCG, that is to say for REATTR. Only the following flags are honored for SCA, that is, for message selection criteria:

MSGTYP
ORIGIN
DEST
MSGID
MSGIDXT

DEST = {YES|NO} selects (YES) or ignores (NO) the DEST attribute in the corresponding VLP_CGMA. The default is NO.

DVE = {YES|NO} selects (YES) or ignores (NO) the DVE attribute in the corresponding VLP_CGMA. The default is NO.

MSGID = {YES|NO} selects (YES) or ignores (NO) the MSGID attribute in the corresponding VLP_CGMA.

Note that if MSGID is YES for a REATTR function, then the current comgroup message ID will be set to the supplied message ID plus one if it is not already greater. This attempts to prevent duplicate message IDs in the comgroup.

The default is NO.

MSGIDXT = VALUE-DEC(0-6) selects (nonzero) or ignores (zero) the MSGIDXT attribute in the corresponding VLP_CGMA. MSGIDXT is only meaningful if MSGID has been specified.

For REATTR, that is to say in SRA, any nonzero value for MSGIDXT here causes the entire MSGIDXT in the selected message(s) to be changed to the MSGIDXT in the REATTR VLP_CGMA.

For message selection, that is to say in SCA, the meaning of MSGIDXT is somewhat different. If the comgroup is using the second type of message ID extension (see MSGID in VLP_STATION), then any nonzero value for MSGIDXT has the effect of selecting the entire MSGIDXT in VLP_CGMA.

On the other hand, if the comgroup is using the first type of message ID extension, namely the six-level structure, then the MSGIDXT in CRITERIA may be wild-carded by appropriate setting of MSGIDXT here. When MSGIDXT is set to N, then the first N levels in MSGIDXT in CRITERIA are used for message selection. For example, if the MSGID in CRITERIA is 7, and the MSGIDXT in CRITERIA is 2-3-4-1-0-0, then setting MSGIDXT to 2 here causes a search for all messages bearing MSGID 7 and MSGIDXT 2-3-?-?-?-?. VLP_CGMA permits the user to specify the various level values individually, as MIDXT1, MIDXT2, ... , MIDXT6.

The default is zero.

MSGSIZE = {YES|NO} selects (YES) or ignores (NO) the MSGSIZE attribute in the corresponding VLP_CGMA.

Note that if MSGSIZE is YES for a REATTR function, then the size of the subject message may only be reduced, not increased.

The default is NO.

MSGTYP = {YES|NO} selects (YES) or ignores (NO) the MSGTYP attribute in the corresponding VLP_CGMA. The default is NO.

ORIGIN = {YES|NO} selects (YES) or ignores (NO) the ORIGIN attribute in the corresponding VLP_CGMA. The default is NO.

PRIO = {YES|NO} selects (YES) or ignores (NO) the PRIO attribute in the corresponding VLP_CGMA.

Note that if PRIO is YES for a REATTR function, then certain limits are placed on the legal values. For subject messages that are not AU event messages, the legal limits for PRIO in the VLP_CGMA are 0 to 256500. If the PRIO in VLP_CGMA is not legal, then no error will be reported; instead the highest possible legal value will be used.

The default is NO.

TYC = {YES|NO} selects (YES) or ignores (NO) the TYC attribute in the corresponding VLP_CGMA. The default is NO.

UTS = {YES|NO} selects (YES) or ignores (NO) the UTS attribute in the corresponding VLP_CGMA. The default is NO.

VLP_CGMA

This macro generates an area for use with M$FWCG. It contains the attributes of a comgroup message. Those attributes which are honored in CRITERIA of M$FWCG are distinguished by the fact that they have default values.

DEST = VALUE-CHAR(8) is the destination of the message. If DEST is blanks, then the 'destination' of the message is the anonymous queue, otherwise DEST is the name of the specific station to which the message was written.

When supplied as part of CRITERIA to M$FWCG, DEST may be wild-carded. The default is '?', meaning that messages with any destination are desired. See also the QSELECT parameter of M$FWCG.

DVE is a collection of bits containing certain attributes of the message as a data record. The contents of DVE are not normally of interest to the issuer of M$FWCG other than as something to be received (as part of ONEATTR or INFORM), and supplied (as part of REATTR) when a message is being saved and reinserted.

MSGID = VALUE-DEC(1-n) is the comgroup message id of the message. Refer to the the discussion of MSGID under VLP_STATION.

When supplied as part of CRITERIA to M$FWCG, MSGID may be specified as zero to indicate that all message IDs are desired.

The default is 0.

MSGIDBXT = VALUE-DEC(0-?) is the alias of MSGIDXT, for use when message ID extensions take their second possible form. See MSGIDXT.

MSGIDXT = VALUE-BIT(36) is the extension to MSGID. Refer to the discussion of MSGID and MSGIDXT in VLP_STATION.

If MSGIDXT is to be supplied when this macro is invoked, it must be specified in its six component parts: MIDXT1, ... , MIDXT6.

MSGIDXT.MIDXT1 = VALUE-DEC(1-62) (See MSGIDXT).

MSGIDXT.MIDXT2 = VALUE-DEC(1-62) (See MSGIDXT).

MSGIDXT.MIDXT3 = VALUE-DEC(1-62) (See MSGIDXT).

MSGIDXT.MIDXT4 = VALUE-DEC(1-62) (See MSGIDXT).

MSGIDXT.MIDXT5 = VALUE-DEC(1-62) (See MSGIDXT).

MSGIDXT.MIDXT6 = VALUE-DEC(1-62) (See MSGIDXT).

MSGSIZE = VALUE-DEC(0-n) is the size in bytes of the message.

MSGTYP = VALUE-CHAR(8) is the type of the message.

When supplied as part of CRITERIA to M$FWCG, MSGTYP may be wild-carded. The default is '?', meaning all types are desired.

ORIGIN = VALUE-CHAR(8) is the name of the station which originated (wrote) the message.

When supplied as part of CRITERIA to M$FWCG, ORIGIN may be wild-carded. The default is '?', meaning messages with any origin are desired.

PRIO = VALUE-DEC(0-262143) is the message priority. Priorities of normal messages lie in the range 1 through 256000. However, some special messages (generated by the comgroup itself), notably the AU event messages, have priorities outside this range.

TYC = VALUE-BIT(36) is a collection of bits to be passed as a TYC value to the eventual reader of the message. See the discussion of TYC under M$DCB. TYC is similar to DVE in that it is not normally of interest other than as a field to be received and supplied.

UTS = VALUE-DEC(0-n) is the system time stamp (in UTS format) of the time at which the message was created.

Section 4

Program Management Services

The frequently used program management services are explained in this section. Additional services that perform privileged or restricted program management functions are described in Section 8 of this manual.

Program Suspension and Termination

The monitor provides services to temporarily suspend a program and to terminate a program and initiate job step processing. The monitor services that provide for program termination pass a value in the Step Condition Code which is available to subsequent job steps as well as to the command processor. This mechanism allows the programmer or command processor to take appropriate actions following the completion of a program.

M$WAIT - Suspend Program

The M$WAIT service causes suspension of program execution for a specified period of real time. Program execution is resumed at the next statement following the M$WAIT request. A suspended program is resumed before its suspension time has expired if an asynchronous event is reported (see M$EVENT). Optionally, the amount of unexpired time may be returned to the user.

The form of the call for this service is:

CALL M$WAIT (FPT_WAIT) [ALTRET(label)];

The parameters for this service are as follows:

UETIME = VARIABLE locates a 1-word area into which the unexpired suspension time, in seconds, is to be stored. If UETIME is not specified, or is NIL, the time is not returned. If the suspension is not terminated prematurely, the unexpired time will be zero. Default=NIL.

UNITS = VALUE-DEC(0-2**36) specifies the number of seconds the program is to be suspended and is treated as modulo 24 hours. Default=0.

M$EXIT - Terminate Program Normally

The M$EXIT service provides for normal termination of the current program. When called bv the user program, M$EXIT typically enters job step processing, following execution of any exit control procedure(s). If the exiting program was called by M$LINK, the M$EXIT service returns to the calling program, following execution of any exit control procedure(s).

For complete information on exit control, see the discussion of M$XCON in Section 6. That discussion includes a description of exit control in both the user program and Alternate Shared Library.

The FPT_EXIT parameters allow for explicit setting of the Step Condition Code and the error code to be used by the HELP facility of IBEX. If no FPT is specified, the Step Condition Code and the error code are both set to zero. The memory in use by the terminating program is released; post association of a debugger is not possible if a program is terminated via M$EXIT.

No FPT is required by M$EXIT but one may be specified to provide for an explicit setting of the Step Condition Code or HELP error code if desired.

The form of the call for this service is:

CALL M$EXIT;

or

CALL M$EXIT (FPT_EXIT) [ALTRET (label)];

When the FPT is specified, the parameters are as follows:

CODE = VARIABLE specifies the 1-word area in memory containing the error code to be returned to the user when requested via the HELP facility of IBEX. The VLP_ERRCODE macro may be used to generate a structure for the error code. The severity field in the error code should reflect the lowest level of the error message that is to be reported by IBEX in response to '?'. The default is NIL.

STEPCC = {OK|ERROR|ABORT} specifies how the Step Condition Code is to be set: OK=0, ERROR=4, ABORT=6. The parameter may also be specified as a decimal value (0-511). The default is 0.

M$ERR - Error Job Step

The M$ERR service provides for error termination of the current job step. M$ERR passes control as specified for M$EXIT. In addition, M$ERR sets the Step Condition Code to 4 if no FPT is specified. M$ERR sends an appropriate message to the listing device if the CODE option is not specified. The memory used by the terminating program is not released until the initiation of the next job step; post-association of a debugger is therefore allowed with a program that terminates via M$ERR.

At error termination of a program (STEPCC>=4) for a batch job, processing proceeds to the next job step, unless there is an overriding IF command in the command file (see CP-6 Programmer Reference for details). At error termination in the course of an online job, the Command Processor returns to the user for the next command.

No FPT is required by M$ERR, but one may be specified if explicit setting of the Step Condition Code or HELP error code if desired. The form of the call for this service is:

CALL M$ERR;

or

CALL M$ERR (FPT_ERR);

When the FPT is specified, the parameters are as follows:

CODE = VARIABLE specifies the 1-word area in memory containing the error code to be returned to the user when requested via the HELP facility of IBEX. The VLP_ERRCODE macro may be used to generate a structure for the error code. The severity field in the error code should reflect the lowest level of the error message that is to be reported by IBEX in response to '?'. The default is NIL.

STEPCC = {OK|ERROR|ABORT} specifies how the Step Condition Code is to be set: OK=0, ERROR=4, ABORT=6. The parameter may also be specified as a decimal value (0-511). The default is 0.

M$XXX - Abort Job

The M$XXX service allows the user to abort the current job. M$XXX passes control as specified for M$EXIT. In addition, M$XXX sets the Step Condition Code to 6 if no FPT is specified. M$XXX also sends an appropriate message to the listing device if the CODE option is not specified. The memory in use by the terminating program is not released prior to entry to the associated command processor. Post-association of a debugger with a program that terminates via M$XXX is allowed.

At a program abort condition (STEPCC>=6), the current job step is aborted. For a batch job, all subsequent job steps are also aborted, unless there is an overriding IF command in the command file. An online user is returned to the Command Processor for the next command.

No FPT is required for M$XXX, but one may be specified if explicit setting of the Step Condition Code or HELP error code if desired.

The form of the call for this service is:

```
CALL M$XXX;
```

or

```
CALL M$XXX (FPT_XXX) [ALTRET (label)];
```

When the FPT is specified, the parameters are as follows:

CODE = VARIABLE specifies the 1-word area in memory containing the error code to be returned to the user when requested via the HELP facility of IBEX. The VLP_ERRCODE macro may be used to generate a structure for the error code. The severity field in the error code should reflect the lowest level of the error message that is to be reported by IBEX in response to '?'. The default is NIL.

STEPCC = {OK|ERROR|ABORT} specifies how the Step Condition Code is to be set: OK=0, ERROR=4, ABORT=6. The parameter may also be specified as a decimal value (0-511). The default is 0.

M$ALIB - Associate Library/Processor

In addition to the options provided by the linker to associate shared libraries and Alternate Shared Libraries, the monitor provides the capability to associate or disassociate these libraries and also debuggers at run-time.

The M$ALIB service allows a user program to control the association of a shared library, Alternate Shared Library or a debugger with his user program.

If the processor to be associated is a shared library and requires disassociation of the current library, both association and disassociation will take place and the alternate return is taken. If ALTRET is not specified, the condition is ignored. Note, however, that if the processor to be associated is an Alternate Shared Library or a debugger and a different processor of the same type is currently associated, the alternate return is taken. If ALTRET is not specified, the program is errored.

If a debugger is to be associated, the M$ALIB FPT may specify a command to be passed to the debugger. DELTA's interface with M$ALIB is described in the DELTA reference manual.

If a processor with the specified name cannot be found or if either virtual memory or physical memory is not available, the M$ALIB ALTRET code sequence is entered.

The form of the call for this service is:

CALL M$ALIB (FPT_ALIB) [ALTRET (label)];

The parameters for this service are as follows:

CMD = VARIABLE specifies the area containing a TEXT command to be passed to the debugger. This option is ignored if associating a shared library or Alternate Shared Library.

DLIB = {YES|NO} specifies, if YES, that the debugger is to be disassociated from the user after processing the command that has been specified via the CMD parameter. This option is ignored if the program making the request is executing under control of the debugger; (i.e., the program was started under control of the debugger or the debugger was invoked via the Command Processor). The default is NO.

This option is ignored if the M$ALIB request is to associate a shared library or an Alternate Shared Library.

ECHO = {YES|NO} specifies, if YES, that the command specified via the CMD parameter is to be echoed through the M$DO DCB. The default is NO.

This option is ignored if the M$ALIB request is to associate a shared library or an Alternate Shared Library.

LIBNAME = VARIABLE locates an area containing the TEXTC run-unit name of the processor or library to be associated. This area can be generated by invoking the VLP_NAME macro. The default is NIL.

QUIET = {YES | NO} specifies, if YES, that the debugger is to write only error messages to the user. The default is NO.

This option is ignored if the M$ALIB request is to associate a shared library or an Alternate Shared Library.

REPLY = VARIABLE locates an area into which the debugger may store a reply to some special commands. This option is ignored if associating a shared library or Alternate Shared Library.

RETRN = {YES|NO} specifies, if YES, that the debugger is to return to the user after processing the single command that has been specified via the CMD parameter. The default is YES.

This option is ignored if the M$ALIB request is to associate a shared library or an Alternate Shared Library.

M$DLIB - Disassociate Library/Processor

The M$DLIB service allows a user program to disassociate a shared library, alternate shared library or a debugger.

If the processor to be disassociated is an alternate shared library or a debugger and that processor has established exit control, the processor's exit control logic will be entered. Should the processor exit his exit control logic other than via M$EXIT, the M$DLIB ALTRET code sequence is entered with the TCB frame containing an error code set to reflect the processor's exit condition. Note, however, that the processor will have been disassociated.

If the processor to be disassociated is a debugger and the program making the request is executing under control of that debugger, i.e. the program was started under control of the debugger or the debugger was invoked via the Command Processor, the disassociation will not take place. DELTA's interface with M$DLIB is described in detail in the "Operational Considerations" appendix to the DELTA reference manual.

If a processor with the specified name cannot be found or if the specified processor is not currently associated, the M$DLIB ALTRET code sequence is entered.

The form of the call for this service is:

CALL M$DLIB (FPT_DLIB) [ALTRET (label)];

The parameter for this service is as follows:

LIBNAME = VARIABLE locates an area containing a TEXTC string designating the processor to be disassociated. This area can be generated by invoking the VLP_NAME macro. The default is NIL.

M$UNSHARE - Unshare Library or Program

The M$UNSHARE service allows the user to unshare a program if it is currently being shared. The FPT will specify if the program or the shared library is to be unshared.

The form of the call for this service is:

CALL M$UNSHARE (FPT_UNSHARE) [ALTRET (label)];

The parameters for this service are as follows:

LIB = {YES|NO} Indicates if the currently associated shared library is to be unshared. The default is NO.

PROG = {YES|NO} specifies, if YES, that the current running program is to be unshared. The default is NO.

Program Structure Control

The monitor provides services to transfer control within the user program or to transfer control to the command processor. The CP-6 system permits the user to structure code as follows:

- As a single run unit which can be loaded in its entirety into available memory in the Instruction Segment. Monitor services described next are not required in such a program.
- As multiple run units. In the course of a single job step, several run units can be called. For instance, the user program may call other programs or processors such as SORT. The called program completely replaces the calling program in memory. The M$LINK and M$LDTRC monitor services are available to transfer control between programs. Note that a program may determine if it was called via M$LINK or M$LDTRC by inspecting B$JIT.PROG_ENTRY.
- As a single overlaid run unit. The user program contains a root node which is always memory-resident, and a number of overlays. An overlaid program has a well-defined tree structure, as discussed in the CP-6 Programmer Reference Manual. The M$OLAY service transfers control between overlays.

M$OLAY - CALL/CANCEL Overlay

The M$OLAY service calls or releases a specified overlay. For a program linked with the NOBREF option, the user must load and transfer control to overlays via explicit calls to M$OLAY. (When the BREF option is used, the linker supplies all calls to M$OLAY.)

M$OLAY with CANCEL=NO brings in both the data and procedure portions of the overlay, unless they are already in memory. If CANCEL=YES, the routine releases the overlay and its forward path. If an error occurs while executing the OLAY routine or if the specified overlay is not found, the alternate return is taken. (If no ALTRET is specified, the job is aborted.)

The form of the call for this service is:
follows:

CALL M$OLAY (FPT_OLAY) [ALTRET (label)];

The parameters for this service are as follows:

CANCEL = {YES|NO} specifies, if NO, that the overlay is to be loaded into memory along with its backward path. YES specifies that the overlay and its forward path are to be released. However, if the overlay was brought in with the NOPATH = YES option, only the overlay itself is released. The default is NO.

ENTER = {YES|NO} is meaningful only when CANCEL=NO. ENTER=YES specifies that M$OLAY is to transfer control to the first ENTDEF in the overlay; the X1 register is set to the location of the instruction following the call to M$OLAY. When writing in PL-6, the programmer should specify ALTRET on the call to M$OLAY and ALTRET on the PL-6 overlay procedure to insure proper return when the overlay exits.

If ENTER=NO, M$OLAY calls in the overlay but returns control to the instruction following the call to M$OLAY, not to the called overlay. The default is NO.

NOPATH = {YES|NO} specifies, if YES, that no overlays on the backward path of the called overlay are to be brought in. If NOPATH=NO, M$OLAY brings in the called overlay and its entire backward path. (Cancelling an overlay obtained with NOPATH=YES releases only that overlay.) The default is NO.

ONAME = VARIABLE specifies the area containing a TEXTC string designating the desired overlay. The area can be generated by invoking the VLP_NAME macro. (An overlay name is limited to 31 characters in length.) The default is NIL.

M$LINK - Link to Program

The M$LINK monitor service causes the calling program's memory (procedure, program data and data segments - except the common segment) to be saved on disk storage making the calling program's memory available for the called program. The called program is then loaded into memory and control is transferred to it. When that program is terminated, control is returned to the program making the M$LINK request.

M$LINK may only be issued from the user domain. Standard shared processors may use M$LINK; special shared processors may not.

Any communication between the calling and called programs must be accomplished through the common data segment or the Control Command Buffer. The called program may determine that it has been entered via M$LINK as the value of B$JIT.PROG_ENTRY will have been set to %PE_LINK#.

The program to be put into execution may be specified either by using the CP_CMD option or by using the NAME, ACCT, PASS, and PSID options. The choice of which to use will depend on whether the calling program wishes to convey DCB assignment information over the M$LINK process.

If the CP_CMD option is used, the command processor will parse the command line and build the assign merge records as specified if the command is presented in standard syntax. When the linked to program is entered the SI, UI, OU, LS and NSSYNTAX bits in B$JIT.PRFLAGS will be set accordingly.

If the NAME, ACCT, PASS, PSID, and CMD options are used, the command will not be parsed by the command processor. The NSSYNTAX bit will be reset in B$JIT.PRFLAGS; the NOSCAN bit will be set in B$JIT.PRFLAGS. If the requested run unit was built with the STDINVOC option, that run unit will not be put into execution; the M$LINK ALTRET code sequence will be entered.

The assign/merge records for #1, #2, #3 and #4 will be written to the image file as part of the M$LINK process.

If the CP_CMD option was specified on the M$LINK command, these records will then be deleted from the *A file and PRFLAGS will be reset. When an image file is restored, these assign/merge records will be re-written into the *A file.

If the NAME option is used, the records are not deleted from the *A file and PRFLAGS will not be reset. The NSSYNTAX and NOSCAN bits in PRFLAGS will be set.

If the specified program cannot be put into execution (i.e. the program cannot be found, the specified program is not a valid program, or an I/O error occurs while attempting to read the program), the ALTRET will be taken or the job will be aborted if an ALTRET was not specified.

A normal return is made to the calling program when the called program exits normally.

When the M$LINKing program is restored, the SI, UI, OU, LS, NSSYNTAX and NOSCAN bits in B$JIT.PRFLAGS will be restored to the values that existed at the time of the M$LINK. The processor privilege bits (B$JIT.PRIV.PRC) will also be restored to the values that existed at the time of the link. B$JIT.PRIV.ACTIVE will be set to those that were active at the time of the link ORed with those that may have been set via an IBEX command while the linked to program was active (B$JIT.PRIV.JOB). The contents of B$JIT.CCBUF, .CCARS and .CCDISP will be restored unless the linked to program was aborted via the user typing a YC - QUIT sequence or any command processor command that implied QUIT.

The following terminal attributes for M$UC are restored to the values as they existed at the time of the link:

M$PROMPT - all options
M$EOM - all options
M$STRMCTL - all options except TABSIM, TABRELATIVE,
          SPACEINSERT, and SENDBKSPACE
M$DEVICE - HDR option.
LDEV - TITLE option

If the called program terminates via M$EXIT, M$ERR or M$XXX specifing an error code the calling program will be re-entered at the M$LINK ALTRET code sequence. The severity of the error code in the ALTRET frame will remain as set by the linked to program, which may or may not be zero. The CP_EXIT# bit will be set in B$JIT.CPFLAGS1.

If the called program aborts for any reason (i.e. calls M$ERR or M$XXX with no error code, calls M$MERC or M$MERCS, or encounters a monitor service error or a fault for which no handling is specified), the calling program will be re-entered at the M$LINK ALTRET code sequence. The CP_EXIT# bit will be reset in B$JIT.CPFLAGS1. The severity of the error code in the ALTRET frame will have been reduced to zero. Thus the calling program's ALTRET code sequence can determine if the requested program was put into execution or not by inspecting the value of the error code severity. However, note that if the the called program aborts and the calling program had not specified ALTRET on the M$LINK request, the calling program will not be aborted as the severity of the error has been set to zero.

The return to the calling program will be to that program's exit control procedure if any of the following conditions should occur while executing the linked to program:

1. When the running user was the target of an operator "X" key-in.

2. When a line disconnect occurs while executing the called program and the user is not "autosaved", or if his program image was saved but he allowed it to time out or explicitly deleted it.

3. When the linked-to program enlarged the size of the common data segment such that restoring the calling program has caused the authorized memory limit to be exceeded.

4. When the job step has been aborted via a Control-Y QUIT command.

Note that if any of these conditions should occur and the calling program does not have exit control, the calling program will be aborted.

Unlike M$LDTRC, the issuing of an M$LINK will not cause control to be passed to the effective Exit Control procedure of the calling program.

If an Alternate Shared Library is associated with the program calling M$LINK, the ASL remains associated when the calling program is restored.

If a debugger is associated with the user, the debugger will also be associated with the called program, unless that program is execute only. In any case, the debugger will remain associated when the calling program is restored.

The form of the call for this service is:

CALL M$LINK (FPT_LINK) [ALTRET (label)];

The parameters for this service are as follows:

ACCT = VARIABLE specifies the location of a TEXT string consisting of eight characters, designating the account from which the program is to be obtained. The area may be generated by invoking the VLP_ACCT macro. This parameter is ignored if the CP_CMD option is specified. The default is NIL.

CMD = VARIABLE specifies the location containing a string of up to 256 characters that is to be passed in the Control Command Buffer to the called program. This string may be in either TEXT or TEXTC format as specified by the TEXTC option. This parameter is ignored if the CP_CMD option is specified. If neither the CMD nor CP_CMD option is specified, the TEXT name of the program specified via the NAME option will be moved to the Control Command Buffer. Default = NIL.

CP_CMD = VARIABLE specifies the location containing the string of up to 256 characters that is to be placed in the Control Command Buffer and parsed by the associated command processor. This command must be in the standard program invocation format if the program to be put into execution has been linked using the STDINVOC option. In any case, the command must begin with the fid of the requested program. The command may be in either TEXT or TEXTC format as specified by the TEXTC option.

When the CP_CMD option is specified, all other options except for TEXTC and ECHO are ignored.

Default = NIL.

NAME = VARIABLE specifies an area containing a TEXTC string of up to 31 characters, designating the program to which control is to be transferred. The area may be generated by invoking the VLP_NAME macro. This parameter is ignored if the CP_CMD option is specified. Default = NIL.

PASS = VARIABLE specifies location of a TEXT string of eight characters, designating the password associated with the program. The area may be generated by invoking the VLP_PASS macro. This parameter is ignored if the CP_CMD option is specified. The default is NIL.

PSID = VARIABLE specifies location of a six-character TEXT string, designating the identification of the pack set on which the program is located. This parameter is ignored if the CP_CMD option is specified. The default is NIL.

TEXTC = {YES|NO} specifies if the command specified via either the CMD or CP_CMD option is in TEXTC (if YES) or TEXT (if NO) format. Default = YES.

M$LDTRC - Load and Transfer to Program

The M$LDTRC monitor service causes the calling program's memory (procedure, program data and data segments - except the common segment) to be released making the calling program's memory available for the called program. The called program is then loaded into memory and control is transferred to it. There is no return to the calling program.

If Exit Control has been established in the calling program, the issuing of an M$LDTRC will cause control to be passed to the effective Exit Control procedure. The M$LDTRC occurs on exit from the Exit Control procedure, via M$EXIT or M$TRTN.

If an Alternate Shared Library is associated with the user program, control is transferred to the ASL exit control procedure (if any). The ASL is disassociated from the user as part of the M$LDTRC process.

M$LDTRC may only be issued from the user domain. Standard shared processors may use M$LDTRC; special shared processors may not.

The program to be put into execution may be specified either by using the CP_CMD option or by using the NAME, ACCT, PASS, and PSID options. The choice of which to use will depend on whether the calling program wishes to convey DCB assignment information over the M$LDTRC process.

If the CP_CMD option is used, the command processor will parse the command line and build the assign merge records as specified if the command is presented in standard syntax. When the linked to program is entered the SI, UI, OU, LS and NSSYNTAX bits in B$JIT.PRFLAGS will be set accordingly.

If the NAME, ACCT, PASS, PSID, and CMD options are used, the command will not be parsed by the command processor. NSSYNTAX bit will be reset in B$JIT.PRFLAGS; the NOSCAN bit will be set in B$JIT.PRFLAGS. If the requested run unit was built with the STDINVOC option, that run unit will not be put into execution and the job step will be aborted.

The called program may determine that it has been entered via M$LDTRC as the value of B$JIT.PROG_ENTRY will have been set to %PE_LDTRC#.

If a debugger is associated with the user, the debugger will also be associated with the called program, unless that program is execute only.

The M$LDTRC process requires up to 118 words in the user's TCB under any of the following conditions:

- o Exit control has been established.
- o An ASL is associated.
- o The user program is executing under DELTA.

If none of these conditions exist, there is no TCB requirement.

If the called program is not found or an I/O error occurs in executing the LDTRC routine, the job is aborted.

The form of the call for this service is:

CALL M$LDTRC (FPT_LDTRC) [ALTRET (label)];

The parameters for this service are as follows:

ACCT = VARIABLE specifies an area containing a TEXT string consisting of eight characters, designating the account from which the program is to be obtained. The area may be generated by invoking the VLP_ACCT macro. This parameter is ignored if the CP_CMD option is specified. The default is NIL.

CMD = VARIABLE specifies the location containing a string of up to 256 characters that is to be passed in the Control Command Buffer to the called program. This string may be in either TEXT or TEXTC format as specified by the TEXTC option. This parameter is ignored if the CP_CMD option is specified. If neither the CMD nor CP_CMD option is specified, the TEXT name of the program specified via the NAME option will be moved to the Control Command Buffer. The default is NIL.

CP_CMD = VARIABLE specifies the location containing the string of up to 256 characters that is to be placed in the Control Command Buffer and parsed by the associated command processor. This command must be in the standard program invocation format if the program to be put into execution has been linked using the STDINVOC option. In any case, the command must begin with the fid of the requested program. The command may be in either TEXT or TEXTC format as specified by the TEXTC option.

When the CP_CMD option is specified, all other options except for TEXTC and ECHO are ignored.

Default = NIL.

NAME = VARIABLE specifies an area containing a TEXTC string consisting of up to 31 characters, designating the program to which control is to be transferred. The area may be generated by invoking the VLP_NAME macro. This parameter is ignored if the CP_CMD option is specified. The default is NIL.

PASS = VARIABLE Specifies the location of a string consisting of eight characters, designating the password associated with the program. The area may be generated by invoking the VLP_PASS macro. This parameter is ignored if the CP_CMD option is specified. The default is NIL.

PSID = VARIABLE Specifies the location of a 6-character string designating the identification of the pack set on which the program is located. This parameter is ignored if the CP_CMD option is specified. The default is NIL.

TEXTC = {YES|NO} specifies if the command specified via either the CMD or CP_CMD option is in TEXTC (if YES) or TEXT (if NO) format. Default = YES.

M$YC - Simulate CONTROL-Y

The M$YC monitor service simulates a Control-Y sequence and causes control to be given to the Command Processor so that a specific command can be processed.

If the Command Processor does not accept M$YC or CMD=NIL, the alternate return is taken. The Command Processor can also signal the monitor to take the alternate return. The error code specified by the command processor will be in the TCB ALTRET frame.

If the command is completed successfully, the normal return is taken, except if the command is QUIT or OFF (in IBEX). In this case, the program enters standard exit processing.

The form of the call for this service is:

```
CALL M$YC (FPT_YC) [ALTRET (label)];
```

The parameters for this service are as follows:

CMD = VARIABLE specifies an area containing a TEXT string of up to 256 characters that is passed to the Command Processor. This string cannot contain a continuation character.

Default = NIL.

ECHO = {YES|NO} specifies, if YES, that the command specified via the CMD parameter is to be echoed through the M$LL DCB. The default is NO.

Default = NO.

LINK = {YES|NO} specifies, if YES, that if the command would normally require running down the current run-unit, then the environment of the current run-unit is to be saved on disk storage prior to processing the command; i.e. the M$LINK process is to be simulated. If NO is specified, and the command would require the current process to be run down, the M$YC ALTRET procedure will be entered.

Default = NO.

NOERR = {YES|NO} specifies, if YES, that any error detected while processing the CMD parameter is not to be printed through the M$DO DCB. The default is NO (i.e., print the error message). When an error is detected by the IBEX Command Processor, the ALTRET to the M$YC Service Request will be taken.

Default = NO.

REPARSE = {YES|NO}. The reparse option is used only when the command specified by the CMD option is known to contain an error. REPARSE=YES specifies that the Command Processor is to print the legal syntax alternatives through the M$LL DCB and to take no other action on the command.

Default = NO.

REPLY = VARIABLE specifies an area into which the Command Processor may store a reply to the command. When the command is processed by the IBEX command processor, the VLR_YC macro should be used to generate the area.

Default = NIL.

VLR_YC

The VLR_YC macro generates storage that may be used as a REPLY area for the M$YC monitor service.

Fields in the structure are as follows:

CPOS = VALUE-UBIN. This field will contain an index into the command that indicates where an error was detected.

This field is valid only on the alternate return to M$YC.

M$CMDVAR - Manipulate Command Variables

M$CMDVAR service manipulates command variables. The variables are stored in character format in the user-inaccessible file, *S. This service stores, fetches, and deletes command variables from the *S file.

The form of the call for this service is as follows:

CALL M$CMDVAR (FPT_CMDVAR) [ALTRET (label)];

The parameters for this service are as follows:

FUN = {STORE|FETCH|DELETE|LIST}. STORE specifies that the contents of VALUE are to become the new value of the NAMEd command variable. FETCH fills VALUE with the current value of the NAMEd command variable. DELETE removes the NAMEd command variable. LIST returns in the VALUE field the name of the command variable following the NAMEd one.

The fully qualified name of this field is FPT_CMDVAR.V.FUN#. Acceptable EQUs are CMD_STORE#, CMD_FETCH#, CMD_DELETE#, and CMD_LIST#.

NAME = VARIABLE locates a TEXTC area containing the name of the command variable to be affected. The name is limited to 31 characters and must start with an alphabetic character.

PACCT = VARIABLE. Locates a VLP_ACCT to be used for private (processor local) command variables.

PNAME = VARIABLE. Locates a VLP_NAME to be used for private (processor local) command variables.

PPSN = VARIABLE. Locates a VLP_SN to be used for private (processor local) command variables.

PRIVATE = {YES|NO}. YES specifies that the command variable is to be local to the calling processor and not an IBEX command variable. If YES is specified, PNAME, PACCT and PPSN tell which processor's private command variables are to be accessed.

VALUE = VARIABLE locates a TEXTC string to be used as a value on STORE operations and filled on FETCH or list operations. The string may be up to 511 characters in length. There are no restrictions on content.

Note that longer command variable values (or those containing special characters) may cause problems if invoked in the IBEX command stream. They may, however, be used to pass instructions or parameters to other routines within the JOB.

Program Save

The monitor provides facilities to interrupt program execution, save the program image, later restore it, and restart program execution. The SAVE/GET facilities save and restore the essential portions of the user's work space (as discussed in detail later). The SAVE process is available via the M$SAVE monitor service or the IBEX SAVE command. The GET process is performed via the IBEX GET command.

The SAVE/GET facility provides these optional features:

- o The user program may request that its exit control procedure receive control prior to the SAVE, to do cleanup and its own checkpointing. (See M$SCON with XCON=YES.)
- o If the preceding feature is used, at a GET command the program is restarted at its exit control procedure with a code indicating that a GET occurred. The exit control procedure can restore any checkpoint data prior to resuming execution at the point of interruption.
- o The user program may save itself at appropriate points by calling M$SAVE and continue execution.
- o The user program may prevent program SAVEs by calling M$SCON with SAVEFLG=NO.

The M$SAVE and M$SCON services and also the GET process are discussed next. The IBEX SAVE and GET commands are discussed in the CP-6 Programmer Reference Manual.

M$SAVE - Save Program

The M$SAVE service interrupts program execution and optionally gives control to the user's exit control procedure before the program image is saved. The contents of the SAVE file are discussed later in this section.

NOTE: The M$SAVE service may be called by the user program only, not by special shared processors. However, if an Alternate Shared Library is associated with the user program, control is transferred to the ASL exit control routine (if any) prior to the SAVE. The ASL is disassociated from the user program as a part of the SAVE process.

The SAVE process requires 87 words in the user's TCB under any of the following conditions:

- o Exit Control for SAVE is requested.
- o An ASL is associated.
- o The program is executing under DELTA.

If none of these conditions exist there is no TCB requirement.

When the SAVE occurs without entry to the user's exit control procedure, execution will resume at the point of interruption. More detail on SAVE with entry to the user's exit control procedure is included below.

If a SAVE is attempted and the user previously called the M$SCON service with SAVEFLG=NO, the alternate return is taken from M$SAVE. If the user requested exit control before the SAVE and the TCB is full, the alternate return is taken from M$SAVE.

The form of the call for this service is:

```
CALL M$SAVE (FPT_SAVE) [ALTRET (label)];
```

The parameters for this service are:

ACCT = VARIABLE specifies location of a TEXT string consisting of eight characters, designating the account in which the SAVE file is to be created. This area may be generated by invoking the VLP_ACCT macro. The default is NIL.

NAME = VARIABLE specifies an area containing a TEXTC string consisting of up to 31 characters, designating the name of the SAVE file to be created. This area may be generated by invoking the VLP_NAME macro. The default is NIL.

PASS = VARIABLE specifies location of a TEXT string consisting of eight characters, designating the password associated with the SAVE file. This area may be generated by invoking the VLP_PASS macro.

PSID = VARIABLE specifies location of a TEXT string consisting of six characters, designating the identification of the packset on which the SAVE file is to be located. The default is NIL.

XCON: SAVE Files, GET Process

XCON Before SAVE

When a SAVE is requested and the user previously called M$SCON with XCON=YES, the user's exit control procedure takes control with a code set to indicating that a SAVE was requested. (B$XCON.CECCB.SAVE if M$SAVE; B$XCON.CECCB.CPSAVE if Control-Y SAVE). The actual SAVE of the program image occurs on exit from this routine, when M$EXIT or M$TRTN is called.

The user must be aware of several rules which apply to monitor service calls in the user's exit control procedure following a SAVE or GET request:

1. To suspend a program after it is saved, the user's exit control procedure calls M$EXIT:

   - o After the program has been saved as a result of a Command Processor SAVE request, control returns to the Command Processor without running down the user program. A subsequent GO command causes the same results as if a GET command had been issued.

    - After the program has been saved as a result of an M$SAVE monitor service request, the user program is run down. A GET command is required to restart the program.

2. To continue the program immediately after it is saved, the user's exit control procedure calls M$TRTN.

    - After the program has been saved as a result of a Command Processor SAVE request, control returns to the Command Processor without running down the user program. A subsequent GO command causes the same results as if a GET command had been issued.

    - After the program has been saved as a result of an M$SAVE monitor service request, the user's exit control will again be entered. B$XCON.CECCB.GET will be set.

3. When entered with a code indicating a GET command occurred, the user's exit control procedure returns control to the user program at the point of interruption for the SAVE, when the M$TRTN call is issued.

4. M$LINK and M$LDTRC are not permitted in an exit control procedure entered because of a SAVE.

SAVE File

The SAVE file contains Read-Only Segment data, program data, program procedure, program dynamic data (M$GDP or M$GVP pages), user and debugger data segments. A header record is also written to the file which contains various control information, including the names (not processor numbers) of all associated processors. Pages mapped into the user work space by M$CVM are saved and restored. That is, the physical-to-virtual mapping is preserved, but the contents of the pages are not written to the file.

An associated debugger is remembered and its data is saved. The debugger is associated and the debugger data is restored upon a GET.

GET Process

At an IBEX GET command, the monitor opens the SAVE file specified. The system version of the file's header is checked against that of the running system and the run unit that was in execution at the time of the SAVE is reopened. If the program was saved under an older version of the operating system, or if the run unit is not the same version, the image cannot be restored and the GET process must be aborted. Otherwise, the user's saved program image is restored in a process similar to that used by the return path for M$LINK.

DCBs that were open at the time of the SAVE will be closed when the restored program is given control. File repositioning is not performed. This is one of the functions the user program must perform itself in Exit Control, if necessary.

When the program image has been restored, the program resumes execution at the SAVE point or at its XCON procedure, depending on how the SAVE was performed.

M$SCON - Set Save Control

The M$SCON service establishes parameters for later use, when a request occurs to save the program image.

The form of the call for this service is:

CALL M$SCON (FPT_SCON) [ALTRET (label)];

The parameters are as follows:

SAVEFLG = {YES|NO} specifies, if YES, that the program may be saved. An Alternate Shared Library or any program which does not wish to allow SAVEs may call M$SCON specifying SAVEFLG=NO. This specification may not be reset once NO is specified. The default is YES.

XCON = {YES|NO} specifies, if YES, that the user's exit control procedure is to be entered prior to a SAVE. This parameter applies to user programs only, as the ASL exit control procedure is always entered on a SAVE. The default is NO.

Memory Control

CP-6 memory management fully utilizes the Virtual Memory and Security feature of the hardware. The CP-6 Concepts and Facilities Manual section entitled Memory Management describes memory protection, virtual address calculation, hardware registers associated with specific portions of the virtual work space, and the concept of domain which derives from the VM&S feature. Instead of reiterating that information, the following discussion focuses on concepts necessary to perform dynamic memory allocations.

Memory is managed as a number of work spaces. Each user runs in a separate work space as do the monitor, the command processor, the debugger, and the alternate shared library. Each work space is a 1024K word virtual address space protected by the NSA hardware at two levels:

- At the page level - The Page Table in the JIT contains an entry for each possible 1K word page in the virtual space. The Page Table entries indicate whether a physical page corresponds to the page table entry and contain flags which control access to the page.
- At the segment level - The work space is organized in segments, that is, contiguous virtual spaces framed by structures called descriptors. The descriptors (stored in the Linkage Segment) contain flags to indicate if the segment is present and to control access to the segment. A SEGID, which is part of a PL-6 pointer variable, identifies a particular descriptor.

A map of the user work space included in Table 4-1 shows the segments into which the virtual memory space is divided. Segments are reserved for job-related tables and structures managed by the monitor, and for the Instruction Segment which contains user data, user procedure, and overlays and also library data and procedure (if used). In addition, virtual space is available for dynamic allocation in the course of program execution. This space may be used for tables, user buffers, etc.

The virtual memory available for dynamic allocation is located in the Instruction Segment and in separate, individual Data Segments in the high-address portion of the virtual work space. The characteristics of the two sources of dynamic memory are described below:

1. Dynamic Data Area of the Instruction Segment.
   The size of this area is dependent on the size
   of the user program and on whether a memory-resident
   library is associated with the program. Because
   the Dynamic Data Area resides in the same segment
   as the user program, dynamic pages in this
   area can be referenced within the user
   program by use of name references or simply by
   their Instruction Segment address. The MGDP/MFDP
   services permit the user to allocate pages as
   a stack appended to the user procedure. The
   MGVP/MFVP services permit the user to allocate
   and deallocate pages anywhere in the Dynamic
   Data Area. The user is responsible for
   managing the Dynamic Data Area, although monitor
   services prevent reallocation of pages that are
   already allocated. Pages can be used for
   read, write, or execute functions.

2. Data Segments. Each user and each special shared processor may request up to eight data segments at run-time. These data segments are virtual spaces which may be created, enlarged, and deallocated under programmer control. Generally, managing a data segment requires less user coding than managing space in the Dynamic Data Area. The monitor determines the location of the segments within the virtual work space. Since data segments are independently addressable segments, the user must reference Data Segments via PL-6 pointer (PTR) variables. Pages can be used for read or write functions but not for execution.

The virtual memory map in Table 4-1 gives the maximum sizes of the Dynamic Data Area in the Instruction Segment and the combined maximum sizes of the Data Segments for each domain. Dynamically allocated pages are set to allow read and write access.

This table can be viewed as the layout of pages in virtual memory or as the layout of words in the page table. Some of the choices have been made to attempt to minimize the collisions within the Page Table Word Associative Memory (PTWAM). These are shown below:

Field	V.P.#	AM row #
JIT	.13	.13
HJIT	.15	.15
ROSEG	.57	.17
IS	.100	.0
UDS	.1050	.10

The more important parts of the structure include the following:

PAGE_TABLE is a page containing the user page table.

JIT_PAGE is a page containing JIT and monitor AUTO.

HJIT is the HJIT page (housekeeping) containing:

- User Linkage Segment
- Command Program Linkage Segment
- Debugger Linkage Segment
- ASL Linkage Segment
- Special Descriptor Storage
- Safe Store Stack
- Argument Segment
- Parameter Segment

ROSEG contains DCBs, TCB, and tree table.

INSTRUCTION_SEGMENT contains the following:

- Library data
- Program data
- Program procedure
- Dynamic data
- Shared library procedure

Several of the monitor services discussed next return a vector to the user. The vector is a structure which conforms to the requirements of the NSA hardware. The vector is said to "frame" an area of virtual memory, that is, the 2-word vector contains the bound of the area (byte size minus one) in the first word along with control information which is not of concern to the user program, and in the second word a PL-6 pointer to the start of the framed area. To access an area allocated by the M$GDP or M$GDS service, the user may simply use the PL-6 pointer from the returned vector. The pointer contains both an offset and a segment identifier (SEGID). For details on the format of the vector, see the discussion of the VLP_VECTOR macro included in this section.

Table 4-1. User Virtual Memory

```
         012345678 012345678 012345678 012345678
        +---------+---------+---------+---------+DCL
     .0|         |         |         |         |1 USER_VIRTUAL_MEMORY,
     .0|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 PAGE_TABLE(0:0) UBIN,
     .1|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 RFU1(0:9)       UBIN,
       z         z         z         z         z
    .13|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 JIT_PAGE(0:0)   UBIN,
    .14|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 RFU2(0:0)       UBIN,
    .15|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 HJIT(0:0)       UBIN,
    .16|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 RFU3(0:1)       UBIN,
    .20|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 FPOOL(0:30)     UBIN,
       z         z         z         z         z
    .57|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 ROSEG(0:13)     UBIN,
       z         z         z         z         z
    .75|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 RFU4(0:2)       UBIN,
       z         z         z         z         z
   .100|         |         |         |         |  2 INSTRUCTION_SEGMENT
   .100|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|     (0:255)        UBIN,
       z         z         z         z         z
   .500|         |         |         |         |  2 STEP_BUFFER_1(0:0)
   .500|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|                    UBIN,
   .501|         |         |         |         |  2 STEP_BUFFER_2(0:0)
   .501|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|                    UBIN,
   .502|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 RFU5(0:5)       UBIN,
       z         z         z         z         z
   .510|         |         |         |         |  2 DEBUGGER_DATA_SEGMENTS
   .510|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|     (0:63)         UBIN,
       z         z         z         z         z
   .610|         |         |         |         |  2 ASL_DATA_SEGMENTS(0:127)
   .610|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|                    UBIN,
       z         z         z         z         z
  .1010|         |         |         |         |  2 COMMAND_PROGRAM_DATA_
  .1010|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|     SEGMENTS(0:31) UBIN,
       z         z         z         z         z
  .1050|         |         |         |         |  2 USER_DATA_SEGMENTS(0:383)
  .1050|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|                    UBIN,
       z         z         z         z         z
  .1650|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|uuuuuuuuu|  2 RFU6(0:87)      UBIN;
       z         z         z         z         z
       +---------+---------+---------+---------+
        012345678 012345678 012345678 012345678 .2000-0-0 total length
```

M$GDS - Get/Enlarge Data Segment

The M$GDS service allocates or enlarges a data segment based on the pointer variable passed to it by the user. The pointer variable is stored in the second word of a 2-word area called the RESULTS area which can be generated by the VLP_VECTOR macro (discussed later).

Allocating a New Data Segment

The next available data segment is allocated if the pointer passed to M$GDS in RESULTS contains zero or ADDR(NIL). The first such available user data segment is always data segment #3. Automatic (data segment #1) and common (data segment #2) must be requested explicitly. Since special shared processors do not have the common data segment, their first available data segment is always data segment #2.

A particular Data Segment can be requested in the case of Common and Automatic data by setting the appropriate bit in the FPT or, for all data segments, by specifying the SEGID of an unallocated Data Segment in the second word of the RESULTS area.

A data segment is always created beginning on a page boundary. (For accounting billing purposes, the charge is based on the number of pages allocated.) The requested number of words is always rounded up to the nearest double word (due to NSA hardware restrictions) before being allocated from the appropriate Data Segment area. The area is determined by the domain of the executing program, and a descriptor for the Data Segment is placed in the user's Linkage Segment.

Enlarging a Data Segment

If the 2-word field named with the RESULTS parameter contains the SEGID of an allocated Data Segment, this is interpreted as a request to increase the size of that Data Segment by the number of words indicated by the SEGSIZE parameter. If either AUTOSEG=YES or COMMON=YES, the SEGID is ignored and, should those segments exist, their size is increased.

NOTE: The AUTOMATIC and COMMON segments have certain conventional uses. The AUTOMATIC data segment is used by most Honeywell language processors. User programs should therefore avoid use of this segment unless it is known to be unused. The COMMON segment is so named because it is the only data segment which can be common to programs which call one another via the M$LINK monitor service; the COMMON data segment can thus be used to pass data.

On successful completion of M$GDS, the vector returned into the field named by RESULTS frames the entire allocated Data Segment. (Specifying SEGSIZE=0 results in a returned vector framing the currently allocated Data Segment.) If no allocation can be made, the RESULTS area is set to VECTOR(NIL) and the alternate return is taken.

NOTE: As the size of some Data Segments increases, it may be necessary to modify the base addresses of other segments. Therefore any data segment descriptors which the user has in descriptor registers may be invalidated following a call to M$GDS (or M$FDS). Such descriptors should be reloaded from the Linkage Segment following every call to M$GDS or M$FDS. (This function is handled automatically for programs using PL-6 and the Honeywell-supplied macro library.)

The form of the call for this service is:

CALL M$GDS (FPT_GDS) [ALTRET (label)];

The parameters for this service are:

AUTOSEG = {YES|NO} specifies, if YES, that the Automatic Segment is desired. The default is NO.

COMMON = {YES|NO} specifies, if YES, that the Common Segment is desired. The default is NO.

FREE = {YES|NO} specifies, if YES, that if the requested memory is not available (user or installation page limit reached), then no memory at all is to be allocated. The default (FREE=NO) allocates as much memory to the segment as possible before returning an error code indicating that the entire requested amount was unavailable.

PROTECTION = {DSREAD|DSWRITE} specifies the access protection to be placed on the entire data segment. The protection on an existing data segment may not be changed by M$GDS, however M$PDS may be used for this purpose.

RESULTS = VARIABLE specifies the location of a 2-word area in which the returned vector is to be stored. The VLP_VECTOR macro can be used to generate this area. The contents of the SEGID field in this area at the call to M$GDS determines whether an existing segment is enlarged or a new segment is allocated as described above. The default is NIL.

SEGSIZE = VALUE-DEC(1-n) specifies the number of words to get or to add to an existing data segment. The default is 0.

M$FDS - Free/Diminish Data Segment

The M$FDS service frees space from a previously allocated Data Segment. The SEGSIZE parameter is interpreted as the number of words to release from the Data Segment, indicated either by the SEGID field of the passed RESULTS variable, by AUTOSEG=YES or by COMMON=YES. The SEGSIZE parameter is rounded up to the nearest doubleword, as for M$GDS.

As with M$GDS, specifying a zero size results in a returned vector framing the currently allocated portion of the Data Segment. This vector is inserted into a two-word field named by the RESULTS parameter.

When all the space allocated to a Data Segment has been released, the returned vector will be VECTOR(NIL).

If any input-output operation is outstanding when M$FDS is called, the user is aborted.

The form of the call for this service is:

CALL M$FDS (FPT_FDS) [ALTRET (label)];

The parameters for this service are as follows:

AUTOSEG = {YES|NO} specifies that the Automatic Segment is desired. The default is NO.

COMMON = {YES|NO} specifies that the Common Segment is desired. The default is NO.

RESULTS = VARIABLE specifies the location of a 2-word area in which the returned vector is to be stored. Prior to that, RESULTS specifies the SEGID of the area to be freed or diminished.

The default is NIL.

SEGSIZE = VALUE-DEC(1-n) specifies the number of words to free, or if 0, that a vector framing the segment is to be returned in RESULTS.

The default is 0.

M$PDS - Protect Data Segment

The M$PDS procedure allows a user to change the access protection on an entire data segment. The options are to limit the data segment to read-only access or to grant it full read/write access.

The form of the call for this service is:

CALL M$PDS (FPT_PDS) [ALTRET (label)];

The parameters for this service are as follows:

PROTECTION = {DSREAD|DSWRITE} specifies the access protection to be placed on the entire data segment.

RESULTS = VARIABLE specifies the location of a 2-word area in which the vector describing the data segment to have its protection modified is to be stored. For this service, the SEGID is the only portion of the vector examined. The default is NIL.

M$GDP - Get Dynamic Pages

The M$GDP service allocates a specified number of pages beginning with the next higher page of the Dynamic Data Area within the user's Instruction Segment. It also increments that page number until one of the following events takes place:

1. The requested number of pages is allocated.
2. The installation-set or user-set limit on the number of physical pages is reached.
3. The instruction segment is full.
4. A page allocated by M$GVP or M$CVM is encountered.

Cases 2-4 result in an alternate return. In all cases, a vector is returned framing the pages that were allocated. Note that requesting zero pages will return VECTOR(NIL).

NOTE: This service is restricted to user-mode programs. Attempted use by a debugger, command processor, or Alternate Shared Library results in an alternate return.

The form of the call for this service is:

CALL M$GDP (FPT_GDP) [ALTRET (label)];

The parameters for this service are as follows:

PAGES = VALUE-DEC(1-n) specifies the number of pages to be obtained. The default is 0.

RESULTS = VARIABLE specifies the location of a 2-word area in which the returned vector is to be stored. The VLP_VECTOR macro is used to generate this area. The default is NIL.

M$FDP - Free Dynamic Pages

The M$FDP service releases a specified number of pages at successively lower addresses of the Dynamic Data Area within the user's Instruction Segment, beginning with the current highest page. It also decrements that page number until one of the following events takes place:

1. The requested number of pages has been released.

2. The first available virtual page is released.

The second case results in an ALTRET. In either case, a vector is returned framing the remaining allocated Dynamic Data Area. If all dynamic pages have been released, the returned vector will be VECTOR(NIL).

Pages released are marked as "not in memory" and any subsequent reference to these pages results in a fault.

NOTE: This service is restricted to user-mode programs. Attempted use by a debugger, command processor, or Alternate Shared Library results in an alternate return.

The form of the call for this service is:

CALL M$FDP (FPT_FDP) [ALTRET (label)];

The parameters for this service are as follows:

PAGES = VALUE-DEC(1-n) specifies the number of pages to be freed. The default is 0.

RESULTS = VARIABLE specifies the location of a 2-word area in which the returned vector is to be stored. The VLP_VECTOR macro is used to generate this area. The default is NIL.

VLP_VECTOR

The VLP_VECTOR macro creates a vector which may be used as the RESULTS area for the MGDS, MFDS, M$GDP and M$FDP services, as well as for any other situations in which a vector is desired.

For M$GDS and M$FDS, the SEGID field in the VLP_VECTOR area can contain a parameter for the monitor service. For M$GDP and M$FDP, the VLP_VECTOR is submitted to the monitor with the fields set to the defaults. On return from the monitor services, the VLP_VECTOR area contains a vector framing the allocated memory. The fields of interest to the user are the second word, BASE and SEGID, which form a PL-6 pointer by which the user can access the allocated memory. (The monitor also stores values in the VLP_VECTOR area before returning the resulting vector to the user. These fields are used by the monitor in maintaining the user's segment descriptors.)

Fields in the VLP_VECTOR area are as follows:

BASE = VALUE-DEC(0-n) specifies the byte offset from the start of the segment identified by SEGID. (BASE is used in modifying the descriptor when TYP=SHRINK.) The default is 0.

FLAGS = VALUE-BIT(9) specifies the bit mask. This bit mask is logically ANDed with the existing flags in the descriptor to update the descriptor. The default is '777'O.

PTR$ is an alternate method of referring to word two of the vector as a pointer, as its structure is identical to that of an NSA pointer.

SEGID = VALUE-BIT(12) specifies the SEGID of the segment descriptor. Segment identifiers are defined in the file B_SEGIDS_C in the :LIBRARY account. The default is '6014'O.

TYP = {COPY|SHRINK} specifies whether the descriptor identified by SEGID is to be copied or modified. When modified (by SHRINK) its BASE and/or BOUND are modified by VSIZE and BASE stored in VLP_VECTOR. The default is SHRINK.

VSIZE = VALUE-DEC(0-n) specifies the byte size minus one (PL-6 SIZEV built-in function) of the area. (If TYP=SHRINK, VSIZE is used in modifying the descriptor identified by SEGID.) The default is 0.

M$GVP - Get Virtual Page

The M$GVP service allocates a specific page of virtual memory within the Instruction Segment of the user program. If the request is allowed, access to the page is set to write-allowed. The ALTRET is taken if one of the following events takes place:

1. The installation-set or user-set limit on the number of physical pages allowed would be exceeded.

2. The page has already been allocated.

3. The page requested is outside the user Instruction Segment.

No results are returned by this service.

NOTE: This service is restricted to user-mode programs. Attempted use by a debugger, command processor, or Alternate Shared Library results in an alternate return.

The form of the call for this service is:

CALL M$GVP (FPT_GVP) [ALTRET (label)];

The parameter for this service is as follows:

PNO = VALUE-DEC(0-255) specifies the Instruction-Segment-relative virtual page number to obtain. The default is 0.

WSQ = (0-511). This specifies the working space quarter in which the page is to be allocated. Zero specifies the caller's instruction segment, non-zero specifies a WSQ previously returned on an open of a Virtual Segment.

M$FVP - Free Virtual Page

The M$FVP service releases a specific page of virtual memory within the Instruction Segment of the user program. The indicated page is released except when the request is for a page that was not allocated via M$GVP or M$GDP, in which case the ALTRET is taken. Pages of data or procedure that are linked with the program (those assembled or compiled with the program) may not be released using this mechanism. No results are returned by this service.

NOTE: This service is restricted to user-mode programs. Attempted use by a debugger, command processor, or Alternate Shared Library results in an alternate return.

The form of the call for this service is:

CALL M$FVP (FPT_FVP)) [ALTRET (label)];

The parameter for this service is as follows:

PNO = VALUE-DEC(0-255) specifies the Instruction-Segment-relative page number to free. The default is 0.

WSQ = (0-511). This specifies the working space quarter from which the page is to be freed. Zero specifies the caller's instruction segment, non-zero specifies a WSQ previously returned on an open of a Virtual Segment.

M$GDDL - Get Dynamic Data Limits

The M$GDDL service returns several limiting values of the Dynamic Data Area. The Dynamic Data Area is the region of memory in the Instruction Segment from which dynamic and sometimes virtual pages are allocated. This is the only memory management service routine which returns more than the usual NSA vector to the area named by the RESULTS parameter. M$GDDL returns three words of information. The first two are an NSA vector framing the entire Dynamic Data Area available to the calling program. The third word returned contains the number of pages (right-justified) which the program may yet obtain through any of the "get" monitor services. For a user with a large memory allocation, this value may be greater than the number of unallocated pages in the Dynamic Data Area, as pages allocated to the Data Segments will also be counted against this value.

NOTE: This service is restricted to user-mode programs. Attempted use by a debugger, command processor, or Alternate Shared Library results in an alternate return.

The form of the call for this service is:

CALL M$GDDL (FPT_GDDL) [ALTRET (label)];

The parameter for this service is as follows:

RESULTS = VARIABLE specifies the location of a three-word area which may be generated by invoking the VLP_GDDL macro. The use of this parameter is described above.

The default is NIL.

VLP_GDDL

The VLP_GDDL macro creates the three-word structure used as the RESULTS area for the M$GDDL service.

VLP_GDDL.AVAIL_PGS# will be set to the remaining number of physical pages of memory which may yet be allocated to the program issuing the call to the M$GDDL service.

VLP_GDDL.DDL_.DD$ will be set to point to the first word of the currently allocated dynamic data area.

VLP_GDDL.DDL_.DDSIZE# will be set to the byte size minus one of the currently allocated dynamic data area.

M$STD - Store Descriptor

The M$STD service allows a user to store a descriptor, shrunk from those available to him, into any of his eight special access descriptors. This allows him to establish independently addressable segments within the virtual space visible to him via those descriptors already in his linkage segment. The access rights for the new descriptor are identical to those of the descriptor being shrunk.

The form of the call for this service is:

CALL M$STD (FPT_STD) [ALTRET (label)];

Parameters for M$STD are:

AREA = VARIABLE specifies the area (PL-6 structure) for which a descriptor is to be built (shrunken). The default is NIL.

USERID = VALUE-BIT(12) specifies the 12-bit user's SEGID of the special access descriptor where the new descriptor is to be stored. The default is 0.

M$PAS - POP Argument Stack

The M$PAS service removes one or more descriptors from the hardware Argument Stack. If there are no descriptors on the Argument Stack, the alternate return is taken.

To remove one descriptor in the stack, the form of the call to this service is as follows:

CALL M$PAS [ALTRET (label)];

To remove more that one descriptor, the form of the call for this service is as follows:

CALL M$PAS (FPT_PAS) [ALTRET (label)];

The parameter for this service is then as follows:

DESCRIPTORS = VALUE-DEC(1-n) specifies the number of descriptors to remove from the Argument Stack. If the value specified is equal to or greater than the number of descriptors currently on the stack, all descriptors are removed and the stack is marked empty. Default = 1.

M$KEYIN - Operator/User Communication

The CP-6 system can interact with multiple operator consoles at different locations and ensure both proper console message switching and system security. Various designations such as SYSTEM cause messages to be routed to consoles designated responsible for the corresponding activity.

The following discussion of M$KEYIN provides adequate information to send user messages and receive text replies. For detailed information on input parsing, see Section 10 in this manual. For a discussion of KEYIN operations, see the CP-6 Operations Reference Manual.

The M$KEYIN service causes a write, read, or write-followed-by-read (keyin) operation to be performed on an operator's console, and is the only service by which a user may communicate with an operator's console.

The M$KEYIN pmme will altreturn when a results field and the no-wait option are specified. The reason being, on a no-wait operation the TYC and the ARS can be found in the no-wait I-O frame of the Task Control Block.

The form of the call for this service is as follows:

CALL M$KEYIN (FPT_KEYIN) [ALTRET(label)];

The parameters for this service are as follows:

DCB = DCBNAME specifies the name of the DCB associated with the device to which this message applies. In order for a message to be sent to a console in charge of a particular device the user must have a DCB open to the device and specify WHO=D.

EVENT = VALUE-DEC(0-?) if non-zero, specifies an event identification to be returned to the user when this operation is complete. If zero, no event is reported. EVENT is ignored (treated as if it were zero) unless WAIT=NO. The default is zero.

MESS = VARIABLE locates a buffer containing the message to be output to an operator's console. MESS=NIL means "don't write".

NOTE: All text sent to consoles is translated to the 96 ASCII graphics and TAB with all other characters outside this set translated to space.

OCTYPE = OPTION specifies the type of operator console when WHO=W. The options are as follows:

DEVICE specifies a console which receives messages about devices (e.g., ERROR, MOUNTs) and is allowed to perform device keyins (e.g., RETRY, MOUNT).

SYSTEM specifies a console which receives messages about central site operations (e.g., JACK,JSON LOGGED ON), and is allowed to perform keyins pertaining to central site operations (e.g., ZAP, START CPU 1).

ADMIN specifies a console that controls a workstation ADMIN consoles can display and alter (e.g., PRIO) only their own workstation's jobs and can control users whose workstation of origin matches this console's WSN.

TAPE specifies console attributes which presently have the same meaning as DEVICE.

DISK specifies console attributes which presently have the same meaning as DEVICE.

UNITREC specifies console attributes which presently have the same meaning as DEVICE.

ADCOM specifies a console with administrative and communications attributes.

TP specifies a console that deals with transaction processing related functions. For example, sending messages to TPAs and receiving messages about TP stations.

The default is ADMIN.

REPLY = VARIABLE locates the buffer to receive the operator's reply. The REPLY is NIL means do not read.

RESULTS = VARIABLE locates 2-word buffer where TYC and ARS are returned.

WAIT = {YES|NO} indicates whether or not this is a wait operation. The default is YES.

WHO = {W|D} indicates whether the addressed console is one associated with a workstation (W) or a device (D). W is the default.

WSN = VALUE-CHAR(8) specifies the workstation name of the workstation to which this message is directed. WSN=' ' or WSN='LOCAL' implies the main computer site. This parameter is meaningful only if WHO=W.

M$ERRMSG - Error Message Reporting

The CP-6 system provides comprehensive error reporting, which is available to report monitor service errors, shared processor errors, or errors in a user program. The error reporting facility is not limited to error messages; it can be used to output messages that conform to the requirements explained here.

In the case of monitor service errors, the monitor service passes an error code to the user, typically in the ALTRET frame of the TCB. The user calls the M$ERRMSG service specifying as parameters the error code stored in the ALTRET frame and a user buffer. The following discussion presents the full information needed by users who are reporting errors on monitor service calls. For users who intend to define their own error codes and error message file, additional information is given under later headings.

ERROR CODE

The 1-word code specified as a parameter for M$ERRMSG must conform to this format:

Field	Content
Functional Code Group	VALUE-BIT(12), that is, 2 characters composed of the low-order 6 bits of the ASCII code
Module ID	VALUE-BIT(6), that is, 1 character composed of the low-order 6 bits of the ASCII code
Monitor Flag	VALUE-BIT(1)
Error Number	VALUE-DEC(0-16383)
Severity Level	VALUE-DEC(0-7)

User's reporting monitor service errors do not alter the error code. However a processor or user can supply an error message file and manipulate the severity level to provide increasingly informative messages in response to requests from a time-sharing user. The processor or user displays one of several messages associated with the same error number but with an increasingly higher severity code (0-7) at each call to M$ERRMSG. (The HELP facility is provided through use of a message file with multiple messages associated with each error condition.)

ERROR DISPLAY

The format of the error display resulting from the M$ERRMSG service is shown below. The text is always displayed (if found); other fields are optional. If the text is not found, the error code is displayed. The full form of the display is as follows:

```
****fff-Mnnnn-s text
```

where

**** indicates the seriousness of the error by 0 to 4 asterisks. Asterisks may be omitted; however, by convention the asterisks are used as follows:

```
   * - information only
  ** - warning message
 *** - serious error
**** - fatal error
```

The FLAGLEV parameter determines the number of asterisks output by M$ERRMSG.

fff-Mnnnn-s represents the error code passed to the M$ERRMSG service as explained above. fff represents the functional code group (FCG) and Module ID (MID) codes. The letter M precedes the error number (nnnnn) for monitor service errors. The severity code(s) follows the error number. The INCLCODE parameter determines whether this code is part of the message output by M$ERRMSG.

text is the message obtained from the message file identified by the FILENAME parameter. The text may include:

- o Text characters.
- o Substitution names represented by %FN, %AC,

%DC, %U1, etc.

o Conditional fields framed by % symbols and containing the substitution character strings.

M$ERRMSG

The M$ERRMSG service obtains an error message based on the error code supplied as a parameter, and stores the message in the user buffer. M$ERRMSG allows phrase and message substitution, allows override of vertical format control, and allows suppression of portions of the error display. The output of the M$ERRMSG service is returned in RESULTS area.

The M$ERRMSG service sends the error display through one or two DCBs specified for output. Correspondence checking is performed to determine if the DCB assignments are identical. Errors encountered in writing through the OUTDCBs are indicated in the RESULTS area. If no output DCBs are specified, the M$ERRMSG service simply stores the message in the buffer and returns.

The form of the call for this service is as follows:

CALL M$ERRMSG (FPT_ERRMSG) [ALTRET(label)];

The required parameters for the service are CODE and BUF. All other parameters are optional.

ALTCODE = VARIABLE specifies the 1-word area in memory containing the error code identifying the alternate message to be read from the file. It is used to form the key to read the file if no message can be found for CODE. To generate the structure for the error code, use the VLP_ERRCODE macro described in this section; or, for monitor service errors, the B$ALT macro discussed in Section 6. The default is NIL.

BUF = VARIABLE specifies the buffer used when reading the file. BUF must have a minimum size of 120 bytes, the maximum size of any record in the file. Because some error messages may consist of multiple records in the file, a 120-byte buffer will not always be large enough to hold the whole message. The caller may alleviate this condition by specifying a larger buffer.

Records from the file are read into the buffer at 120-byte offsets until the buffer is full or the message is complete. Status is returned to the caller indicating how many records are in the buffer, and whether or not the message is complete. If the buffer is not large enough to hold all of the records of the message, then the contents of the buffer will vary depending on whether the caller supplied OUTDCBs through which to output the messages. Given a buffer with enough space for n records, if no OUTDCBs have been specified, the buffer will contain the first n records of the message. If OUTDCBs have been specified, then the buffer will contain the first n-1 records in the first n-1 slots, and the last record of the message in the nth slot.

Example:

Suppose the user passes a 360-byte buffer and tries to read varying size error messages.

1. 2-record error message

1	Record 1	120
121	Record 2	240
241		360

On a normal return, the RESULTS area indicates that the buffer contains the complete message and it is two records.

2. 4-record error message - no OUTDCBs specified.

1	Record 1	120
121	Record 2	240
241	Record 3	360

On an alternate return, the RESULTS area indicates that the buffer holds three records but that the message is incomplete.

3. 4-record error message - OUTDCBs specified.

1	Record 1	120
121	Record 2	240
241	Record 4	360

On a normal return, the RESULTS area indicates that the buffer contains three records but that the message is incomplete. With OUTDCBs specified, a normal return from M$ERRMSG occurs after the complete 4-record message is written through the user DCBs.

CODE = VARIABLE specifies the 1-word area in memory containing the error code identifying the message to be read from the file. It is used to form the key to read the file. To generate the structure for the error code, use the VLP_ERRCODE macro described in this section; or, for monitor service errors, the B$ALT macro discussed in Section 6.

DCB = DCBNAME specifies the DCB on which the error condition occurred. DCB is used to obtain variable information which is substituted into the error message for special strings %FN, %AC, %TA, %UF, %DC and %SN. If an error message contains these special strings and DCB is not specified, then blanks are substituted, or the conditional phrase is not printed. The default is NIL.

FIELD1 = VARIABLE specifies the location of the TEXTC field to be substituted into the error message for the special string %U1. If an error message contains any special string %Ux and the corresponding FIELDx is not supplied, then blanks are substituted, or the conditional phrase is not printed. The default is NIL.

FIELD2 = VARIABLE specifies the location of the TEXTC field to be substituted for %U2. The default is NIL.

FIELD3 = VARIABLE specifies the location of the TEXTC field to be substituted for %U3. The default is NIL.

FILEACCT = VARIABLE specifies the area containing the account for the error message file. The VLP_ACCT macro may be invoked to generate this area. The default is NIL.

FILENAME = VARIABLE specifies the area containing the TEXTC name of a file to be read for the error message. The VLP_NAME macro may be invoked to generate this area. If the named file cannot be opened or if no error message or default message can be found in the file, this condition is treated as if no FILENAME were specified. If FILENAME is not specified or is NIL or if MON is not 0, M$ERRMSG analyzes the FCG of the CODE passed to it. If the first letter of the FCG is X (for a library) or Z (for a language processor) or Q (forms) or Y (installation-specific), the error message file is assumed to be: xxERRMSG, where xx stands for the 2-character FCG (for example, ZF for Fortran). If the first letter of the FCG in CODE is not Q, X, Y or Z or MON='1'B, the monitor service error message file (:ERRMSG) is used. The default is NIL.

NOTE: The system permits foreign language versions of error message files. Each user may optionally have a single character in the JIT which specifies his native language. If this language character is specified, it is used to select the error message file. For example, if the user's language character is 'F' for French, then instead of :ERRMSG, the file :ERRMSGF, which can be defined to contain French text, is accessed.

FILEPASS = VARIABLE specifies the area containing the password for the error message file. The VLP_PASS macro may be invoked to generate this area. The default is NIL.

FLAGLEV = VALUE-DEC(0-4) Error messages may be preceded by some number of asterisks (*). The number is an indication of the seriousness of the message. FLAGLEV gives the number of asterisks, from 0-4, that precede the message. The default is 0.

INCLCODE = {YES|NO} indicates whether or not the error code should be output before the error message. The default is YES.

L6_ALTCODE = {YES|NO} specifies, if YES, that the code specified by the ALTCODE parameter is in unpacked LCP-6 error code format. Default = NO.

L6_CODE = {YES|NO} specifies, if YES, that the code specified by the CODE parameter is in unpacked LCP-6 error code format. Default = NO.

LANG = VALUE-CHAR(1) specifies the character that is to be assumed to be the native language when selecting the error message file.

MY_LANG = {YES|NO} specifies, if YES, that the default native language key is that of the user making the M$ERRMSG request. NO specifies that the character specified via the LANG option is to be used in selecting the error message file. The default is YES.

NREAD = {YES|NO} specifies, if YES, that the buffer contents should be output as passed, without reading the error message file but with substitution. The default is NO.

OUTDCB1 = DCBNAME specifies the first DCB through which to output the error message. The default is NIL.

OUTDCB2 = DCBNAME specifies the second DCB through which to output the error message. The default is NIL.

POINTER = VALUE-DEC(0-LENGTHC(BUF)) specifies, if non-zero, that a pointer ( ) should be output on a line before the error message and in which character position it should be placed. If POINTER is zero, or greater than LENGTHC(BUF), no pointer is output. If the requested message cannot be found, the pointer is output and M$ERRMSG will ALTRETURN after printing the pointer. Note that this option will not work with NREAD = YES. The default is 0.

RESULTS = VARIABLE specifies an area in the user's data where any status information about a call to M$ERRMSG is returned to the caller. The information returned consists of bits indicating unusual conditions encountered in the processing of the CALL, and a field containing the count of records from the message file returned in the user's buffer. The VLR_ERRMSG macro described later in this section can be invoked to generate this area. The default is NIL.

SOURCE = {PASS|TRAP|ALTRET} specifies where M$ERRMSG should look for the error code and DCB to report on. PASS, the default, indicates that the error code will be passed in CODE. TRAP indicates that M$ERRMSG should look in B$EXCFR.ERR for the error code. ALTRET tells M$ERRMSG to look in B$ALT.ERR for the error code and DCB.

The fully qualified name of this field is FPT_ERRMSG.V.SOURCE#. Acceptable EQUs are SOURCE_PASS#, SOURCE_TRAP#, and SOURCE_ALTRET#.

SUBAC = {YES|NO} specifies, if YES, that %AC in the error message is to be considered a candidate for substitution. %AC (account) is obtained from the data control block specified by the DCB parameter. If SUBAC=NO, the %AC is blanked out or the conditional phrase that contains %AC is eliminated. The default is YES.

SUBDC = {YES|NO} specifies, if YES, that %DC in the error message is to be considered a candidate for substitution. %DC (DCB name) is obtained from the DCB parameter. If SUBDC=NO, the %DC is blanked out or the conditional phrase that contains %DC is eliminated. The default is YES.

SUBFILE = {YES|NO} specifies, if YES, to use the default system error message file if the file passed by the user does not contain a message for the code passed in this call. The default is YES.

SUBFN = {YES|NO} specifies, if YES, that %FN in the error message is to be considered a candidate for substitution. %FN (file name) is obtained from the data control block specified by the DCB parameter. If SUBFN=NO, the %FN is blanked out or the conditional phrase that contains %FN is eliminated. The default is YES.

SUBLANG = {YES|NO}. Specifies, if YES, that a message will be in the default language if the user's native language is not available. The default is YES.

SUBMESS = {YES|NO} specifies, if YES, to use a substitute message, (i.e., one with less severity) or the default message, if the error message file contains no message corresponding to the CODE specified. The default is YES. If NO substitute message is available, the text of the error code is used (See INCLCODE).

SUBSN = {YES|NO} specifies, if YES, that %SN in the error message is to be considered a candidate for substitution. %SN (serial number or set name) is obtained from the DCB parameter. If SUBSN=NO, %SN is blanked out or the conditional phrase that contains %SN is eliminated. The default is YES.

TYC = {YES|NO} specifies, if YES, the TYC (type of completion) information from the specified DCB should be output after the error message. TYC is meaningful only if DCB was specified. The default is NO.

VFC = VALUE-CHAR(1) may be specified when writing the error message. This code is used when writing the message through either or both OUTDCBs. The default is zero meaning that the user's default VFC will be used.

Error Message Field and Phrase Substitution

An error message record may contain one or more special strings which indicate that text is to be substituted in their place. This text may be either passed by the user or obtained from a DCB. The following is a list of these special strings.

User passed:
%U1 – specifies text passed in FIELD1 is to be inserted in the message in this place.
%U2 – specifies text passed as FIELD2.
%U3 – specifies text passed as FIELD3.

From DCB:
%FN – the text substituted for this string will be FILENAME[.ACCOUNT] from the passed
      DCB. The account is appended if it is not the current File Management account.
%DC – specifies the text DCB name of the passed DCB.
%AC – specifies that the text account from the passed DCB is to be placed in the
      message.
%SN – specifies the text serial number from the DCB.
%TA – the text to be substituted will take one of these forms:
  FT#xxxxxx – if the passed DCB is open to a freeform tape.
  LT#xxxxxx/filename – if the passed DCB is open to a labeled tape.
  MT#xxxxxx – if the passed DCB is open to a managed tape.
%UF – specifies that an M$UNFID function is to be performed on the passed DCB and the
      resulting text is to be placed in the message.

The M$ERRMSG service makes field and phrase substitutions to the record read from the message file, depending on parameters specified. However, if the expanded record exceeds the size of one buffer slot, the substitution is truncated. A record shown in the following example demonstrates the various substitutions that can occur for one record in the message file.

Example:

End of File %on file %FN %%on DCB %DC%

contains two conditional phrases of which neither, either or both may appear in the final message. The finished message will be one of the following:

M$ERRMSG Parameters		
SUBFN	SUBDC	Message
=NO	=NO	End of File

=YES	=NO	End of File on file FILE.ACCT
=NO	=YES	End of File on DCB M$EI
=YES	=YES	End of File on file FILE.ACCT on DCB M$EI

When calculating the length of a record in the message file, the user must consider the maximum size of the message after substitutions are performed. The record must not produce an actual text message of more than 120 characters.

SUBMESS OPTION

The M$ERRMSG service attempts to find a message in the message file with a key that reflects the exact error code specified in the CODE parameters. If there is no such message and if the SUBMESS=YES parameter is specified, the service searches the file for a suitable replacement message. On a normal return following use of a substitute message, the returned status indicates that an alternate message was used.

The messages suitable for substitution are listed below in the order of the search.

1. A record with the same FCG-MID and ERR# as the requested CODE but a smaller severity (SEV); that is, a less verbose message for the same error.
2. A record with the same FCG, ERR# and SEV but a null MID. This is the default message for this error from this functional code group. A processor which detects the same error in many modules need only have one entry in the message file for the error.
3. A record with the same FCG and ERR# but a null MID and smaller SEV field than the passed code. This is a less verbose FCG default message.
4. A record with the same ERR# and SEV but a null FCG-MID field; that is, the default message for this error condition and level of detail.
5. A record with the same ERR# but a null FCG-MID and a smaller severity; that is, a less verbose default message for this error condition.
6. The text version of the error code (as if INCLCODE has been set).

VLR_ERRMSG

The VLR_ERRMSG macro generates an area which receives the results of the M$ERRMSG service. The area contains these fields:

ALTMSG = VALUE-BIT(1) if set means that M$ERRMSG stored a substitute message in the user buffer.

BADCALL = VALUE-BIT(1) if set specifies that the call contained one or more illegal parameters.

BADPOINT = VALUE-BIT(1) if set specifies that the POINTER option is bad.

BADUSUB = VALUE-BIT(1) if set specifies that a bad user substitution field has been encountered.

BUFERR = VALUE-BIT(1) if set specifies that the buffer is not there or bad.

CODEUSED specifies the code for the message that was in fact output to the caller. CODEUSED is in the normal format for monitor service errors.

ERRCODE1 specifies the error code for OUTDCB1, in the normal format for monitor service errors.

ERRCODE2 specifies the error code for OUTDCB2, in the normal format for monitor service errors.

INERR = VALUE-BIT(1)   if set specifies error on reading the message.

MSGCOMP = VALUE-BIT(1) if set means that the entire message is stored in the user buffer.

NNATIVE = VALUE-BIT(1) if set means that the returned message is not in the user's native language.

NOFILE = VALUE-BIT(1) if set specifies that M$ERRMSG was unable to open the message file.

NOMSG = VALUE-BIT(1) if set means that M$ERRMSG was unable to find any message for the error code.

OUTERR1 = VALUE-BIT(1)   if set specifies an error on OUTDCB1.

OUTERR2 = VALUE-BIT(1)   if set specifies error on OUTDCB2.

RECCNT = VALUE-DEC(0-n)   specifies a count of records read into the user buffer.

TRUNCSUB = VALUE-BIT(1) if set specifies that one or more substitution fields were truncated because expanding them would have overflowed the buffer slot.

VLP_ERRCODE

The VLP_ERRCODE macro generates a standard CP-6 error code. This is the code that will be returned to the user in the altreturn frame. It is also passed to M$ERRMSG to identify a message.

ERR# = VALUE-DEC(0-16383). This field contains the number that identifies a particular error condition. The file B_ERRORS_C contains a list of these ERR#s for the monitor.

FCG = VALUE-BIT(12). This field contains the two special six bit characters that identify the functional code group that is reporting the error. For convenience, it may be specified as text or as a quote string (e.g., FCG=FM).

MID = VALUE-BIT(6). This field contains the special six bit character that identifies which module in the functional code group is reporting the error. For convenience, it may be specified as text or as a quote string (e.g., MID='M').

MON = VALUE-BIT(1). This bit is set if this error is reported by the monitor.

SEV = VALUE-DEC(0-7). This field serves a double purpose. Within the monitor it is used to indicate the seriousness of an error. When passed by the user to M$ERRMSG it indicates the level of detail requested in the error message.

M$HELP - HELP Message Reporting

The CP-6 system provides a comprehensive HELP facility for display of information stored in a central location. HELP is used primarily to provide quick reference information on system or user processors. HELP messages for a each processor are stored in a separate database which may be available throughout the system and capable of being utilized to produce other forms of documentation.

To respond to a HELP command, a processor may call the X$HELP library service, which in turn parses the parameters of the HELP command and calls the M$HELP monitor service. X$HELP isolates the calling processor from the complexities of M$HELP and provides uniform handling of HELP commands by all processors.

FILE OF HELP MESSAGES

The file of HELP messages is a keyed file created by HERMAN.X. The file typically contains the following:

- o Abstract describing the processor and how to request further HELP. There may be breaks between increasingly detailed levels of information within the abstract.
- o Topics describing the major topics of interest concerning a processor. Each topic is assigned a keyword.
- o Subtopics containing information related to the previous topic. A subtopic that is assigned a keyword is a message that can be obtained by a specific request naming the keyword and the topic that it is under. A subtopic without a keyword creates a break between levels of messages related to a topic.
- o Synonyms providing alternate ways to request a topic or subtopic.

For an explanation of how to cıeate a HELP message file, refer to the CP-6 Programmer Guide.

M$HELP

The caller of M$HELP normally specifies the processor name (PROCNAME) and optionally, the account and password. Based on this information M$HELP determines which HELP file to read. The form of the filename is HELP:name:l. Name is the processor name (PROCNAME) passed by the caller and l is the optional native language byte from the JIT. The account for a HELP message file defaults to :SYS.

The caller must also specify a buffer (BUF) to receive the text and an area (RESULTS) to receive status information. The RESULTS area is generated by invoking the VLR_HELP macro.

M$HELP provides two categories of functions: (1) obtaining HELP messages and (2) listing information about HELP. To obtain HELP messages the following parameters apply: KEY1, KEY2, RANGE, ALL, MORE, SUBTOPIC, SUBSTOPIC; LIST=NO, the default, must be taken. The following functions may be requested by using the parameters indicated:

- o Access all messages for a range of topics (RANGE=YES; KEY1, KEY2 are meaningful).
- o Access first level message for a topic specified by keyword (i.e., KEY1; SUBTOPIC is meaningful).
- o Access first level message for a subtopic specified by keywords (i.e., KEY1 identifying the topic, KEY2 identifying the subtopic; SUBTOPIC and SUBSTOPIC are meaningful).
- o For a previously requested topic or subtopic, access the next level message (MORE=YES; RESULTS area which must be unchanged since previous call to M$HELP indicates how much information has already been displayed).
- o For a previously requested topic or subtopic, access the remainder of messages for that topic (ALL=YES; RESULTS area which must be unchanged since previous call to M$HELP indicates how much information has already been displayed).

The SUBTOPIC and SUBSTOPIC parameters permit the caller to request use of a default or substitution message if the requested message cannot be found. If SUBTOPIC=YES and M$HELP cannot find the requested keyword, M$HELP supplies the default message (abstract). If SUBSTOPIC=YES and if KEY2 is specified but KEY2 cannot be found, M$HELP re-accesses the HELP message file, disregarding KEY2, to obtain a substitute message. In these cases, the RESULTS area indicates that the original request was met with a default or substitute message.

After the text is read into the buffer, M$HELP performs output if the user supplied OUTDCB1 and OUTDCB2 parameters. The discussion of the BUF parameter describes how the output function is performed. If no DCBs are specified and the buffer is large enough to contain the text requested, M$HELP returns so that the caller may perform the output operation.

On return to the caller, the RESULTS area reflects the status of the service call. On a normal return, the RESULTS area contains information to allow a subsequent call for M$HELP to respond to the timesharing user's entry of ? (MORE=YES) or ?? (ALL=YES).

The alternate return is taken in the following situations:
if no OUTDCBs are specified and the buffer is too small to contain the text requested; if OUTDCB1 or OUTDCB2 is specified and output cannot be completed through either DCB (M$HELP performs correspondence checking and also verifies that the OUTDCBs are actually DCBs) and the buffer is too small to contain the requested test; if there are database errors, or if bad parameters are passed. Information in the RESULTS area and TCB indicate the reason for the alternate return.

To list information about the available HELP messages, the caller of M$HELP specifies a buffer and RESULTS area. These functions may be requested by using the additional parameters indicated:

- o List processor names that provide HELP (LIST=YES, PROCNAME= NIL; PROCACCT, KEY1, KEY2, and RANGE are meaningful).
- o List TOPICS (i.e., topic keywords) for a specific processor requested by range or by (possibly) truncated keyword (LIST=YES; PROCNAME, PROCACCT, RANGE, KEY1, KEY2 are meaningful).

This list of processor names or keywords is presented to the user in the same way that the message text is presented.

The form of the call for this service is as follows:

CALL M$HELP (FPT_HELP) [ALTRET (label)];

The parameters for this service are as follows:

ALL = {YES|NO} indicates, if YES, that all remaining HELP on the current topic is to be output. This is valid only for HELP text (LIST = NO). The default is NO.

BUF = VARIABLE specifies the buffer used when reading the file of help messages. BUF must have a minimum size of 79 bytes, the maximum size of any record in the file. Because the text requested may consist of multiple records in the file, a 79-byte buffer will not always be large enough to hoid the whole message. The caller may alleviate this condition by specifying a larger buffer.

Records from the file are read into the buffer at 79-byte offsets until the buffer is full or the message is complete. Status is returned to the caller indicating how may records are in the buffer, and whether or not the message is complete. If the buffer is not large enough to hold all of the records of the message, then the contents of the buffer will vary depending on whether the caller supplied OUTDCBs through which to output the messages. Given a buffer with enough space for n records, if no OUTDCBs have been specified, the buffer will contain the first n records of the message. If OUTDCBs have been specified, then the buffer will contain the first n-1 records in the first n-1 slots, and the last record of the message in the nth slot.

EXAMPLE:

Suppose the user passes a 237-byte buffer and tries to read varying size text.

1. 2–record error text.

```
        1          Record 1          79
      121          Record 2         158
      159                           237
```

   On a normal return, the RESULTS area indicates that the buffer contains the complete message and it is two records.

2. 4–record text – no OUTDCBs specified.

```
        1          Record 1          79
       80          Record 2         158
      159          Record 3         237
```

   On an alternate return, the RESULTS area indicates that the buffer holds three records but that the message is incomplete.

3. 4–record error message – OUTDCBs specified.

```
        1          Record 1          79
       80          Record 2         158
      159          Record 4         237
```

   On a normal return, the RESULTS area indicates that the buffer contains three records but that the message is incomplete. With OUTDCBs specified, a normal return from M$HELP occurs after the complete 4–record message is written through the user DCBs.

KEY1 = VARIABLE locates a TEXTC area containing the first keyword. When used in conjunction with RANGE, RANGE = YES indicates the first in a range of requested HELP items. When RANGE = YES and KEY1 is blank or omitted, the first existing HELP item is obtained. RANGE = NO indicates that KEY1 frames the main topic word that, with any existing sub–topic word (KEY2), defines a single HELP item if LIST = NO. When LIST = YES, KEY1 frames the starting characters of a wildcard.

KEY2 = VARIABLE locates a TEXTC area containing the second keyword. When used in conjunction with RANGE, RANGE = YES indicates the last in a range of requested HELP items. When RANGE = YES and KEY2 is blank or omitted, the last existing HELP item is obtained. RANGE = NO indicates the entry is the subtopic word that, with any existing topic word (KEY1), defines a single HELP item.

LANG = VALUE–CHAR(1) specifies user native language byte from the JIT. LANG is used to select a HELP file in the user's own language. The default is blank, meaning no special language.

LIST = {YES|NO} indicates, if YES, that the user wishes to see a list of the available topics for the HELPing processor. If the PROCNAME is blank or omitted, M$HELP will list the processors in the account indicated by PROCACCT, for which HELP exists. If NO, the user wishes to see the actual HELP text. The default is NO.

MORE = {YES|NO} indicates, if YES, that the next message(s) of HELP text for the current topic is to be output. This is valid only for HELP text. The default is NO.

OUTDCB1 = DCBNAME is the first DCB through which to output the HELP message.

OUTDCB2 = DCBNAME is the second DCB through which to output the HELP messages.

PROCACCT = VARIABLE locates a VLP_ACCT area containing the account of the processor for which HELP is requested.

PROCNAME = VARIABLE locates a VLP_NAME area containing the name of the processor for which HELP is requested. The processor name must not exceed 24 characters in length.

PROCPASS = VARIABLE locates a VLP_PASS area containing the password for the processor for which HELP is requested.

PROCPSN = VARIABLE locates a six-character area containing the packset of the processor for which HELP is requested.

RANGE = {YES|NO} is used in combination with KEY1 and KEY2 to set limits on the data output. If RANGE is YES, KEY1 and KEY2 are to be used as inclusive limits on a range of items to display. If RANGE = NO and LIST = YES, KEY1 may be provided as a wildcard; in this case, any item in the display list that starts with the keyword is printed. If RANGE = NO and LIST = NO, the keywords are used to select a single message by topic and subtopic. The default is NO.

RESULTS = VARIABLE locates a VLR_HELP area passed by the user where status information about the call to M$HELP is returned. Because this also contains scratch area for M$HELP, it is required for callers who wish to use the MORE or ALL functions or do their own output.

SUBSTOPIC = {YES|NO} indicates whether or not to use a substitute message (i.e., the one that would have resulted if KEY2 were not passed) if M$HELP is unable to find the specified one. SUBSTOPIC = YES is valid only when LIST = NO and RANGE = NO. The default is YES.

SUBTOPIC = {YES|NO} indicates whether or not to use the default message if the requested topic does not exist. The default is NO.

XLATEKEY = {YES|NO} indicates, if YES, that the user wishes M$HELP to translate the passed KEYWORDs into upper case and try again if a message isn't found the first time. The default is YES.

VLR_HELP

The VLR_HELP macro generates an area which receives the results of the M$HELP. The area contains these fields:

ALTMSG = VALUE-BIT(1) if set means that M$HELP stored a substitute message in the user buffer.

BADCALL = VALUE-BIT(1) if set specifies that the call contained one or more illegal parameters.

BUFERR = VALUE-BIT(1) if set specifies that the buffer is not there or bad.

ERRCODE1 specifies the error code for OUTDCB1, in the normal format for monitor service errors.

ERRCODE2 specifies the error code for OUTDCB2, in the normal format for monitor service errors.

INERR = VALUE-BIT(1) if set specifies error on reading the message.

MSGCOMP = VALUE-BIT(1) if set means that the entire message is stored in the user buffer.

NNATIVE = VALUE-BIT(1) if set means that the returned message is not in the user's native language.

NOFILE = VALUE-BIT(1) if set specifies that M$HELP was unable to open the message file.

NOMSG = VALUE-BIT(1) if set means that M$HELP was unable to find any message for the error code.

OUTERR1 = VALUE-BIT(1) if set specifies an error on OUTDCB1.

OUTERR2 = VALUE-BIT(1) if set specifies error on OUTDCB2.

RECCNT = VALUE-DEC(0-n) specifies a count of records read into the user buffer.

M$TIME - Time and Date

The M$TIME service returns date and time information, obtained from one of several possible sources, in several optional formats. The source of the date and time may be an input from the calling program or the current system elapsed time clock. The output format options include three character string formats and two binary formats.

One of the possible source and output formats is the CP-6 Universal Time Stamp (UTS), which is a 36 bit binary value indicating the elapsed time since a base date and time, in units of 1/100 second. UTS data is used throughout the CP-6 system as the standard notation for time.

A second type of UTS format, denoted by the FUTS option is provided to facilitate exchange of times between host and FEP systems. The FEP UTS is a 32 bit value expressing elapsed time in units of .133 seconds. In CP-6 host systems the FEP UTS is contained in one word with bits 0-3 set to zero.

The M$TIME service takes the alternate return if the source date is prior to January 1, 1978. (Dates prior to that time can occur when processing data from other systems and cannot be converted to UTS). The form of the call for this service is as follows:

CALL M$TIME (FPT_TIME) [ALTRET (label)];

The parameters for M$TIME are:

DATE = VARIABLE specifies a character string buffer to contain the month, day and year. The size must be that shown in the table for the selected DEST format and must correctly describe the input character string when used with the SOURCE=LOCAL or SOURCE = ANS option. The default is NIL.

DAY = VARIABLE specifies a character string buffer for the day of the week for LOCAL or EXT format or the Julian date for ANS format outputs. The size must be that shown in the table for the selected format. The default is NIL.

DEST = OPTION Specifies the format of the results of the M$TIME service. The table below illustrates the formats and gives the size, in characters, of the buffer. Results are not returned if a buffer is not specified. The default is EXT. The options are listed below.

EXT, ANS, LOCAL are described in the following table.

OPTION	BUFFER	SIZE	FORMAT	EXAMPLE
EXT	DATE	10	'MON DD ''YY'	DEC 31 '77
	TIME	11	'HH:MM:SS.SS'	23:59:59.99
	DAY	3	'DAY'	SUN
LOCAL	DATE	8	'MM/DD/YY'	12/31/77
	TIME	11	'HH:MM:SS.SS'	23:59:59.99
	DAY	3	'DAY'	SUN
ANS	DATE	6	'YYMMDD'	771231
	TIME	8	'HHMMSSSS'	23595999
	DAY	5	'YYDDD'	80366

UTS specifies that a UTS is to be computed from the specified input and returned in the TSTAMP buffer.

FUTS specifies that an FEP UTS is to be computed from the specified input and returned in the TSTAMP buffer.

TUN specifies that the timer since midnight, in milliseconds, is to be computed from the specified input and returned in the TSTAMP buffer.

SOURCE = OPTION Indicates the source of the date and time. The options are as follows:

CLOCK	specifies that the current date and time, obtained from the system elapsed time clock, are to be returned.
UTS	specifies that the date and time are to be calculated from an input UTS value found in the TSTAMP buffer.
FUTS	specifies that the date and time are to be calculated from an input FEP UTS value found in the TSTAMP buffer.
LOCAL	specifies that the date and time are to be calculated from input character string data found in the DATE and TIME buffers. Input data must be in the LOCAL format described in the table above except that TIME is optional and defaults to zero. (Note that leading zeros may be omitted from any field and the seconds or hundredths of seconds fields may be omitted, in which case zero is assumed). The exact length of the string must be reflected in TIME_.BOUND. Output data will be stored into the DATE and TIME buffers if the EXT, ANS, or LOCAL options are specified for DEST.
ANS	specifies that the date and time are to be calculated from input character string data found in the DATE and TIME buffers. Input data must be in the ANS format described in the table above except that TIME is optional and defaults to zero. Output data will be stored into the DATE and TIME buffers if the EXT, ANS, or LOCAL options are specified for DEST.
TUN	specifies that the input for time is a binary value of milliseconds since midnight to be found in the TSTAMP buffer. Date is to be the current date.

The default is CLOCK.

TIME = VARIABLE specifies a character string buffer to contain the time of day. The size must be that shown in the table for the selected DEST format and must correctly describe the input character string when used with the SOURCE=LOCAL or SOURCE=ANS option. The default is NIL.

TSTAMP = VARIABLE specifies a 1-word buffer to contain either a host or FEP UTS, or milliseconds since midnight. The default is NIL.

M$XEQTIME - Execution Time

The Execution Time service returns, in microseconds, the amount of execution and service time expended during the current job, in microseconds. The times are either processor or user times, depending on the mode of the caller.

The form of the call for this service is as follows:

CALL M$XEQTIME (FPT_XEQTIME) [ALTRET (label)];

The parameter for this service is:

BUF = VARIABLE specifies a two-doubleword buffer. The first double word contains execution time; the second double word contains service time. Both values are in microseconds.

M$STATE - User State Information

The User State Information service returns information describing the current execution state, the amount of time expended and the amount of time remaining, for a specified user.

The form of the call for this service is as follows:

CALL M$STATE (FPT_STATE) [ALTRET(label)];

The parameters for this service are:

BUF = VARIABLE specifies a four word block of memory to contain the results of the M$STATE service. The contents of BUF will be:
Word 0 will contain the execution state number of the specified user (UBIN WORD).
Word 1 will contain a four character mnemonic state name (CHAR (4)).
Word 2 will contain the execution time remaining in the job or session in 1/100 second units (UBIN WORD).
Word 3 will contain the execution time used during the job or session in 1/100 second units (UBIN WORD).

SYSID = VALUE-DEC(0-n) is the SYSID of a user whose current execution state and time parameters are to be returned.

M$JOBSTATS - Job Status

The M$JOBSTATS service has several functions, all involved with job status operations. The function to be performed is selected by ORDER in the FPT.

Some initial concepts are required in order to use this service properly. For the purposes of this service, the term 'job' is defined as follows: a 'job' is either a batch job (together with its input symbiont file, and output symbiont files if any) or the symbiont output produced by a non-batch user. In other words, if a non-batch user with sysid X produces symbiont output, then this service considers the 'job' of sysid X to be those output symbiont files, and they are precisely what can be manipulated and/or checked by the use of this service. On the other hand, a batch 'job' is considered by this service to be not only the symbiont output produced by the batch user, but also the user himself and his input control command (symbiont) file. Thus, this service can manipulate and/or check the status of the batch user, as well as his symbiont output; for example the CANCEL function of M$JOBSTATS, if directed at a batch 'job' can abort the batch user, or delete his symbiont output, or both.

A batch 'job' is defined for the purposes of this service to be either 'current' or 'obsolete'. A batch 'job' becomes current upon receipt by PRESCAN. If it fails PRESCAN, it becomes obsolete immediately. After passing PRESCAN the job enters the waiting-to-run state and remains current thereafter until:

1- It is cancelled before being run.

or

2- It is run to completion (note that 'completion' can include being aborted or Xed) and all of its symbiont output (if any) has been disposed of, either through being printed, or through being deleted (via CANCEL).

The only access to the status of 'current' jobs is through this service. Once a job becomes 'obsolete', its final status is written into the :JOBSTATS file in :SYS, and its status is no longer available through this service. Note that if a batch job fails PRESCAN, it becomes immediately obsolete and its record is written into the job statistics file, :JOBSTATS, with an indication of exactly why it failed PRESCAN. The structure of the obsolete file is defined elsewhere. Note also that when a batch job terminates its run and has created symbiont output, information pertinent to the run is available in the obsolete file even though the status of its still-current symbiont output is accessible via this service.

Jobs can be accessed (both through this service and within the obsolete file) either by SYSID or by JOBNAME and ACCOUNT. A jobname is only unique within a specific account. However, in the obsolete file within a given account, only one record of a given jobname is kept (that of the most recently submitted job with that name); on the other hand, the job queues can clearly contain more than one job of a given jobname and account at a specific time. It is important to note that when a job is accessed by jobname, only one job of that name is affected, namely the job which is highest in the queues; in other words, for those M$JOBSTATS functions which access a single job by name, only the first job of the given name and account which is found, is affected. Since the order of submittal of two such jobs is not necessarily the order in which they will appear in the queues, it is important to be aware of this detail.

The form of the call for this service is:

CALL M$JOBSTATS (FPT_JOBSTATS) [ALTRET (label)];

The parameters for this service are as follows:

ACCT=VARIABLE Locates a VLP_ACCT area. If a job is being CHECKed or CANCELled by jobname, then ACCT specifies the account in which the named job is to be found. The default, if not specified, is the account of the service call issuer.

BUF=VARIABLE Locates the buffer to be mapped when a REMAP function is executed.

CRITERIA=VARIABLE Locates a VLP_CRITERIA area when a SRCH or SPILL function is being executed. This area defines the criteria which control exactly which jobs are returned by the search.

DISP=OPTION Specifies the final disposition of the selected file for an ORDER=SPILLED function. The available options are:

REQUEUE - The file is to be marked as eligible for output

REMOVE - The file is to be deleted

SPILLED - The file is to be placed on the SPILLED queue, marked as having been spilled, and deleted except for its FIT. Note that in this case, the file can still be status-checked by M$JOBSTATS CHECK, and retrieved by M$JOBSTATS SRCH but not by M$JOBSTATS SPILL.

The default is REQUEUE.

JOBNAME=VARIABLE Locates a VLP_JOBNAME area. If JOBNAME is specified on a CANCEL or CHECK function, then the job is being identified by jobname, not by SYSID.

NUM = VALUE-DEC(0-?) Specifies the output symbiont file NUM for the CHECKF, CANCELF, UNHOLDF, SPILLED, and FILL functions. The default is 0.

ORDER=OPTION Specifies which of the various M$JOBSTATS functions is to be performed. The available options are:

CHECK - This function checks the status of a job. The job is identified by jobname if JOBNAME is specified, and otherwise by sysid, via the SYSID parameter. If the job is not current, this is not considered an error. The RESULTS area receives the current status of the job (see VLR_JOBSTATS_CHECK for the definition of the returned status). Note that if the checked job is not from the service call issuer's account, then if the issuer does not have the PR_DISPJOB privilege, a 'not found' status is returned.

CHECKF - This function checks the status of a particular output symbiont file. The job which produced the file is identified as for CHECK, and the particular file is specified by the NUM parameter (the files produced by a job have NUMs of 1,2, etc.). The RESULTS area receives the current status of the job, just as for CHECK, but also receives the status of the specified file. If the specified job is not found, the RESULTS area receives a status of 'not found' for both job and file. If the specified job does exist but the specified file was not found, then the RESULTS area receives the status of the job, and a 'not found' status for the file.

CANCEL - This function cancels a job, the job being identified as for CHECK. The OUTPUT parameter specifies whether or not the job's output is to be cancelled. Note that in the case of a non-batch 'job', this function with OUTPUT=NO causes no action since the 'job' consists only of output. For a batch job, the job is deleted if it is waiting to run, and aborted if it is running. If the job has already run and OUTPUT is not selected, an error is returned indicating that no action has been taken. Note that if OUTPUT is selected and the batch job is running but has yet to produce any output, then the job is aborted and its output discarded. Finally, note that if the account of the job does not match the account of the issuer of the service call, an error is returned and no action taken.

CANCELF - This function cancels (deletes) a particular output symbiont file, identified as for CHECKF. If the specified file is not found, a 'not found' error is returned.

UNHOLD - This function removes a job's symbiont output from HOLD status (via LDEV, a user can specify that a certain number of copies of an output symbiont file are to be printed, after which the file is put in HOLD state, to be printed only via operator intervention, or by use of this service). The job is identified as for CHECK. If the job has no current output, an error is returned. If the job's account does not match that of the service call issuer, an error is returned and no action taken. Otherwise, the job's output is removed from HOLD and becomes a candidate for printing.

UNHOLDF - This function removes from hold a particular output symbiont file, identified as for CHECKF. If the specified file is not found, a 'not found' error is returned.

SRCH - This function searches the queue of current output (OUTPUT=YES) or waiting batch jobs (OUTPUT=NO), according to the criteria specified by CRITERIA. Each job in the selected queue which meets the specified criteria causes a frame of information concerning the job to be deposited in the RESULTS area. An error is returned if no such jobs are found. Note that if the service call issuer does not possess the special authorization (PR_DISPJOB), jobs not in his account are not returned in the results area.

SPILL - This function may only be issued by a user having PR_FMEFT privilege. Its purpose is to mark a portion of the output symbiont queues for spill. The files to be spilled are selected in the same way as for the SRCH function, that is, by means of VLP_CRITERIA. For a given set of criteria, the same set of files will be selected by SRCH and SPILL, except that SPILL will not receive files which are currently being output, or which are in concurrent mode, or which have been marked as SPILLED (see ORDER=SPILLED) function; the SPILL issuer need not have the PR_DISPJOB privilege to receive files not in his account. The SPILL function returns one frame into the RESULTS area for each file found, and simultaneously marks each such file for SPILL, so that it will not be output. Note that this marking of the file is done in such a way that it is not remembered across system recoveries. Note also that the SPILL function will return files which are already marked for SPILL.

SPILLED - This function may only be issued by a user having PR_FMEFT privilege. Its purpose is to notify the system of the final disposition of a file which has been marked for SPILL (see the SPILL function). The DISP parameter tells the system the desired disposition. The file is identified as for CHECKF, and an error is returned if that file is not already marked for SPILL.

FILL - This function may only be issued by a user having PR_FMEFT privilege. Its purpose is to notify the system that an output symbiont file which was previously spilled, has now been restored. The file is identified as for ORDER=SPILLED, and an error occurs if it is not found or not marked as having been spilled. This function causes the system to re-queue the file for printing.

REMAP - This function may only be issued by the SLUG and OUTSYM system ghosts. Its purpose is to map or remap a segment of the memory-resident job status information into the monitor WSQ page table used by the other functions of this service. The segment to be mapped is identified by STATSEG, and BUF frames the segment. If the page table contains insufficient room to perform the mapping, an error is returned. The default is REMAP.

OUTPUT= {YES|NO}. For the CANCEL function, this specifies whether (YES) or not (NO) the job's output is to be deleted. For the SRCH function, it specifies which queue is to be searched, the output symbiont queues (OUTPUT=YES), or the batch queue (OUTPUT=NO). The default is NO.

RESLIST=VARIABLE Locates a VLR_JOBSTATS_ISRCH_RES area. If RESLIST is specified on a batch queue (OUTPUT=NO) SRCH function, then the virtual resource requirement lists for the returned jobs are deposited in this area.

RESULTS=VARIABLE Locates a VLR_JOBSTATS_OSRCH or VLR_JOBSTATS_ISRCH area when a SRCH function is being executed. This area receives the results of the search. RESULTS locates a VLR_JOBSTATS_CHECK area when a CHECK/CHECKF function is being performed; this area receives the result of the CHECK. RESULTS locates a VLR_JOBSTATS_OSRCH area when a SPILL function is being performed; this area receives the results of the SPILL.

SNLIST=VARIABLE Locates a VLR_JOBSTATS_ISRCH_SN area. If SNLIST is specified on a batch queue (OUTPUT=NO) SRCH function, then the packset lists for the returned jobs are deposited in this area.

STATSEG=OPTION STATSEG is used with the REMAP function to identify the segment to be mapped. The options are defined in FOS_SUBS_C.

SYSID=VALUE-DEC(0-n). SYSID identifies the job for the UNHOLD, CANCEL, and CHECK functions, in the absence of the JOBNAME.

VLP_CRITERIA

The macro generates an area for use with the SPILL or SRCH function of the M$JOBSTATS service. It contains the criteria according to which the selected queue is to be searched. All of the specified criteria are logically ANDed together, that is, only those jobs meeting all of the specified criteria are returned by the service. Each criterion is specified as a VALUE of appropriate type, with the default value indicating that this criterion is not to be applied to the search.

ACCT=VALUE-CHAR(8). Specifies the account of the desired jobs.

DEVTYP = VALUE-CHAR(2). For an output queue search, specifies the two-character device type (e.g., LP) destination of the desired files.

FORM = VALUE-CHAR(6). For an output queue search, specifies the device FORM required by the desired files.

HOLD = {YES|NO}. For an output queue search, specifies that only files in HOLD are to be returned. See also PRIO1.

MFW = {YES|NO}. For an output queue search, specifies that only files in the multi-file wait queue (i.e., waiting for a concurrent output mode file to be finished) are to be returned. See also PRIO1.

NORUN= {YES|NO}. For a batch queue search, NORUN=YES requests that the only jobs to be returned are those which require either partition change, or a (currently unmounted) shared packset, in order to run. Note that this criterion will NOT detect jobs requiring non-available physical resources (e.g., resource line printers).

NOTACCT = {YES|NO}. Applies only if an account is supplied. If NOTACCT=YES, causes entities NOT from the specified account to be selected. The default is NO.

NOTFORM = {YES|NO}. Applies only if a formname is supplied. If NOTFORM=YES, causes entities NOT for the specified formname to be selected. The default is NO.

NOTTNAME = {YES|NO}. Applies only if a TNAME is supplied. If NOTTNAME=YES, causes entities NOT from the specified TNAME to be selected. The default is NO.

NOTWSN = {YES|NO}. Applies only if a workstation is supplied. If NOTWSN=YES, causes entities NOT from the specified workstation to be selected. The default is NO.

NSFORM = {YES|NO}. For an output queue search, NSFORM=YES requests that only files with non-standard FORM be returned.

PRIO=VALUE-DEC(0-15). Specifies the low end of the range of priorities for jobs which are to be returned (applies to batch queue and output file searches). See PRIO1.

PRIO1=VALUE-DEC(1-15). Specifies the high end of the range of priorities for jobs which are to be returned. Note that the values for PRIO and PRIO1 must lie between 0 and 15, which are the limits for batch and output priorities. One consequence of this is that one cannot, for example, use priority range to select batch jobs waiting for 'runafter' time, or output files in HOLD, for the 'priorities' of these entities lie outside the range 0-15. To select such jobs, other criteria must be specified (see NORUN, MFW, HOLD, RUNAFTER, WPS, and SPILLED). Note that the criteria NORUN, MFW, HOLD, RUNAFTER, WPS, RUNNING, SPILLED, PRIO, and PRIO1 select which queues are to be searched, and thus are mutually exclusive. The default, achieved by specifying none of these, is to search all queues, subject to the other criteria.

RUNAFTER= {YES|NO}. For a batch queue search, specifies that only jobs in the 'runafter' queue are to be returned. See also PRIO1.

RUNNING= {YES|NO}. For a batch queue search, RUNNING=YES requests that only jobs which are currently running be returned. See PRIO1.

SPILLED = {YES|NO}. For an output queue search, specifies that only files which have been spilled are to be returned. See also PRIO1.

SYSID=VALUE-DEC(0-n). Specifies the low end of the desired sysid range.

SYSID1=VALUE-DEC(0-n). Specifies the high end of the desired sysid range, that is, jobs with sysid X such that SYSID <= X <= SYSID1 are to be returned.

TNAME = VALUE-CHAR(8). For an output queue search, specifies the TNAME (terminal name) of the desired output files.

WPS = {YES|NO}. For a batch queue search, WPS = YES requests that only jobs waiting for a shared packset be returned. See also PRIO1.

WSN = VALUE-CHAR(8). For a batch queue search, specifies the workstation of origin or the WSN of the desired jobs. Jobs whose WOO or WSN match the given value are returned. For an output queue search, WSN specifies the workstation of destination or the WOD of the desired jobs. Jobs whose WOD match the given value are returned. Note that the supplied WSN may be wild-carded, that is, if WSN contains a '?' character, then it selects all workstations matching the given string up to the '?'.

VLP_JOBNAME

This macro generates an area for use with the CHECK, CHECKF, UNHOLD, UNHOLDF, CANCEL, and CANCELF functions of the M$JOBSTATS service, containing a jobname. In identifying a batch job, the jobname is always qualified by an account, and the two together uniquely identify a batch job, provided that the JOB card of that job specified a jobname.

JOBNAME=VALUE-CHAR(31). Specifies the name of the desired job.

VLR_JOBSTATS_CHECK

This macro generates an area which receives the results of the CHECK and CHECKF functions of the M$JOBSTATS service.

AHEAD = VALUE-DEC(0-n). For the CHECK function, AHEAD is set to the number of jobs ahead of the checked job if the status is JS_WAIT, and is set to the number of output files ahead of the checked job's highest-priority output file if the status is JS_PRT or JS_PWAIT. For CHECKF, AHEAD is set to the number of output files ahead of the checked output file.

CODE = VALUE-DEC(0-n). Set to the status code for the job. These codes are listed in CP_6_SUBS as JS_XXXX#, and have the following meanings:

- o JS_RUN - Running
- o JS_WAIT - Waiting to run
- o JS_PRT - The job has finished running and at least one output file is printing
- o JS_PWAIT - The job has finished running and has at least one output file, but none currently printing
- o JS_SPILLED - The job has finished running and has one or more output files, all of which have been spilled.
- o JS_AP - The job is awaiting PRESCAN
- o JS_NOTF - The job is not current and cannot be found; it may be listed in the obsolete file.

DEVAVAIL = VALUE-BIT(1). For CHECK, if the status is JS_PWAIT, DEVAVAIL is set if there are currently any output symbiont devices capable of writing any of the job's output files, and is reset if not. For CHECKF, DEVAVAIL is set if there are currently any output symbiont devices capable of writing the checked file, and is reset if not.

DEVNAME = VALUE-CHAR(4) For the CHECK function, set to the name of an output symbiont device on which one of the job's output files is printing, provided that the job's status is 'printing'. (If at least one of the currently-printing files is being output on a device of type 'LP', then one of these is chosen). For the CHECKF function, set to the name of the device on which the checked file is printing, if the file's status is 'printing'.

FCODE = VALUE-DEC(0-n) Set to the status of the specified output file by the CHECKF function (not meaningful during a CHECK function). The status codes are in CP_6_SUBS as FS_XXXX.

FPRIO = VALUE-DEC(0-n) Set to the priority of the specified output file by the CHECKF function (not meaningful during a CHECK function). The priorities are defined in CP_6_SUBS as FP_XXXX.

IPRIO = VALUE-DEC(0-n) is the priority queue which the job is either currently in, or is waiting to be placed in.

NOF = VALUE-DEC(0-n) Set to the total number of output symbiont files thus far produced by the checked job. Note that this count is only guaranteed to be correct for batch jobs, since other (e.g., on-line or ghost) users do not have their output file counts accumulated throughout their entire session.

NOFDONE = VALUE-DEC(0-n) Set to the number of output symbiont files which were produced by the checked job and have been completely dealt with (e.g., finished output or cancelled).

NOFPRINTING = VALUE-DEC(0-n) Set to the number of output symbiont files produced by the checked job which are currently being output.

NOFSPILLED = VALUE-DEC(0-n) Set to the number of output symbiont files produced by the checked job which have been spilled.

PRIO = VALUE-DEC(0-n). Set to the job's priority if CODE is JS_WAIT (waiting to run). See JP_XXXX in CP_6_SUBS.

SYSID = VALUE-DEC(0-n) Set to the checked job's SYSID unless the status is 'not found'. This is useful if the job was identified by jobname, and the user wishes to know the sysid.

TNAME = VALUE-CHAR(8). For the CHECK function, TNAME is the name of the TERMINAL to which the device given in DEVNAME belongs.

VLR_JOBSTATS_OSRCH

This macro defines an area to receive the results of the SPILL and SRCH (OUTPUT=YES) functions of the M$JOBSTATS service. NFIL is the number of file frames to be allocated in the area. Note that the service will return an error if the number of files selected by the criteria exceeds the number of frames allocated.

F = ARRAY(0:NFIL) CHAR(104) is the array of file frames to be filled as a result of the search. Note that an element of F is a structure defined by VLA_JOBSTATS_OSRCH. NFIL is the number of file frames to be allocated in the area.

FOUND = VALUE-DEC(0-n). Set to the number of files found.

VLA_JOBSTATS_OSRCH

This macro defines the structure of an element of the array VLR_JOBSTATS_OSRCH.F. The based form of this structure must be used to reference any of the individual items of an array element. Each array element is an even number of words in length.

ACCT = VALUE-CHAR(8) is the account of the user who created the file.

AHEAD = VALUE-DEC(0-n) is the number of output files ahead of this one in the queue for this terminal.

ATTR = VALUE-BIT(18) is the mask of required device attributes.

BOOTID = VALUE-CHAR(3) is the first 3 characters of the file name in :SYMO.

COPIES = VALUE-DEC (0-n) specifies the number of copies to be printed.

CREATED=VALUE-DEC(0-n) is the system time stamp corresponding to the file's creation time.

DEVNAME = VALUE-CHAR(2) if the file is not currently being output, is blanks. Otherwise it is the last half of the name of the device on which the file is being output (the first half being DEVTYP).

DEVTYP = VALUE-CHAR(2)   specifies the destination device type (e.g., 'LP').

FLG contains the following flags:

FLG.DEVAVAIL = VALUE-BIT(1) set if there are currently any output symbiont devices capable of writing this file, and reset if not.

FLG.GLC = VALUE-BIT(1) set if OUTSYM has received the last chunk of this file (reset for a concurrent mode file whose last chunk has not yet been received.)

FLG.SAP = VALUE-BIT(1) set if file is to be placed in HOLD after SAPCOPIES copies have been printed.

FLG.SPILL = VALUE-BIT(1) set if the file has been marked for spill by an M$JOBSTATS SPILL function.

FORM = VALUE-CHAR(6) is the name of the required device form for this file.

GRANS = VALUE-DEC(0-n)   is the size of the file in granules.

HICNUM = VALUE-DEC(0-n) is the CCC part of the name of the last-received concurrent mode chunk for this file.

JNAME = VALUE-CHAR(31) is the jobname of the batch job which created this file (if the creator was not a batch job, or if it was a batch job with no jobname, this field is blanks).

MODE = VALUE-DEC(0-n) is the mode of the user who created this file. See MODE_XXX# in CP_6_SUBS.

NUM = VALUE-DEC(0-n) corresponds to the second part of the file's name in account :SYMO (output symbiont file names are of the form BBBSSSSSSNNNNNNCCC where BBB is the bootid (a value for this boot), SSSSSS is the sysid, NNNNNN is the NUM, and CCC is the concurrent mode chunk number).

PRIO = VALUE-DEC(0-n)   is the priority of the file. See FP_XXXX in CP_6_SUBS.

SAPCOPIES = VALUE-DEC(0-n) specifies the number of copies to be printed before putting the file into HOLD, if FLG.SAP is set.

SYSID = VALUE-DEC(0-n)   is the sysid of the user who produced the file.

TNAME = VALUE-CHAR(8) is the destination TNAME (terminal name).

WOD = VALUE-CHAR(8)   is the job's WSN of destination.

VLR_JOBSTATS_ISRCH

This macro defines an area to be used with the batch queue (OUTPUT=NO) form of the SRCH function of the M$JOBSTATS service. It defines a number (NJOB) of of frames for jobs, and receives one frame for each job which is returned.

FOUND = VALUE-DEC(0-n)   is the number of jobs returned.

J = ARRAY(0:NJOB) CHAR(256) is the array of job frames returned, each entry corresponding to one job. Note that the structure of an entry is defined by VLA_JOBSTATS_ISRCH. NJOB specifies the number of frames for jobs.

VLR_JOBSTATS_ISRCH_RES

This macro defines an area to be used with the batch queue (OUTPUT=NO) form of the SRCH function of the M$JOBSTATS service. It defines a number (NRES) of frames for resources, and receives, for returned jobs which require resources, information about those requirements.

R = ARRAY(0:NRES) is the array of resource frames returned. Each job returned which requires resources causes the service to store into R one frame for each resource required by the job. The job frame in VLR_JOBSTATS_ISRCH.J will contain the number of resources required, and the index into the VLR_JOBSTATS_ISRCH_RES.R array of the first resource frame. Each element of the array is an even number of words in length.

R.ATTR = VALUE-BIT(18) is the mask of required attributes for this resource (e.g., if the resource is 'LP', then these are the required device attributes).

R.TYP = VALUE-CHAR(2)   is the resource type name (e.g., 'MT').

R.VNUM = VALUE-CHAR(2) is the resource number with respect to this type of resource (e.g., if the user requests a virtual tape drive 'MT03', this field for that resource will be '03').

VLR_JOBSTATS_ISRCH_SN

This macro defines an area to be used with the batch queue (OUTPUT=NO) form of the SRCH function of the M$JOBSTATS service. It defines a number (NSN) of frames for packsets, and receives, for returned jobs which require packsets, information about those sets.

S = ARRAY(0:NSN) is the array of packset frames returned. Each job returned which requires packsets causes the service to store into S one frame for each packset required by the job. The job frame in VLR_JOBSTATS_ISRCH.J will contain the number of packsets found, and the index into the VLR_JOBSTATS_ISRCH_SN.S array of the first packset frame. Each element of the array is an even number of words in length.

S.FLG contains the following flags describing the use of the packset.

S.FLG.EXCL = VALUE-BIT(1)   set if the packset is to be exclusive.

S.FLG.PUB = VALUE-BIT(1)   set if the packset is to be public or shared.

S.NAME = VALUE-CHAR(6)   is the packset name.

VLA_JOBSTATS_ISRCH

This macro defines the structure of an element of the array VLR_JOBSTATS_ISRCH.J. The based version of this structure must be used to reference any of the individual fields in an array element. Each array element is an even number of words in length.

ACCT = VALUE-CHAR(8)   is the job's account.

AHEAD = VALUE-DEC(0-n) is the number of jobs ahead of this one in the queue. It does not count running jobs.

FLG is a mask of the following flags:

FLG.03 = VALUE-BIT(1)   set if FOLLOW specified.

FLG.ACC = VALUE-BIT(1)   set if ACCOUNT specified.

FLG.NPI = VALUE-BIT(1)   set if NPI (no priority increment) specified.

FLG.O1 = VALUE-BIT(1)   set if ORDER specified.

FLG.RA = VALUE-BIT(1)   set if DEFER specified.

FLG.RR = VALUE-BIT(1)   set if RERUN was specified.

FLG.RR2 = VALUE-BIT(1)   set if currently being rerun.

IPRIO = VALUE-DEC(0-n) is the priority queue which the job is either currently in, or is waiting to be placed in.

JDID = ARRAY is an array describing the FOLLOW dependencies (if FLG.O3 is set), with one entry for each follow condition specified by this job.

JDID.FOLLOW_TYPE = VALUE-DEC(0-n) is the type of FOLLOW relation specified. The codes are:

0	STEPCC condition
1	OK
2	ERROR
3	ABORT
4	follow all jobs in this job's account
other	undefined

JDID.RELATION = VALUE-DEC(0-n) is the relation with the followed job's final STEPCC (if JDID.FOLLOW_TYPE(k) is 0); for example, '>=' indicates that the followed job's final STEPCC must be greater than or equal to n for this job to run, n being the contents of JDSTEPCC(k) where this entry is JDID(k). The codes are:

0	<
1	<=
2	=
3	
4	>=
5	>
other	undefined

JDID.SATISFIED = VALUE-BIT(1) indicates whether (set) or not (reset) this dependency has been satisfied.

JDID.SYSID = VALUE-DEC(0-n) is the sysid of the job to be followed. If JDID.FOLLOW_TYPE(k) = 4, JDID.SYSID(k) = 0.

JDSTEPCC = ARRAY is an array of final STEPCC values of the jobs to be followed, parallel to JDID, for those followed jobs with JDID.FOLLOW_TYPE = 0.

JNAME = VALUE-CHAR(31)   is the jobname, if any.

NFOLL = VALUE-DEC(1-8) is the number of elements within the JDID array if FLG.O3 is set.

NRES = VALUE-DEC(0-n) is the number of different types of resources required by this job.

NSN = VALUE-DEC(0-n)   is the number of packsets required by this job.

PART = VALUE-BIT(16)   is the job's eligible partition mask.

PI = VALUE-DEC(0-n)   is the current value of the priority increment accumulator.

PNUM = VALUE-DEC(0-n)   is the selected partition number.

PRIO = VALUE-DEC(0-n)   is the current priority of the job. See JP_XXXX in CP_6_SUBS.

RATIM = VALUE-DEC(0-n) is the time in UTS format after which the job may be run, if FLG.RA is set.

RESX = VALUE-DEC(0-n) is the index into the VLR_JOBSTATS_ISRCH_RES.R array of the first resource frame for this job.

RRT = ARRAY(0:11)UBIN(18) Is the table of resource requirements, indexed by resource type:

0	Memory
1	Spindles
2	Tapes
3	Other physical devices
4-11	Pseudos 1-8

SNX = VALUE-DEC(0-n) is the index into the VLR_JOBSTATS_ISRCH_SN.S array of the first packset for for this job.

START_TIME = VALUE-DEC(0-n) is the time the job started running, in UTS format.

SUBMIT_TIME = VALUE-DEC(0-n) is the job submittal time in UTS format.

SYSID = VALUE-DEC(0-n) is the job's sysid.

TIM = VALUE-DEC(0-n) is the maximum runtime of the job, in units of 1/100 seconds.

U = VALUE-DEC(0-n) is the user number of the job, if it is running.

UNAME = VALUE-CHAR(12) is the job's user name.

WOO = VALUE-CHAR(8) is the job's WSN of origin.

WSN = VALUE-CHAR(8) is the job's effective WSN.

M$DISPLAY - Display System Information

The M$DISPLAY service returns the current values of three system load parameters:

1. The current number of active users and the current number of active users of each type,
2. The execution time multiplication factor,
3. The number of milliseconds within which 90 percent of the responses to terminal requests are made.

The response time and ETMF values apply to all operations during the last full minute of system usage.

The form of the call for this service is as follows:

CALL M$DISPLAY (FPT_DISPLAY);

The parameter for this service is:

RESULTS = VARIABLE. Specifies an area in the user's data area where the results of the M$DISPLAY service are to be stored. This area may be generated by invoking the VLR_DISPLAY macro. The default is NIL.

VLR_DISPLAY

The VLR_DISPLAY macro generates an area containing the following fields:

BUSERS = VALUE-DEC(0-n). Contains the number of currently active batch users.

ETMF = VALUE-DEC(1-n) contains the current execution time multiplication factor.

GUSERS = VALUE-DEC(0-n). Contains the number of currently active ghost users.

OUSERS = VALUE-DEC(0-n). Contains the number of currently active online users.

RESP = VALUE-DEC(0-n) contains the number of milliseconds that just exceeds the response time of 90 percent of the responses to terminal requests.

TPUSERS = VALUE-DEC(0-n). Contains the number of currently active Transaction Processing users.

USERS = VALUE-DEC(1-n). Contains the total number of currently active users.

M$MONINFO - Get Information About the Running Monitor

The M$MONINFO service returns information about the site and the running monitor. There are five information structures that may be obtained. The information returned in each of these structures is described in the VLR_SITEINFO, VLR_MONINFO, VLR_MONPTRS, VLR_SYMBINFO, and VLR_HEADER structures. Any user may obtain the information in SITEINFO and HEADER. The SPCLMM or EXMM privilege is required to obtain MONINFO, MONPTRS or SYMBINFO.

HEADER = VARIABLE. Specifies the location that receives the current header for the system. The structure that is returned is defined by the VLR_HEADER macro. This area may be generated via the VLR_HEADER macro which is described later in this section.

MONINFO = VARIABLE specifies the location of the area that is to receive the information about the running monitor. The information returned is described in the VLR_MONINFO macro.

MONPTRS = VARIABLE specifies the location of the area that is to receive pointers to various monitor tables. The pointers returned are described in the VLR_MONPTRS macro. These pointers are monitor linkage segment pointers and are not directly usable by user programs. The M$SAD monitor service may be used to access the data pointed to by these pointers.

SITEINFO = VARIABLE specifies the location of the area that is to receive the site information. The contents of this area is described in the VLR_SITEINFO macro.

SYMBINFO = VARIABLE specifies the location of the area that is to receive the symbiont scheduling information. The contents of this area are described in the VLR_SYMBINFO macro.

VLR_SITEINFO

This macro defines the structure for the SITEINFO returned by the M$MONINFO service. This information is available to any user.

ANSPROT = VALUE-DEC(0-3) Receives the level of ANS tape protection in the system. The levels are:

- 0 – Unprotected system
- 1 – Semi-protected system
- 2 – Fully protected system

MINI_ID = VALUE-CHAR(3). Is the identifier of the mini-integration for this system. This is used by Honeywell during development and should always be blanks for released (non-beta) systems.

MON_UTS = VALUE-DEC(0-n) Receives the UTS time/date of the creation of M:MON.

PATCHWEEK = VALUE-CHAR(4) Contains the installation's current patch revision week. This field cannot be set using the macro invocation.

SALUTATION = VALUE-CHAR(79) Receives the TEXTC logon salutation.

SITE_ID = VALUE-CHAR(6) Receives the installation SITE ID. Each installation is assigned a unique SITE ID by Honeywell.

SITE_NAME = VALUE-CHAR(119) Receives the site name in TEXTC format.

VERSION = VALUE-CHAR(4) Receives the system version. The first three characters are the version of the CP-6 system. The fourth character is set to 'A' for a "new files" boot (i.e., the user responds Y to NEW FILE SYSTEM?) and incremented for each boot "under files" (i.e., when the user responds N or S to NEW FILE SYSTEM?). The fourth character is incremented from A-Z, a-z, 0-9.

VLR_MONINFO

This macro defines the structure for the MONINFO returned by the M$MONINFO service. This information is available only to users with SPCLMM or EXMM privilege.

ACORE = VALUE-DEC(0-n) Receives the number of unallocated physical pages of memory that are available to users.

BOOTFLG - Receives the type of system boot (IT_BOOTFLG). The values defined for BOOTFLG are in I_SUBS_C.

INIT_UTS = VALUE-DEC(0-n) Receives the UTS time/date of the last system initialization (full recovery, disk boot, or tape boot).

MUAIS - Number of user slots.

NODE# = VALUE-DEC(0-255) is the node number for this node.

NODE_NAME = VALUE-CHAR(8) is the node name for this node.

NOUSERS = {YES|NO} indicates whether (YES) or not (NO), the system is preventing users from logging on. This bit is set by a 'N' response to the 'DO YOU WANT USERS' question asked by AARDVARK, and by the NOUSERS keyin. When set, the ON TS, ON BA, etc numbers are set to zero, and no users are permitted to start (this means no logons accepted - all types; no batch jobs started; no initial STARTUP processing by GOOSE). The first ON XX N keyin with N nonzero will reset this flag.

NUM_NODES = VALUE-DEC(0-255) is the number of nodes in the network.

PCORE = VALUE=DEC(0-n) Receives the highest physical page number.

SCOUNT = VALUE-DEC(0-n) Receives the number of screeches since the last tape boot.

SUA_UTS = VALUE=DEC(0-n) Receives the UTS time/date of the last single user abort.

VLR_MONPTRS

This macro defines the structure for the MONPTRS returned by the M$MONINFO service. This information is available only to users with SPCLMM or EXMM privilege. Note that the level 2 names in this structure do not end with the '#' character.

B$P$ Points to the beginning of the processor tables.

B$USRT$ Points to the beginning of the user tables.

N$DCT$$ Points to the array of pointers indexed by the DCT index for peripheral devices.

NI$CHT$ Points to the channel table.

NI$DVT$$ Points to the array of DVT pointers indexed by DCT.DVTX.

VLR_SYMBINFO

This macro defines the structure of the SYMBINFO results area of the M$MONINFO service. It contains the data associated with output symbiont file priority assignment set up by CONTROL.

GRANLOW (0:14) = VALUE-DEC(0-n) is an array of granule counts. The 0th element is unused; the other elements must be assigned values in a strictly decreasing fashion (i.e., if k > m then GRANLOW(k) < GRANLOW(m)). OUTSYM uses this table if GRANSCHED is nonzero to assign priorities to output symbiont files according to their granule count, as follows: the priority assigned to a K-granule file is the minimum N such that K > GRANLOW(N), unless the submittal priority is 0 or 15 (in which case submittal priority is used). (Note that for this purpose, K is the granule size of the file times the number of copies.)

GRANSCHED = VALUE-DEC(0-n) instructs OUTSYM whether (~0) or not (0) to assign output symbiont file priorities by granule count or by submittal priority. If GRANSCHED is zero, then the GRANLOW array values are meaningless.

HOLDEXPIRE = VALUE-DEC(0-n) specifies the time interval in hours for which the 'HOLD' attribute for symbiont output will be honoured. After the specified number of hours has elapsed from the time of submittal, the file will be taken out of hold status and made a candidate for output. HOLDEXPIRE is intended for use in situations where the system manager wishes to prevent files being 'held' for arbitrarily long periods of time. A value of zero specifies that no expiration is to be performed.

The default is 36.

STREAMFILEID = VALUE-CHAR(3) is the output symbiont file BOOTID being used currently.

WATCH = {YES|NO} indicates whether (YES) or not (NO), a message is to be printed on the the appropriate device console whenever an output symbiont device commences or ceases output of a file. The message names the file and the device, and is for informational purposes only.

The default is NO.

VLR_HEADER

This macro defines the structure for the current system header as returned by the M$MONINFO service.

CNT - (0-120). CNT is the size in bytes of the current header in VLR_HEADER.HEADER#.

HEADER = VALUE-CHAR(120). HEADER is the current system header.

M$PROCNAME - Return Processor Names

The M$PROCNAME service returns the names of the Shared Processors associated with the program in execution.

Each parameter for this service locates a VARIABLE where the TEXTC name of a processor is to be returned to the user. If no processor of a given type is associated, the byte count field of this TEXTC area is set to zero and the text is set to all blanks.

The VLP_NAME macro may be used to generate this area. The size must be large enough to contain the full 31 character name.

The form of the call for this service is:

CALL M$PROCNAME (FPT_PROCNAME) [ALTRET(label)];

The parameters are as follows:

ASL = VARIABLE. Locates the area where the TEXTC name of the associated Alternate Shared Library is to be returned. The default is NIL.

CP = VARIABLE. Locates the area where the TEXTC name of the associated Interactive Command Processor is to be returned. The default is NIL.

DB = VARIABLE. Locates the area where the TEXTC name of the associated Interactive Debugger is to be returned. The default is NIL.

SHARELIB = VARIABLE. Locates the area where the TEXTC name of the associated Shared Library is to be returned. The default is NIL.

SHAREPROC = VARIABLE. Locates the area where the TEXTC name of the associated Standard Processor is to be returned. The default is NIL.

M$SPRIV and M$RPRIV - Privileges

The Job Information Table (JIT) contains data describing the job and its current status. The user may read the JIT, using a based structure generated via the B$JIT macro from the B$JIT file. The JIT is divided into several sections: common data, user data, memory management data, file management data, scheduler data, job step data, resource management, and several dummy areas. The user can modify portions of the common and user JIT by using the services described next. To modify other portions of the JIT directly, the user must have the necessary privilege. A full discussion of the JIT is included in the CP-6 System Support Reference Manual.

CONTROL OF CURRENT PRIVILEGES

The Job Information Table contains flags to control any special actions allowed for the current program. The JIT includes privilege bits for a user program or for a processor running in the user's working space. (In the CP-6 system individual privileges are assigned, rather than privilege levels. Authorization of any specific privilege does not imply any other privilege.)

B$JIT.PRIV.JOB is set by the !PRIV command of IBEX. The privileges allowed a user program are established by SUPER and stored in the user authorization record. At job initiation these privilege bits are set in B$JIT.PRIV.AUTH.

Processors may also include privileges of their own. These privileges requested by an option of the Linker are stored in the head record of the run unit. At initiation of a processor fetched from the :SYS account, these bits are copied to B$JIT.PRIV.PRC. (If the run unit is not from :SYS, B$JIT.PRIV.PRC is set to zero.)

The privileges effective for the currently running user program or processor are set in B$JIT.PRIV.ACTIVE. These active privileges are the combination (logical "OR") of B$JIT.PRIV.JOB and B$JIT.PRIV.PRC. The monitor services discussed next permit changes to the active privileges. A program generally issues these monitor services to temporarily deny itself privileges it does not require and thus avoids inadvertent use of unneeded privileges. Only privileges granted to the user (by SUPER) or to a processor (by an option of the Linker) can be set or reset by the monitor services.

NOTE: The user may also use the PRIVILEGE command to change privilege bits in B$JIT.PRIV.JOB as long as the privileges requested are granted in B$JIT.PRIV.AUTH.

The M$SPRIV and M$RPRIV services set and reset the active privileges (B$JIT.PRIV.ACTIVE). At M$SPRIV the privilege bits specified on the PRIV parameter are turned on only if the corresponding bits are set in B$JIT.PRIV.JOB or B$JIT.PRIV.PRC. All bits specified and allowed are set in B$JIT.PRIV.ACTIVE. If not all specified bits are allowed, the bits representing privileges granted are set or reset, and the alternate return is then taken with the severity code set to 0.

If the AUTH option is specified, then B$JIT.PRIV.AUTH is the sole criterion used in determining if the request can be satisfied.

If the PPRIV option is specified, the normal return is taken if all bits specified are present in B$JIT.PPRIV, or if all bits on in B$JIT.PRIV.PRC are also on in B$JIT.PRIV.AUTH. No change is made to B$JIT.PRIV.ACTIVE if PPRIV is specified.

At M$RPRIV the privilege bits set on the PRIV parameter are turned off in B$JIT.PRIV.ACTIVE. If not all specified bits are on, the bits that were on are reset, and the alternate return is taken. The form of the call for this service is:

CALL M$SPRIV (FPT_PRIV)[ALTRET (label)];

or

CALL M$RPRIV (FPT_PRIV)[ALTRET (label)];

The parameters for the M$SPRIV and M$RPRIV services are as follows:

AUTH = {YES|NO} specifies, if YES, that the AUTH field from B$JIT.PRIV.AUTH is used to verify the operation instead of the JOB field. Default=NO.

PPRIV={YES|NO} specifies, if YES, that the PRIV parameter is compared against B$JIT.PPRIV. If PRIV is fully contained, or B$JIT.PRIV.PRC is fully contained, in B$JIT.PRIV.AUTH, the normal return is taken. No change is made to B$JIT.PRIV.ACTIVE.

PRIV = VALUE-BIT(36) specifies the privilege bits to be set on or off. All zeros means that no change is to be made to the privilege bits. Default=0. It is recommended that the %EQU's for the privileges contained in JIT be used to specify the privileges.

M$SSWITCH and M$RSWITCH - Pseudo Switch

These monitor services are available to set and reset pseudo switches. These switches allow the user to specify program options at run-time. The M$SSWITCH service turns on any switch specified by a bit setting on the SWITCH parameter. The M$RSWITCH service turns off any switch specified by a bit setting on the SWITCH parameter. (These functions are also available through the SWITCH command.)

The form of the call for this service is:

CALL M$SSWITCH (FPT_SWITCH);

or

CALL M$RSWITCH (FPT_SWITCH);

No alternate return is defined for these monitor services.

The parameter for these services is as follows:

SWITCH = VALUE-BIT(36) specifies the pseudo switches to set or reset. All zeros means that no change is to be made to the pseudo switches. Default=0.

M$CHGUNIT - Increment Unit Counter

The M$CHGUNIT service permits the user to increment any of eight counters, four 18-bit counters in B$JIT.JOBUNIT and four 18-bit counters in B$JIT.STEPUNIT. These counters are primarily intended for charging proprietary software for each invocation of a processor. A step counter is incremented only if step accounting or processor accounting is in effect. If a counter overflows, it is set to the maximum value; the alternate return is then taken with the severity code set to 0.

The form of the call for this service is:

CALL M$CHGUNIT (FPT_CHGUNIT) [ALTRET (label)];

The parameters for the M$CHGUNIT service are as follows:

STEP = {YES|NO} specifies, if yes, that the step counter(s) are to be incremented. NO specifies that the job counter(s) are to be incremented. Default=YES.

UNIT0 = VALUE-DEC(0 to 2**18-1) specifies the value by which to increment the first 18-bit counter. Default=0.

UNIT1 = VALUE-DEC(0 to 2**18-1) specifies the value by which to increment the second 18-bit counter. Default=0.

UNIT2 = VALUE-DEC(0 to 2**18-1) specifies the value by which to increment the third 18-bit counter. Default=0.

UNIT3 = VALUE-DEC(0 to 2**18-1) specifies the value by which to increment the fourth 18-bit counter. Default=0.

M$USRFIELD - Set JIT User Field

The M$USRFIELD service allows the user program to set selected values in B$JIT.USERWORD and B$JIT.INSTWORD. Each of these portions of the JIT is divided into four 18-bit entries. These entries are not used by the operating system and are defined by the installation or user.

The form of the call for this service is as follows:

```
CALL M$USRFIELD (FPT_USRFIELD);
```

No alternate return is defined for this service.

The parameters for the M$USRFIELD service are as follows:

FIELD0-FIELD3 = VALUE-DEC(0-n) Specifies the value to set into each of the four fields. The value 262143 specifies that the particular slot is not to be changed. 262143 is the default.

USER = {YES|NO} specifies, if yes, that values are to be set in B$JIT.USERWORD. NO specifies that values are to be set in B$JIT.INSTWORD. The default is YES. If NO is specified, EXMM privilege must be active.

Section 5

Terminal Services

Terminal services allow the user program to specify special control of terminal functions and to obtain information about terminal attributes. These services provide information about both the logical and physical attributes of the terminal. Several services allow the user to change the terminal attributes. In a normal environment, terminal attributes are not modified. However, these services are provided to accommodate users requiring special control of terminals.

The profile of a terminal, which describes the physical characteristics of the terminal, is established by the system manager via the SUPER processor discussed in the CP-6 System Support Reference Manual. These characteristics include such information as platen width, type of cursor positioning, and other terminal attributes. When the user logs on, the default profile is established unless the user specifies an alternate terminal profile. Any terminal profile can be modified somewhat by IBEX commands and the following monitor services:

M$PLATEN
M$PROFILE
M$STRMCTL
M$STRMATTR
M$STRMTAB

Probably the most commonly used terminal service is M$PROMPT. The M$PROMPT service allows a user to set a prompt for input on the terminal. This prompt is a character string that is displayed to request input from the terminal. This service permits the user to control positioning of the prompt by specifying the parameter for vertical form control (VFC). If VFC is set, the first character of the prompt character string specifies the VFC spacing.

USER INTERFACE

The services described in this section allow the user to modify the logical characteristics of a time-sharing terminal.

M$PROMPT and M$GPROMPT - Prompt Control

The M$PROMPT service changes the character string that is used to prompt for input on the terminal. The default prompt characters are:

!	for the command processor domain
>	for the debugger domain
*	for USER PROGRAMS

M$PROMPT changes the prompt for a given stream interfacing with the terminal device.

The form of the call for this service is as follows:

```
CALL M$PROMPT (FPT_PROMPT) [ALTRET (label)];
```

The M$GPROMPT service retrieves the character string that is the current prompt for input on the terminal. The size of the prompt string is returned in the actual record size in the DCB. The form of the call for this service is as follows:

CALL M$GPROMPT (FPT_PROMPT) [ALTRET (label)];

Parameters for the M$PROMPT and M$GPROMPT service are as follows:

BIN = {YES|NO} is currently unused.

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

PROMPT = VARIABLE specifies the character string to be used as the prompt on the terminal. Subsequent M$READs by this user cause the specified prompt to be displayed on the terminal. The maximum prompt length is 30 characters; this does not include VFC.

If PROMPT = NIL, the default prompt is used. The default is NIL.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

TRANS = {YES|NO} specifies transparency. If YES is specified, the prompt string is sent literally to the terminal; that is, no translation or interpretation is done. If NO is specified, the prompt is processed normally as it is sent to the terminal. The default is NO. This feature is not currently implemented.

VFC = {YES|NO} specifies whether the first character of the prompt string is the VFC. If NO is specified, the VFC default is used. The VFC default is '@' or extend, which causes the prompt to start in the current character position and the cursor/carriage is positioned to the column immediately following the last character of the prompt. Refer to the CP-6 Programmer Reference Manual for a complete list of VFC codes. The default is NO.

M$GTRMTAB and M$STRMTAB - Tab Settings

The M$GTRMTAB service to obtain the device tab setting of the terminal. Device tabs are the physical tabs of the terminal. The system uses device tabs to optimize carriage movement. Device tabs are only used by the system and do not affect logical tabs or tab characters, with the following exceptions. When an M$WRITE is issued to the terminal with a DCB that specifies ORG=TERMINAL and TRANS=YES or when tab simulation is not used (in VLP_TRMCTL TABSIM=NO), a tab character causes the terminal to move to the next device tab.

NOTE: To set normal horizontal tabulation at a terminal in order to align text, use the M$DEVICE not the M$STRMTAB service. Refer to Section 3 for a discussion of M$DEVICE.

The form of the call for this service is as follows:

CALL M$GTRMTAB (FPT_TRMTAB) [ALTRET (label)];

The M$STRMTAB service is used to set the device tab setting of the terminal. The form of the call for this service is as follows:

CALL M$STRMTAB (FPT_TRMTAB) [ALTRET (label)];

Parameters for FPT_TRMTAB are as follows:

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

TAB = VARIABLE specifies a buffer from which the tab settings are moved. The structure of the buffer is generated by the VLP_TAB macro. When specifying device tabs via VLP_TAB, MARGIN sets the first device tab and TAB specifies the remainder of the tabs. (VLP_TAB is defined in Section 3.) If the terminal has no device tabs or if the device tabs are unknown, 0 is returned for all tab positions.

The default is NIL.

M$PLATEN - Set Page Format

The M$PLATEN service is used to describe the page format on a terminal. The user can specify the following:

- o The maximum number of characters to be written per line on the terminal.
- o The number of lines to be printed following each automatic heading.
- o The number of lines to be skipped between the last print line on the page and the top of the next page.
- o The number of lines to be skipped between the top of a page and the heading line on the page.
- o The paper width in columns, height in lines, the part of that height that is not to be printed in, and the location of page boundary within the non-printing area may be described.

The parameters of M$PLATEN are shown in relation to the printer page layout in the following diagram.

```
                      -<------ WIDTH ------>-
|---------------      - - - - - - - - - - -   (PERFORATION)
|                     *                    *              |
|       |             *                    *              |
|FIRSTLINE            *                    *              |
|       |             *                    *              |
|       V                                                 |
|---------------      *TITLE AREA . . . .  *              |
|                     *PRINT LINE . . . .  *              |
|       |             *PRINT LINE . . . .  *              |
|       |             *         .          *              |
|   LENGTH            *         .          *              |
|       |             *         .          *              |
|       |             *         .          *              |
|       V             *PRINT LINE . . . .  *              |
|---------------                                          |
|                     *                    *              |
|       |             *                    *              |
|   LIMBO             *                    *              |
|       |             *                    *              |
|       V             *                    *              |
|---------------      - - - - - - - - - - - (PERFORATION) |
```

The form of the call is as follows:

CALL M$PLATEN (FPT_PLATEN) [ALTRET (label)];

Parameters for the M$PLATEN service are as follows:

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

PLATEN = VARIABLE specifies a buffer that contains the platen information. The structure of the buffer is generated by the VLP_PLATEN macro described next in this section.

The default is NIL.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

VLP_PLATEN

The VLP_PLATEN macro describes the page format of a terminal. Parameters for the macro are as follows:

EXTWID = {YES|NO} specifies whether the value of WIDTH is the number of columns by which the carriage width exceeds 254. When WIDTH = 255, this parameter is ignored.

The default is NO.

FIRSTLINE = VALUE-DEC(0-254) specifies the height of the area which is to be left blank between the beginning of the page or screen and the first legal print line.

The default is 255, which causes this parameter to be ignored.

LENGTH = VALUE-DEC(0-254) specifies the number of printable lines on the page. A value of zero specifies that the form is to be treated as one continuous page. There are FIRSTLINE+LENGTH+LIMBO total lines on one page or screen.

The default is 255, which causes this parameter to be ignored.

LIMBO = VALUE-DEC(0-254) specifies the number of lines to be left blank between the last legal print line and the bottom of the page or screen.

The default is 255, which causes this parameter to be ignored.

LPI = VALUE-DEC(0-254) is currently unused.

WIDTH = VALUE-DEC(0-254) specifies the maximum number of characters to be written per line on the terminals.

The default is 255, which causes this parameter to be ignored.

M$EOM - Set EOM Characters/Timeout

This service is used to specify the activation character set and the read time-out for the terminal. The activation character set includes all characters to be interpreted as end-of-message characters. The default activation character set is:

EOT, LF, FF, CR, SUB, FS, GS, RS, US.

The form of the call for this service is as follows:

CALL M$EOM (FPT_EOM) [ALTRET (label)];

The parameters for this service are as follows:

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

EOMTABLE = VARIABLE locates a table. The structure of the table is generated by the VLP_EOMTABLE macro which is described next in this section.

The activation set is unchanged if the parameter is NIL. The default is NIL.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

SUPPHALT = {YES|NO} YES specifies that output on the terminal may not be halted by actions of its operator (ESC-H, etc.). NO allows halting. The default is NO.

TIMEOUT = VALUE-DEC(1-255) specifies a time-out period to be used on subsequent M$READs to the terminal. If a M$READ is not completed within this time period, it is terminated with a timed-out completion type. For delayed-read devices, namely comgroup terminal stations, this time-out period will only begin counting once the first character has been sent from the device. In other words, a "silent" delayed-read device will not time out.

The units for TIMEOUT are specified by the UTYPE parameter. The default is 0 which indicates no time-out.

UTYPE = OPTION specifies the units of TIMEOUT. Options are:

MIL10	10 milliseconds
SEC	seconds
MIN	minutes

The default is SEC.

VLP_EOMTABLE

VLP_EOMTABLE generates a table that specifies the activation character set for a terminal. The contents of this table are specified as follows:

VALUES = OPTION can be set to any one of the following options:

STD Specifies the following standard end-of-message character set:

```
EOT LF FF CR SUB FS GS RS US
```

DELTA Specifies the following DELTA end-of-message character set:

```
EOT LF FF CR SUB FS GS RS US HT
[ ]      { } = /
```

ALFNUM Specifies the following non-alphanumeric end-of-message character set:

```
SOH STX ETX EOT ENQ ACK BEL BS
LF  VT  FF  CR  SO  SI  DLE DC2 DC4
NAK SYN ETB SUB FS  GS  RS  US
! " # $ % & ' ( ) * + , - . / :
; < = > ? @ [ \ ]   _ ' { | } ~
```

NGRF Specifies the following non-ASCII graphic end-of-character set:

```
SOH STX ETX EOT ENQ ACK BEL BS
LF  VT  FF  CR  SO  SI  DLE  DC2 DC4
NAK SYN ETB SUB FS  GS  RS  US
```

VALUE-DEC(0-255) [,VALUE-DEC(0-255)...] specifies up to 16 values for the 32 bytes in the EOMTABLE, as explained below.

The EOMTABLE contains 32 bytes. The first 16 bytes of EOMTABLE represent the ASCII characters with the parity bit set to zero; the second 16 bytes of EOMTABLE represent the ASCII characters with the parity bit set to one.

Each byte in EOMTABLE represents an 8-character column of the ASCII code chart. (Only the low-order 8 bits of each byte in EOMTABLE is used.) Each bit set on represents an activation character. The bit whose value is 128 corresponds to the top character in the column; 1 corresponds to the last character in the column. Thus the first value in the VALUE parameter and the first byte in EOMTABLE represent NUL (128) through BEL (1). The second value and second byte of EOMTABLE represent characters BS through SI. The third value and third byte of EOMTABLE represent characters DLE through ETB, and so forth. Figure 5-1 shows the ASCII code chart with decimal values used to calculate the VALUE parameter.

The values for the standard activation character set provide an example: VALUE= 8, 44, 0, 47, 0*12. The values correspond to the ASCII code set as follows:

Decimal 8 or binary 00001000 sets EOT as an activation character in column 1.

Decimal 44 or binary 00101100 sets LF, FF, and CR in column 2.

Decimal 47 or binary 00101111 sets SUB, FS, GS, RS, and US in column 4.

0 indicates that there are no activation characters in column 3 and 5 – 16.

EVEN BYTES OF EOMTABLE

HIGH–ORDER OCTAL DIGITS—> / LOW–ORDER OCTAL DIGIT

00	02	04	06	10	12	14	16	v	
NUL	DLE	SP	0	@	P	'	P	0	128
SOH	DC1	!	1	A	Q	a	q	1	64
STX	DC2	=	2	B	R	b	r	2	32
ETX	DC3	#	3	C	S	c	s	3	16
EOT	DC4	$	4	D	T	d	t	4	8
ENQ	NAK	%	5	E	U	e	U	5	4
ACK	SYN	&	6	F	V	f	v	6	2
BEL	ETB	'	7	G	W	g	w	7	1

ODD BYTES OF EOMTABLE

HIGH–ORDER OCTAL DIGITS—> / LOW–ORDER OCTAL DIGIT

01	03	05	07	11	13	15	17	v	
BS	CAN	(	8	H	X	h	x	0	128
HT	EM	)	9	I	Y	i	y	1	64
LF	SUB	*	:	J	Z	j	z	2	32
VT	ESC	+	;	K	[	k	[	3	16
FF	FS	,	<	L	\	l	|	4	8
CR	GS	–	=	M	]	m	}	5	4
SO	RS	.	>	N	~	n	–	6	2
SI	US	/	?	O	_	o	DEL	7	1

^
|
Decimal values for calculation of VALUE parameter of VLP_EOMTABLE

Figure 5–1. CP–6 ASCII Codes

M$SINPUT - Set to last Input

The M$SINPUT service is used to define a record that is to be processed as if it were the last input line entered by the user. This record is used for any editing functions on the next M$READ to the terminal. If the next M$READ specifies REREAD, this line is printed after the prompt is issued. This service allows the user to process a record in a file in the same manner as a record entered from the terminal. See also SINPUTSIZE in M$READ.

For DCBs open with ORG=SE, this service is used to position the cursor on the screen, and supplies the key of the record instead of the data, since the data is already known.

The form of the call is as follows:

CALL M$SINPUT (FPT_SINPUT) [ALTRET (label)];

Parameters for the M$SINPUT service are as follows:

BUF = VARIABLE locates a buffer that contains the record to be processed as the last input line. If POSITION=YES was specified, the first character of the buffer specifies where to position the carriage within this line if an M$READ with REREAD is performed. The value is a character number (1-255). If zero is specified here, the carriage is positioned at the end of the input.

For DCBs open with ORG=SE, the buffer contains the key of the record that is to be edited next, and must be KEYL bytes long. (KEYL is specified on a call to M$OPEN.) In addition, the REREADPOS option must be used instead of POSITION.

The default is NIL.

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

POSITION = {YES|NO} YES specifies that the first character of BUF provides carriage positioning (see BUF). NO specifies that the buffer is entirely text and the carriage is to be positioned after it on a REREAD.

The default is NO.

REREADPOS = VALUE-DEC(0-?) specifies where to position the carriage or cursor within the text if an M$READ with REREAD is performed. The value is a character number in the text. If zero is specified here, the carriage is positioned at the end of the text. The POSITION option overrides this one, but provides only positioning in the first 255 bytes of the input text.

The default is zero.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

M$TRMPRG - Purge Terminal Buffers

This service allows the user to release current unprocessed terminal I/O. Input buffers waiting to be read, output buffers waiting to be sent, and consecutive BREAKs can be purged.

The form of the call is as follows:

CALL M$TRMPRG (FPT_TRMPRG) [ALTRET (label)];

Parameters for the M$TRMPRG service are as follows:

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

PURGEINPUT = {YES|NO} specifies whether type-ahead data accumulated in the input buffers is to be purged.

The default is NO.

PURGEOUTPUT = {YES|NO} specifies whether data remaining in the output buffers is to be purged.

The default is NO.

RSTBRK = {YES|NO} specifies whether the current terminal break count is to be reset. RSTBRK=YES can be used to reset the break count to avoid having a CONTROL Y simulated because of four consecutive breaks.

The default is NO.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

M$DRIBBLE - Control Recording of Terminal Interactions

The M$DRIBBLE service is used to initiate or terminate the recording of information read or written to the user's terminal on a file or device assigned to the M$DRIBBLE M$DR.

The form of the call for this service is as follows:

CALL M$DRIBBLE (FPT_DRIBBLE) [ALTRET (label)];

The parameters for the service are as follows:

DRIBBLE = {YES|NO} YES specifies that all records read or written to the user's terminal are also to be written to the file. NO specifies that the dribble file is to be closed and output discontinued. The default is YES.

M$TRMISC - Set Miscellaneous Controls

The M$TRMISC service is used to allow or inhibit SENDing and BROADCASTing to the terminal by operators. The procedure call is of the form:

CALL M$TRMISC (FPT_TRMISC) [ALTRET (label)];

The parameters for this service are as follows:

ANNOUNCE = {YES|NO} indicates whether tape and disk mounts will be announced on the terminal by the following:

...MOUNTING LT#ABCDEF

The default causes this parameter to be ignored.

BROADCAST = {YES|NO} indicates whether operator BROADCAST and SENDALL keyin messages may be sent to the terminal.

The default causes this parameter to be ignored.

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

SEND = {YES|NO} indicates whether operator SEND keyin messages may be sent to the terminal.

The default causes this parameter to be ignored.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

M$GTRMCTL and M$STRMCTL - Trm Cntl

TERMINAL CONTROL/ATTRIBUTE MANIPULATION

The services described in this section allow the user to manipulate the terminal attributes and to specify special control of the terminal.

The M$GTRMCTL service provides the user with the current terminal control information. Terminal control information includes terminal control flags set by PROFILE, escape-character sequences, or M$STRMCTL.

The form of the call for this service is as follows:

CALL M$GTRMCTL (FPT_TRMCTL) [ALTRET (label)];

The M$STRMCTL service is used to programmatically set flags that control various characteristics of the terminal. Many of these flags can also be set by the terminal user through the use of escape character sequences or via a profile.

The form of the call for this service is as follows:

CALL M$STRMCTL (FPT_TRMCTL) [ALTRET (label)];

Parameters for FPT_TRMCTL are as follows:

DCB = DCBNAME specifies the name of the DCB associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

TRMCTL = VARIABLE locates a buffer that contains or receives the terminal control information. The structure of the buffer is generated by the VLP_vlptype macros described next in this section (see the VLPTYPE option). The default is NIL.

VLPTYPE = OPTION specifies which of a number of terminal control structures is to be used to set or receive the control attributes. The name of the macro describing the TRMCTL variable may be constructed by precedirg the VLPTYPE option with a prefix of VLP_, e.g. VLP_TRMCTL. The options are TRMCTL, WINDOW, and EDTCTL. Preprocessor equates for the values used for this option are constructed by following the option name with a #, and preceding the option name with TRMCTL_, e.g. TRMCTL_TRMCTL#.

The default is TRMCTL.

VLP_TRMCTL

The VLP_TRMCTL macro defines terminal control information. Parameters for the VLP are as follows:

ACTONTRN = {YES|NO} YES specifies that the activation (EOM) characters are to apply to transparent reads to the terminal. NO specifies that only activation on byte count will be performed. BREAK functions normally independently of this option.

The default causes this parameter to be ignored.

NOTE: The default activation character set assumes that a normal read, not a transparent read is performed. For example, the CR character received as '215'O is only recognized as EOM character '015'O, i.e., the value of CR after a normal read strips off the parity bit. When performing transparent reads, the user may want to call M$EOM supplying an activation character set that includes EOM characters in both odd and even parity.

APLLCNRM = {YES|NO} YES specifies that if lower case characters are sent to the terminal while it is in APL mode, they are simply printed as upper-case. NO specifies that they are to be upper-case and underscored.

The default causes this parameter to be ignored.

AUTOTABCLM = VALUE-DEC(0-254) specifies the initial cursor position to be used when a read with DVBYTE.REREAD=NO (a new record request) occurs. Any characters saved by the wordwrap facility or in type-ahead will, of course, be displayed beginning at AUTOTABCLM as well. A value of zero or one specifies that the input data is to begin immediately after the prompt.

The default causes this parameter to be ignored.

BREAKCOUNT = VALUE-DEC(0-3) specifies the current break count. A current break count of three causes the next BREAK to be interpreted as control-Y. The specification of any value greater than three for M$STRMCTL is ignored.

The default causes this parameter to be ignored.

DISPINPUT = {YES|NO} YES specifies that the system is to display input characters on the terminal (ECHOPLEX). NO specifies that input display is to be inhibited.

The default causes this parameter to be ignored.

EDITOVR = {YES|NO} YES specifies that for CRT devices that visual fidelity is to be maintained while using input editing to insert or delete characters. Visual fidelity means the image on the screen matches the input record. NO specifies that upspacing and backslashes will be displayed to indicate character insertions and deletions. For ASYNC devices, do not specify YES unless all of the following is true:

BLANKERASES = YES
CUU_CHR or CUP_ALG was specified in the profile
CUB_CHR or CUP_ALG or DEVICEBS was specified in the profile
CUF_CHR or CUP_ALG was specified in the profile
CUD_CHR or CUP_ALG or DEVICELF was specified in the profile

For hardcopy devices, YES specifies that character-delete input operations (e.g., depressing the DEL key) should overstrike the deleted character with a \ (backslash). NO specifies that the \ should simply be displayed.

ENBENTEXTSRDCHR = {YES|NO} YES specifies that when entering or exiting the Subordinate Real Device(s) (SRDs) affected by this monitor service, a character string should be sent transparently to the device. When entering, the character string specified on the last M$WRITE's KEY with KEYTYPE = ENTSRDCHR is sent. When exiting, the character string specified on the EXTSRD_CHR option in the device's profile is sent. This feature is used for preserving the status of graphics devices.

The default causes this parameter to be ignored.

FULLDUXPAPERTAPE = {YES|NO} YES specifies that terminal input is handled using the following paper tape conventions:

1. DC3 (X-OFF) and DC1 (X-ON) characters are sent to the terminal so that the input does not overflow available buffering space.

2. A DEL character on input is ignored.

3. A LF character received as the first character of an input message is ignored when M$READ is using a default input function table (CP5, CP5S1, CP5S2).

NO specifies normal processing of terminal input.

The default causes this parameter to be ignored.

HALFDUXPAPERTAPE = {YES|NO} YES specifies that the FULLDUXPAPERTAPE conventions are to be followed and that absolutely no characters are to be sent to the terminal for either output or echoing. NO specifies normal processing of terminal input.

The default causes this parameter to be ignored.

HSCROLL = {YES|NO} specifies, if YES and if RETYPOVR and EDITOVR are also true, that input editing is to occupy only one line, even if the input record is longer than the space remaining on the line after the prompt. The HSMARGIN and HSSHIFT parameters may be used to control the display characteristics of this mode.

The default causes this parameter to be ignored.

HSMARGIN = VALUE-DEC(0-254) specifies, if HSCROLL=YES, the closest the cursor is to be allowed to approach the edge of an input area without shifting a partially displayed record to reveal more of it. Values less than %SCROLL_PERCENT# (128) specify the distance in columns. Higher values specify a percentage of the width of the window as %SCROLL_PERCENT#+percent.

The default causes this parameter to be ignored.

HSSHIFT = VALUE-DEC(0-254) specifies, if HSCROLL=YES, the number of columns that an input image is to be shifted when the user attempts to move the cursor closer than HSMARGIN to an incompletely displayed end of the record. The keyword CENTER may be specified to request that the character under the cursor be positioned as closely as possible to the center of the displayed portion of the record when a shift occurs. Otherwise, the value is specified in columns or as a percentage of the width of the input area. The preprocessor equates %SCROLL_PERCENT# and %SCROLL_CENTER# may be used to set the value at runtime.

The default causes this parameter to be ignored.

INSERTMODE = {YES|NO} specifies, if YES, that when the cursor is positioned over an input character, any new characters entered are inserted to the left of the cursor, and that the DEL character deletes only the character (if any) under the cursor. NO specifies that the behavior in these situations depends on SENDBKSPACE and the position relative to the insertion window (see Programmer Reference Manual (CE40)).

The default causes this parameter to be ignored.

LOWERCASE = {YES|NO} YES specifies that any uppercase alphabetic characters input from the terminal are to be sent to the user program as lowercase letters. In addition, the following five characters are translated:

Character	Translation
@	`
[	{
\	|
]	}
↑	~

NO specifies normal processing of terminal input. This feature is provided to enable terminals that are upper-case only to input lower-case characters.

The default causes this parameter to be ignored.

MSGHALT = {YES|NO} specifies, if YES, that output to the terminal is to be halted after an operator message until the user enters the customary output continuation character.

The default causes this parameter to be ignored.

MSGLINE = {YES|NO} specifies, if YES and the profile indicates that the terminal has such a facility, that the message line of the terminal is to be used to display all operator messages.

The default causes this parameter to be ignored.

NOOPTMIZ = {YES|NO} YES specifies that carriage positioning optimization is not to be done. This is useful for writing to mini-computers and paper tape punches. NO specifies that data is to be written as efficiently as possible.

The default causes this parameter to be ignored.

OUTPUTDISCARD = {YES|NO} YES specifies that output is not sent to the terminal. M$WRITE requests appear to work to the user program but no characters are sent to the terminal. An M$READ that can be immediately satisfied by typed-ahead input is processed normally. An M$READ that cannot be satisfied by typed-ahead input turns off output-discard mode. NO specifies normal processing of output.

The default causes this parameter to be ignored.

PAGEHALT = {YES|NO} YES specifies that when the end of a page is reached, output is halted until the user types a carriage return. NO specifies that output continues in this situation. YES is often used to prevent a high speed CRT from printing faster than its operator can read.

For 3270, it is recommended that this option be used for timesharing. It is difficult to type any input (PA or PF) when data is being sent to the device; PAGEHALT provides convenient breakpoints. Use of this option together with PRINTHALT is especially helpful when using the pagewrap scrolling technique since the screen is never cleared and a new attribute character is never inserted.

The default causes this parameter to be ignored.

PARITYCHECK = {YES|NO} YES specifies that characters input from the terminal are to be checked for proper parity (see VLP_TRMATTR). A character with bad parity is echoed as a pound sign (#) and transmitted to the program as a SUB character; a parity error completion type is reported on the M$READ. NO specifies that parity is not to be checked.

The default causes this parameter to be ignored.

PRINTHALT = {YES|NO} YES specifies that if the terminal is being halted at the end of each page of output that the message (CR TO GO) is to be printed to prompt the user to continue the output. NO specifies that no prompt be issued in this situation.

The default causes this parameter to be ignored.

For 3270, it is recommended that this option be used for timesharing. It is difficult to type any input (PA or PF) when data is being sent to the device. PRINTHALT is useful because it is desirable to maintain a protected attribute character on the screen at all times. PRINTHALT effectively pre-empts the last line to provide a place for the attribute character. Use of this option together with PAGEHALT is especially helpful when using the pagewrap scrolling technique since the screen is never cleared and a new attribute character is never inserted.

RELPAGE = {YES|NO} YES specifies that if the terminal is being halted at the end of each page of output, that the page boundary will be relative to the last read which halted the terminal (was not satisfied by type-ahead). Thus the terminal would halt when the last input line had just scrolled off the top of the screen. NO specifies that the halts occur every screen size of lines independent of whether they are inputs.

The default causes this parameter to be ignored.

RETYPOVR = {YES|NO} YES specifies that retypes (ESC-R) are to overprint the line being retyped. NO specifies that they are to be printed on the next line. YES should always be specified for 3270 keyboard display devices.

The default causes this parameter to be ignored.

SCROLL = {YES|NO} YES specifies that the terminal is to be scrolled, that is, treated as though the media is a continuous form. NO specifies that when a page has been filled, a screen erase is sent and a new page started. NO should always be specified for IBM 3270 type display devices.

The default causes this parameter to be ignored.

SENDBKSPACE = {YES|NO} YES specifies that when the cursor is positioned over a character (A), not at the right edge of the insertion window, and with INSERTMODE=NO, if the user enters another character (B), then the original character, a backspace, and the overstriking character are returned to the reading program (A - BS - B). NO specifies that in this situation only the overstriking character is sent (B). Note that no backspaces may be involved in the above operation, ESC-CR (e.g.) might be used.

The default causes this parameter to be ignored.

SIMPERF ={YES|NO} YES specifies that underscores will be printed where the perforation would normally appear on the terminal form. This gives the appearance of perforated forms on terminals with continuous form paper. NO inhibits this printing.

The default causes this parameter to be ignored.

SINPUTSZ = VALUE-DEC(0-254) specifies the minimum length of lines typed which are to be remembered by the system to be re-used (ESC-D). A value of zero specifies that all lines are saved.

The default causes this parameter to be ignored.

SPACEINSERT = {YES|NO} YES specifies that horizontal tab (HT) characters input from the terminal are to be replaced by an appropriate number of blanks based on user tabstop settings prior to being placed in the M$READ buffer. NO specifies that HT characters are to be placed in the buffer unchanged. (This can result in a significant space saving in large files.)

The default causes this parameter to be ignored.

TABRELATIVE = {YES|NO} YES specifies that on input and input echoing user tabstops are relative to the column where input was prompted. NO specifies that user tabstops are relative to column 1.

The default causes this parameter to be ignored.

TABSIM = {YES|NO} YES specifies that when an HT character is sent to the terminal, the cursor is positioned to the next logical tab stop. If no tabstops are set, each tab character is replaced by a single blank. NO specifies that HT characters are sent to the terminal unchanged. (This can result in the cursor being positioned one space or several spaces depending on the device tabs of the terminal.)

The default causes this parameter to be ignored.

TRUNCATE = {YES|NO} YES specifies that output lines which exceed the platen width will be truncated to fit on one line. NO specifies that such lines will wrap to the next line.

The default causes this parameter to be ignored.

UPPERCASE = {YES|NO} YES specifies that any lowercase alphabetic characters input from the terminal are to be sent to the user program as uppercase letters. NO specifies normal processing of terminal characters.

For 3270, UPPERCASE is turned on in all supplied profiles. IBM 3270-type devices have no caps lock key, only a shift lock. For timesharing, it is convenient to have all lowercase characters shifted to uppercase. ESC-U may be used to toggle this.

The default causes this parameter to be ignored.

WORDWRAPCLM = VALUE-DEC(0-254) specifies the maximum length of lines containing blanks that will be returned to the reading program. If a longer line is entered, the characters past the space before WORDWRAPCLM will be saved for use in the next read operation. A value of zero disables this facility.

The default causes this parameter to be ignored.

WRAPPAGE = {YES|NO} YES specifies for CRTs with SCROLL=NO that the screen is to be erased one line at a time, such that the line below the cursor is always blank. NO specifies that the entire screen is to be erased after using the bottom line.

The default causes this parameter to be ignored.

VLP_EDTCTL

The VLP_EDTCTL macro defines terminal editing (ORG=SE) information. Parameters for the VLP are as follows:

HSALL = {YES|NO} specifies whether, if HSCROLL is YES (see VLP_TRMCTL), all the records of a screen editing window are to be scrolled simultaneously whenever any horizontal shift occurs. NO specifies that only the record containing the cursor is shifted by horizontal cursor motion and that a shifted record is realigned with the rest of them when the cursor is away. The repaint function (ESC -) may be used if HSALL=NO to align the other records with the cursor record.

The default causes this parameter to be ignored.

KEYINCR = VALUE-DEC(1-65535) specifies default key increment to be used to insert new records. See KEYTYPE. The maximum value 65535 requests that key incrementation is to be disabled, requiring that record insertion be performed by the host program.

The default is zero, which causes this parameter to be ignored.

KEYTYPE = OPTION specifies the key incrementation algorithm to be used to insert new records. The options are as follows:

BIN10 specifies that KEYINCR is to be divided by 10 until it fits. If KEYINCR=1000, then the first of 1000, 100, 10, and 1 that fits is used.

BINHLF specifies that KEYINCR is to be halved. If KEYINCR=32, then the first of 32, 16, 8, 4, 2, and 1 that is small enough will be used.

BIN521 specifies that division by 5, 2.5, and 2 is to be repeated. If KEYINCR=1000, then the sequence 1000, 500, 200, 100, 50, 20, 10, 5, 2, and 1 is used.

STRING specifies that key incrementation is not possible.

The default causes this parameter to be ignored.

VSMARGIN = VALUE-DEC(0-254) specifies the closest that the cursor is to be allowed to approach the top or bottom of the window without shifting the image to reveal more of the file. Values less than %SCROLL_PERCENT# (128) specify the distance in lines. Higher values specify a percentage of the height of the window as %SCROLL_PERCENT#+percent.

The default causes this parameter to be ignored.

VSSHIFT = VALUE-DEC(0-254) specifies the lines to shift the image when the user attempts to move the cursor closer than VSMARGIN to an incompletely displayed end of the file. The keyword CENTER may be specified to request that the line under the cursor be positioned as close as possible to the center of the displayed portion of the file when a shift occurs. Otherwise, the value is specified in lines or as a percentage of the height of the window. The preprocessor equates %SCROLL_PERCENT# and %SCROLL_CENTER# may be used to set the value at runtime.

The default causes this parameter to be ignored.

VLP_WINDOW

The VLP_WINDOW macro defines terminal control information. Parameters for the VLP are as follows:

BTMBRDR = {YES|NO} specifies whether space is to be reserved at the bottom to separate this window from the one below it. The default causes this parameter to be ignored, or is NO for a new window.

FWINDOW = VALUE-CHAR(4) specifies the name of the window from which this window is to be created. The name is of the form 'UCnn'. The parameter is ignored for M$STRMCTL.

The default causes this parameter to be ignored.

HBRDRCHR = VALUE-CHAR(1) specifies the single ASCII character to be used to mark the horizontal border space(s). It will fill every character position of the farthest line of the border space. The default causes this parameter to be ignored.

HBRDRSIZ = VALUE-DEC(0-15) specifies the size in lines of the horizontal border space(s) (TOPBRDR and BTMBRDR). The default is zero, which causes this parameter to be ignored. Note that the border presence parameters must be used to remove a border – HBRDRSIZ=0 won't work.

LENGTH = VALUE specifies the size of the window, in absolute or incremental lines or as a percentage of the full screen (or of the current size for an existing window). The absolute option is indicated at MACRO invocation time by the absence of + or – as the first character of the substitution expression, and at runtime or in a preprocessor expression as %G_WINDOW_ABSVAL#+lines. The percentage option may be spelled at MACRO invocation as a decimal fraction (e.g., .85), or at runtime or in a preprocessor expression as %G_WINDOW_PERCENT#+value. This option is ignored for a new window if POSITION=LEFT or RIGHT. The default is +0, which doesn't change an existing window, but is specially interpreted for a new one as .99.

LFTBRDR = {YES|NO} specifies whether space is to be reserved at the left to separate this window from the one to the left. The default causes this parameter to be ignored, or is NO for a new window.

MINLENGTH = VALUE-DEC(0-254) specifies the minimum length for this window in lines. The default is 255, which causes this parameter to be ignored unless this is a new window, in which case it means the length is to be fixed. See REMOVABLE also.

MINWIDTH = VALUE-DEC(0-254) specifies the minimum width for this window in columns. The default is 255, which causes this parameter to be ignored unless this is a new window, in which case it means the width is to be fixed. See REMOVABLE also.

ORG_PST – for M$GTRMCTL only, returns the absolute position of the top-left corner of the window. The field names are LINE (UBIN BYTE) and COLUMN (UBIN 2 Bytes).

POSITION = OPTION specifies the portion of FWINDOW that this (new) window is to occupy. The options are TOP, BOTTOM, LEFT, and RIGHT. This parameter is ignored if the window already exists.

REMOVABLE = {YES|NO} specifies whether the window size is permitted to drop below MINWIDTH or MINLENGTH. If so, the window will be removed from the screen when it is too small, but its context will be maintained. The default causes this parameter to be ignored, or is NO for a new window.

RHTBRDR = {YES|NO} specifies whether space is to be reserved at the right to separate this window from the one to the right. The default causes this parameter to be ignored, or is NO for a new window.

TOPBRDR = {YES|NO} specifies whether space is to be reserved at the top to separate this window from the one above it. The default causes this parameter to be ignored, or is NO for a new window.

VBRDRCHR = VALUE-CHAR(1) specifies the single ASCII character to be used to mark the vertical border space(s). It will fill every character position of the farthest column of the border space. The default causes this parameter to be ignored.

VBRDRSIZ = VALUE-DEC(0-15) specifies the size in columns of the vertical border space(s) (LFTBRDR and RHTBRDR). The default is zero, which causes this parameter to be ignored. Note that the border presence parameters must be used to remove a border - HVBRDRSIZ=0 won't work.

WIDTH = VALUE specifies the size of the window, in absolute or incremental columns or as a percentage of the full screen (or of the current size for an existing window). The absolute option is indicated at MACRO invocation time by the absence of + or - as the first character of the substitution expression, and at runtime or in a preprocessor expression as %G_WINDOW_ABSVAL#+columns. The percentage option may be spelled at MACRO invocation as a decimal fraction (e.g., .85), or at runtime or in a preprocessor expression as %G_WINDOW_PERCENT#+value. This option is ignored for a new window if POSITION=TOP or BOTTOM. The default is +0, which doesn't change an existing window, but is specially interpreted for a new one as .50.

M$GLINEATTR - Get Line Attributes

The GLINEATTR service provides the user with information concerning the physical connection of the terminal to the system.

The form of the call is as follows:

CALL M$GLINEATTR (FPT_GLINEATTR) [ALTRET (label)];

Parameters for the M$GLINEATTR service are as follows:

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

LINEATTR = VARIABLE locates a buffer that contains the physical line information. The structure of the buffer is generated by the VLP_LINEATTR macro described next in this section.

The default is NIL.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

VLP_LINEATTR

The VLP_LINEATTR describes the physical connection of the terminal to the system. The information in this VLP is never set by the user; the user can only look at the information. Parameters for the VLP are as follows:

FOREIGN = {YES|NO} indicates whether the terminal is connected directly to a CP-6 node (NO) or via another network (YES). If FOREIGN=YES, the other information in the buffer refers only to the physical connection of the foreign network to the CP-6 system.

HARDWIRE= {YES|NO} indicates whether the terminal is hardwired (YES) or dial-up (NO).

LINESPEED=VALUE-DEC(0-15) indicates the line speed of the terminal. The current values are:

0 =	50	BITS-PER-SECOND
1 =	75	
2 =	110	(10 CPS)
3 =	134	
4 =	150	(15 CPS)
5 =	200	
6 =	300	(30 CPS)
7 =	600	
8 =	1050	
9 =	1200	(120 CPS)
10 =	1800	
11 =	2000	
12 =	2400	
13 =	4800	
14 =	9600	
15 =	19200	

Values 0,1,3,8,10 and 11 are not currently supported.

MULTIDEVICE = {YES|NO} indicates whether this terminal is the only device on the line (NO) or is a member of a cluster (for example, IRBT, multi-drop line, foreign network).

NODE = VALUE-CHAR(4) this is the identifier of the CP-6 node to which the terminal is most directly connected.

PORT = VALUE-CHAR(4) identifies the port (see CP-6 System Support Reference Manual) to which the terminal is most directly connected.

PROFILE = VALUE-CHAR(12) specifies the current terminal profile. This is a TEXTC string in which the first byte contains the character count. Profile names are a maximum of eleven characters.

PROTXTC - the value returned for profile name on M$GLINEATTR is actually in TEXTC format. PROTXTC redefines PROFILE and includes PROTXTC.CNT and PROTXTC.TXT.

M$STRMATTR and M$GTRMATTR - Attributes

The M$STRMATTR service is used to describe physical attributes of the terminal in the rare cases that M$PROFILE is inappropriate for this purpose. The procedure call is of the form:

CALL M$STRMATTR (FPT_TRMATTR) [ALTRET (label)];

The monitor M$GTRMATTR service is used to obtain certain physical attributes of the terminal. The procedure call is of the form:

CALL M$GTRMATTR (FPT_TRMATTR) [ALTRET (label)];

The parameters for these services are as follows:

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

TRMATTR = VARIABLE locates a buffer that contains the terminal attributes. The structure of the buffer is generated by the VLP_TRMATTR macro described in this section.

The default is NIL.

VLP_TRMATTR

The VLP_TRMATTR macro describes the physical attributes of the terminal. Parameters for the VLP are described below. If a parameter is not specified the value is not changed.

APL= {YES|NO} YES indicates that the APL character set is present, or is being simulated, on the terminal. When YES is specified, input and output characters are processed as the APL character set. NO specifies that normal translation is used.

This parameter is ignored for M$STRMATTR.

ATRSCRNPOS = {YES|NO} specifies whether a field attribute definition occupies a printable column on the terminal, such that a character cannot also be displayed there.

The default causes this parameter to be ignored.

AUTONL = {YES|NO} YES specifies that the terminal will itself perform a new line function (CR+LF) immediately after printing in (or spacing through) the right margin column. NO means that the cursor will move to the right of the right margin column to a column that cannot be printed in.

The default causes this parameter to be ignored.

BIN = {YES|NO} indicates whether binary output to this device is legal. This option does not apply to timesharing terminals.

The default causes this parameter to be ignored.

BLANKERASES = {YES|NO} YES means that a blank (Space) character displayed on the terminal in a position occupied by another character erases the previous character. NO means that the previous character is unchanged.

The default causes this parameter to be ignored.

CHARSETNAME = VALUE-CHAR(8) specifies the name of the terminal character set; that is, how the characters sent to and received from the terminal are translated into ASCII. Current defined values are:

APLBF APL bit-paired translation. Lowercase ASCII characters are printed as underscored uppercase characters. Device: ASYNC

APLBP APL bit-paired translation. Lowercase ASCII characters are printed as uppercase characters on output. Device: ASYNC

APLTF APL typewriter-paired translation. Lowercase ASCII characters are printed as underscored uppercase characters on output. Device: ASYNC

APLTP APL typewriter-paired translation. Lowercase ASCII characters are printed as uppercase characters on output. Device: ASYNC

ASC8 8-bit ASCII. Like ASCII-95, but it passes 8-bit characters (X'00' through X'FF') through on input rather than stripping the parity bit. Device: ASYNC

ASC64 ASCII-64 translation (the default). Device: ASYNC URP

ASC64B ASCII-64 variant translation for Teletype Model 35. This table interprets Altmode (X'7E', tilde) as escape on input. Device: ASYNC

ASC64C ASCII-64 variant translation for Teletype Model 33, Xerox 7015, Hazeltine 2000, etc. This table exchanges NOT (tilde) and OR (vertical line) in CP-6 system with right bracket and left bracket at the terminal. It also interprets Altmode (X'7E', tilde) as escape on input. Device: ASYNC

ASC95 ASCII-95 translation. Device: ASYNC URP

ASC95M ASCII-95 variant translation for CDI1203 Miniterms operating in STD mode. Converts the character generated by the ESC key (without CTRL) to an ESC. Device: ASYNC

EBCI EBCDIC input is translated to CP-6 ASCII. Device: RBTD

EBCO CP-6 ASCII output is translated to EBCDIC. Device: RBTD

XDSI Xerox extended EBCDIC input is translated to CP-6 ASCII. Device: RBTD

XDSO CP-6 ASCII output is translated to Xerox extended EBCDIC. Device: RBTD

SDS7012 CP-6 ASCII operator console output is translated to an uppercase EBCDIC subset; this should be specified for Xerox model 7012 operator consoles. Device: RBTD

NONE No translation to or from CP-6 ASCII will be performed. Device: RBTD

3276STD Standard EBCDIC-ASCII translation with all characters that can be displayed by the 3276 type devices. The following translation is used for those characters not in both sets:

EBCDIC		ASCII
not-sign	<—>	vertical-arrow
cent-sign	<—>	left-bracket
solid-vert-line	<—>	right-bracket
field-mark	<—>	escape

Note: Not all translations hold for both input and output. The arrows indicate the direction of translation. Device: 3270

3276LIM Same as the previous table except useful for data-entry keyboards that do not have all keys, or for others that prefer a more obvious EBCDIC-ASCII translation at the expense of not being able to enter or display square brackets:

EBCDIC		ASCII
not-sign	<—>	tilde
solid-vert-line	<—>	broken-vert-line
cent-sign	<—>	vert arrow
left-paren	<—	left-bracket
right-paren	<—	right-bracket
broken-vert-line	—>	broken-vert-line

Device: 3270

3270 For pre-IBM 3276 type device (limited character set). Same as previous table with the following differences:

EBCDIC		ASCII
right-paren	<—	right-brace
left-paren	<—	left-brace
pound-sign	<—	accent-grave
pound-sign	<—	backslash

Device: 3270

The default is blank.

CRISNL = {YES|NO} YES specifies that when a carriage return character is sent to the terminal a new line function (CR+LF) is performed. NO specifies that CR has some other function (see DEVICECR). If CRISNL=YES is specified, DEVICECR=YES must not be specified.

The default causes this parameter to be ignored.

CURSORUP = {YES|NO} YES specifies that the terminal is capable of moving its cursor upward (in the opposite direction of line feed). NO specifies this feature is not present.

This parameter is ignored for M$STRMATTR.

DEVICEBS= {YES|NO} YES indicates that a BS character sent to the terminal causes a backspace. NO specifies that a BS character be processed normally, which may or may not cause a backspace.

The default causes this parameter to be ignored.

DEVICECR = {YES|NO} YES specifies that a carriage return character sent to the terminal causes exactly a carriage return (moves the cursor to left margin of the same line). NO means that CR has some other (or no) effect.

The default causes this parameter to be ignored.

DEVICELF = {YES|NO} YES specifies that a line feed character sent to the terminal causes exactly a line feed (moves the cursor down one line without horizontal movement.) NO means that LF has some other (or no) effect.

The default causes this parameter to be ignored.

DEVPOSOPTS = {YES|NO} indicates whether the terminal combines contiguous horizontal positioning requests into a single internal operation (device positioning optimization). YES implies that if a tab to column 10 followed by a backspace were sent to the terminal, the carriage would go directly to column 9.

The default causes this parameter to be ignored.

DEVSCROLL = {YES|NO} YES specifies that the terminal is self-scrolling. This means it prints on a continuous form or rolls old lines off its screen as new ones are added at the bottom. NO means that lines are overwritten when the screen is filled or the device is not a CRT.

The default causes this parameter to be ignored.

DVCFLWCTRALG = OPTION specifies the flow control algorithm used by a buffered device to prevent overfilling its buffer.

NONE specifies that no algorithm is used.

ENQACK or ETXACK specifies that CP-6 is to send an ENQ or ETX and wait for the device to respond with ACK whenever enough characters have been sent to fill the buffer. The size of the buffer is specified in the profile for the device.

DC1DC3 specifies that the device sends DC3 to halt the flow of characters, and DC1 to resume it. This algorithm is also commonly known as XON/XOFF.

The default causes this parameter to be ignored.

EXTWID = {YES|NO} YES indicates that WIDTH is the number of columns by which the terminal width exceeds 254. NO indicates that WIDTH specifies the number of columns in the terminal line.

The default is NO.

FCNTBL = {'CP5'|'CP5S1'|'CP5S2'|'FRM1'|'NO'} is the name of a table which tells the terminal handler how to interpret control functions. If a name unknown to the system is specified, the current table is not changed. CP5 identifies the full input function table and is the default. CP5S1 and CP5S2 identify subset tables. FRM1 identifies forms functions. NO is a table containing only necessary line-control functions (e.g., disconnect).

Table Name	Escape Characters		Special Characters		Control Strings
NO					DISCONNECT
CP5	SOH	ESC	BS	DC2	BREAK
	STX	$	HT	DC3	DISCONNECT
	EOT	(	LF	CAN	OVERRUN
	LF	)	FF	EM	PARITY CHECK
	VT	–	CR	ESC	
	CR	A–Y	DC1	DEL	
	DC2	DEL			
	DC4				
CP5S1	SOH	ESC	BS	DC2	BREAK
	STX	$	HT	CAN	DISCONNECT
	EOT	–	LF	EM	OVERRUN
	LF	A–B	FF	ESC	PARITY CHECK
	VT	D–O	CR	DEL	
	CR	Q–R			
	DC2	U–Y			
	DC4				
CP5S2	SOH	I	BS	DC2	BREAK
	EOT	J	HT	CAN	DISCONNECT
	LF	K	LF	ESC	OVERRUN
	VT	N	FF	DEL	PARITY CHECK
	CR	Q	CR		
	DC2	R			
	D	V			
	F	X			
	G	DEL			
FRM1	LF	J	BS	DC2	BREAK
	CR	K	HT	CAN	DISCONNECT
	ETB	M	LF	EM	OVERRUN
	ESC	N	CR	DEL	PARITY CHECK
	–	O	ETB		
	(	Q			
	)	R			
	A	U			
	B	V			
	D	X			
	G	Y			
	H	i			
	I				

HEIGHT = VALUE-DEC(0–254) indicates the number of lines on the screen of a screen-type terminal. HEIGHT=0(zero) for non-screen-type terminals.

The default is 255, which means that the height is not altered.

INPUT = {YES|NO} YES specifies that the terminal is capable of receiving input.

The default causes this parameter to be ignored.

LFISNL = {YES|NO} YES specifies that when a line feed character is sent to the terminal a new line function (CR+LF) is performed. NO specifies that LF has some other function (see DEVICELF). If LFISNL=YES is specified, then DEVICELF=YES must not be specified.

The default causes this parameter to be ignored.

LIMBOCLM = {YES|NO} YES indicates that a backspace operation, immediately after displaying a character in the rightmost column of the terminal, has the same effect as for any other column. NO specifies that the cursor remains in the rightmost column after a character is displayed there. Applicable only if AUTONL=NO.

The default causes this parameter to be ignored.

LOCALECHO = {YES|NO} YES indicates that any characters received from the terminal have already been displayed on the terminal. This option is especially useful for very slow connections where the normal CP-6 echo mechanism becomes distracting. NO specifies normal processing.

The default causes this parameter to be ignored.

LOWERCASEPRINTS = {YES|NO} YES indicates that the terminal is capable of printing (displaying) lower-case letters. NO indicates that the terminal is not capable of printing lower-case letters.

The default causes this parameter to be ignored.

MAXATRS = VALUE-DEC(0-254) specifies the number of field definition positions permitted on any line of the device.

The default causes this parameter to be ignored.

MINREC = VALUE-DEC(0-254) specifies the minimum length of a record to be sent to the terminal. This option does not apply to timesharing terminals. If a record of less than MINREC is output, it is padded with spaces or zeros (depending on BIN in the M$WRITE) out to MINREC. Zero specifies that there is no minimum.

The default is 255, which means that MINREC is not altered.

NO_FLDTRM = {YES|NO} specifies whether an otherwise unterminated field definition extends to the end of the device. NO specifies that such a field extends only to the end of the line.

The default causes this parameter to be ignored.

OPDTAB = {YES|NO} YES specifies that the terminal has device tabs which can be set by the operator. NO specifies that they are programmatic (PROGDTAB) or nonexistent.

The default causes this parameter to be ignored.

OUTPUT = {YES|NO} specifies whether the terminal is capable of receiving output.

The default causes this parameter to be ignored.

PARITY = OPTION specifies the type of character parity at the terminal. Possible options are:

Option	Parity
NONE	On input the parity bit is ignored. On output the parity bit of translated characters is zero and the parity bit of transparent characters is unchanged.
ZERO	Parity is 0.
ODD	Parity is odd.
EVEN	Parity is even.
ONE	Parity bit is 1.
DC	Do not change parity.

The default causes this parameter to be ignored.

PRINTTYPE = {YES|NO} YES specifies that the terminal prints on a form or screen such that vertical positioning is sensible. NO specifies that the terminal produces records only (like a card punch) and vertical positioning is meaningless.

The default causes this parameter to be ignored.

PROGDTAB = {YES|NO} YES indicates that the terminal device tabs are programmatic. NO indicates that the terminal device tabs are manual.

The default causes this parameter to be ignored.

SPCBFRPRT = {YES|NO} if SPCBFRPRT=NO then any VFC will be translated such that all paper motion will follow print (e.g., space 4 and print VFCs will be translated into write a blank line and space 4 followed by write the read data and don't space). If SPCBFRPRT=YES, no special VFC translation is done.

The default causes this parameter to be ignored.

SPEED = VALUE-DEC(0-15) indicates the line speed of the terminal. The current values are:

0 =	50 BITS - PER - SECOND.	
1 =	75	
2 =	110	(10 CPS)
3 =	134	
4 =	150	(15 CPS)
5 =	200	
6 =	300	(30 CPS)
7 =	600	
8 =	1050	
9 =	1200	(120 CPS)
10 =	1800	
11 =	2000	
12 =	2400	
13 =	4800	
14 =	9600	
15 =	19200	

Values 0,1,3,8,10 and 11 are not currently supported. The default, which is a value of 255, requests that the current speed not be changed.
WARNING: CHANGING THE SPEED OF YOUR TERMINAL TO ONE IT
DOES NOT SUPPORT MAY PREVENT YOU FROM RETURNING IT
TO A USABLE SPEED.

TRUOVRPRT = {YES|NO} YES specifies that the terminal performs overprinting. NO specifies that the terminal is not capable of overprinting (that is, upspace cannot be inhibited or multiple characters cannot be displayed in one position). NO specifies that the terminal is not capable of over printing.

The default causes this parameter to be ignored. Not currently implemented.

TTYTYPE = {YES|NO} YES specifies that the terminal is capable of displaying what it sends. YES is typically specified for a device such as a keyboard printer or CRT. NO specifies that input is not displayed.

The default causes this parameter to be ignored.

WIDTH = VALUE-DEC(0-254) specifies the number of printable columns on the terminal. This number can be increased by specifying the EXTWID parameter. (If the WIDTH parameter specified by VLP_PLATEN is greater than the VLP_TRMATTR WIDTH, the line is continued on an additional line on the terminal.)

The default is 255, which means that width is not altered.

M$PROFILE - Set Terminal Profile

The M$PROFILE service changes the terminal profile. Terminal profiles are established by the system manager via SUPER.

The form of the call is as follows:

CALL M$PROFILE (FPT_PROFILE) [ALTRET (label)];

Parameters for the M$PROFILE service are as follows:

BUF = VARIABLE locates a buffer into which the profile is returned if the user wishes to look at the profile. It is recommended that this buffer be at least 1K bytes. If BUF is nil then the M$PROFILE will be done, but nothing will be returned to the user.

The default is NIL.

DCB = DCBNAME associated with the device. The DCB must be assigned to a timesharing or comgroup terminal device.

The default is M$UC which is assigned to the user's timesharing terminal.

PROFILE = VARIABLE locates a buffer that contains the profile name. Profile names are identical to file names; therefore, the structure of the buffer may be generated with the VLP_NAME macro. Profile names are a maximum of 11 characters.

The default is NIL.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

M$SETFP - Load FEP Program

The M$SETFP service is used to invoke a FEP program to be used in communicating with a FEP device. The run unit of the FEP program resides in a file which this service describes. As a result of M$SETFP it is sent to the front-end and put into execution on behalf of the FEP device described in the call. This service is available on ORG=FPRG DCBs only. The procedure call is of the form:

CALL M$SETFP (FPT_SETFP) [ALTRET (label)];

The parameters for this service are as follows:

DCB = DCBNAME specifies the name of the DCB associated with the device. The DCB must have fep program organization (ORG=FPRG). DCB must be specified.

FPRG = VARIABLE locates a buffer that describes the FEP program to be invoked. The structure of the buffer is generated by the VLP_FPRG macro. The calling program must have input access to the file described. There is no default.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

VLP_FPRG

The VLP_FPRG macro describes a FEP program. Parameters for the macro are as follows:

ACCT = VALUE-CHAR(8) specifies the account of the FEP program file to be loaded. The default is blanks, specifying the current account.

DEBUG = {YES|NO} specifies, if yes, that this FPRG is to be debugged and that as such, it is not to be started but is instead to be held awaiting commands from a debugger such as:

DEBUG <station> ON <comgroup>

DEBUG is only honored for comgroup station connected FPRGs since DCB-connected FPRGs will be debugged if their owner is running under control of a debugger when this M$SETFP is issued.

The default is NO.

LDBYPRO = VALUE-DEC(0-511) specifies, if non-zero, that the FPRG is to be found from the associated PROFILE, not from this VLP.

NAME = VALUE-CHAR(31) specifies the name of the FEP program file to be loaded. NAME must be specified.

PASS = VALUE-CHAR(8) specifies the password of the FEP program file to be loaded. The default is blanks.

PROFILE = VALUE-CHAR specifies a profile name in TEXTC format when LDBYPRO was specified.

PSN = VALUE-CHAR(6) specifies the packset name of the FEP program file to be loaded. The default is blanks, specifying the current packset name.

M$DCLFLD - Declare a Field

The M$DCLFLD service is used to declare a field on a given virtual device. The user specifies the location of the field, initializing text if there is any, and field attributes.

The form of the call is as follows:

CALL M$DCLFLD (FPT_DCLFLD) [ALTRET (label)];

Parameters for the M$DCLFLD service are as follows:

BUF = VARIABLE locates a buffer which may contain initializing text for the field. The default is NIL.

CLM = VALUE-DEC(1-254) specifies the column that the field begins in, within the given virtual device.

DCB = DCBNAME specifies the name of the DCB associated with the device. The DCB must have ORG=FORM and be assigned to a timesharing terminal device.

DCB must be specified.

FLDATR = VARIABLE Locates a VLP_FLDATR area, which specifies the attributes of the field being defined.

ID = VALUE-DEC(0-65535) specifies a field identifier to be used when referencing this field.

LIN = VALUE-DEC(1-254) specifies the line within the virtual device that the field is to be placed on.

RRR = {YES|NO} YES specifies Return Receipt Request. This option specifies that the function is not complete until it reaches its destination correctly. Functions to a user terminal are normally considered complete when the data leaves the user buffer. There is no way to know if the data reached the terminal correctly or if any errors occurred performing the function. If RRR is set, the function is not considered complete until the data reaches the terminal and any error status has been returned. The default is NO.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

WDT = VALUE-DEC(1-254) specifies the width (number of columns) that the field will occupy within the virtual device.

VLP_FLDATR

The VLP_FLDATR macro describes the graphic rendition and input qualification attributes of a field in a virtual device.

At compile time, the graphic rendition for the field may be initialized abstractly by specifying ABSGRPRND (e.g., ABSGRPRND = EMPHASIS), or directly by specifying ABSGRPRND = NONE (the default) and specifying individual graphic rendition items (e.g., UNDSCR = YES, SLWBLN = YES).

At run time, the graphic rendition may be manipulated abstractly by setting VLP_FLDATR.ABSGRPRND to the proper abstract graphic rendition value. Or it may be manipulated directly by setting and resetting the items subordinate to VLP_FLDATR.GRPRND (e.g., VLP_FLDATR.GRPRND.UNDSCR = '1'B). Or all the graphic renditions may be manipulated directly and simultaneously by assigning the proper bit string to VLP_FLDATR.GRPRND (e.g., VLP_FLDATR.GRPRND = %G#GRPRND_UNDSCR | %G#GRPRND_SLWBLN). The G#GRPRND_... items are defined by the G#GRPRND_E macro.

The input qualification items may be initialized at compile or run time. At run time, individual qualifications may be manipulated by setting and resetting the items subordinate to VLP_FLDATR.QLF (e.g., VLP_FLDATR.QLF.ACPNMR = '1'B), or all qualifications may be manipulated simultaneously by assigning the proper bit string to VLP_FLDATR.QLF (e.g., VLP_FLDATR.QLF = %G#QLF_ACPNMR | %G#QLF_ACPALP). The G#QLF_... items are defined by the G#QLF_E macro. Parameters are as described below.

ABSGRPRND = {NONE | NULL | HIDDEN | EMPHASIS | INPUT | OUTPUT | ERROR | WARNING} specifies the abstract graphic rendition to be applied to this field. The corresponding graphic rendition set is determined by the device's profile. NONE (the default) indicates the graphic rendition is not specified abstractly, but by the GRPRND item and its subordinate definitions.

ACPALL = {YES|NO} specifies whether or not to allow the input of all characters into this field.

ACPALP = {YES|NO} specifies whether or not to allow the input of alphabetic characters into this field.

ACPGRP = {YES|NO} specifies whether or not to allow the input of graphic characters into this field.

ACPNMR = {YES|NO} specifies whether or not to allow the input of numeric characters into this field.

CNS = {YES|NO} specifies whether this field is constant. YES means don't erase this field unless the erase function explicitly declares that constant fields are to be erased. (See the CNS option of M$ERASE.) Constant fields are implicitly protected, to a greater extent than fields that are only 'protected'.

DCRINT = {YES|NO} specifies whether or not the field is to be displayed with decreased intensity.

FSTBLN = {YES|NO} specifies whether or not the field is to be displayed with fast blinking.

HDN = {YES|NO} specifies whether or not the field is to be hidden (not displayed).

INCINT = {YES|NO} specifies whether or not the field is to be displayed with increased intensity.

MSTENT = {YES|NO} specifies whether or not data must be input into this field.

PRT = {YES|NO} specifies whether or not to protect this field. Data cannot be input into a field that is protected. (Also see the PRT option of M$ERASE.)

PRTGRD = {YES|NO} specifies whether or not to protect and guard this field. (The "guard" feature is not currently implemented.) Data cannot be input into a field that is protected.

RCRVID = {YES|NO} specifies whether or not the field is to be displayed with reverse video.

SLWBLN = {YES|NO} specifies whether or not the field is to be displayed with slow blinking.

UNDSCR = {YES|NO} specifies whether or not the field is to be displayed with underscoring.

M$MDFFLD - Modify a Field

The M$MDFFLD service is used to modify a field on a given virtual device. The user specifies a key defining the location of the field, initializing text if there is any, and field attributes(optionally). If no key is specified, all fields selected by the M$SLCFLD service are modified.

The form of the call is as follows:

CALL M$MDFFLD (FPT_MDFFLD) [ALTRET (label)];

Parameters for the M$MDFFLD service are as follows:

BUF = VARIABLE locates a buffer which may contain initializing text for the field. The default is NIL.

DCB = DCBNAME associated with the device. The DCB must have ORG=FORM and be assigned to a timesharing terminal device.

DCB must be specified.

FLDATR = VARIABLE locates a VLP_FLDATR area, which specifies the attributes of the field being defined.

KEY = VARIABLE locates an optional area containing a key to identify which field is to be modified. If no key is specified, all selected fields are modified.

KEYTYPE = {COORD|FLDID|NONE} specifies the keytype to be used on this function.

RRR = {YES|NO} YES specifies Return Receipt Request. This option specifies that the function is not complete until it reaches its destination correctly. Functions to a user terminal are normally considered complete when the data leaves the user buffer. There is no way to know if the data reached the terminal correctly or if any errors occurred performing the function. If RRR is set, the function is not considered complete until the data reaches the terminal and any error status has been returned. The default is NO.

SETGRPRND = {YES|NO} specifies whether or not the field's graphic rendition is to be changed.

SETQLF = {YES|NO} specifies whether or not the field's qualification attributes are to change.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

M$SLCFLD - Select a Field for Modification

The M$SLCFLD service is used to select a field for future modification by other field related services. The form of the call is as follows:

CALL M$SLCFLD (FPT_SCLFLD) [ALTRET (label)];

Parameters for the M$SLCFLD service are as follows:

DCB = DCBNAME associated with the device. The DCB must have ORG=FORM and be assigned to a timesharing terminal device.

DCB must be specified.

KEY = VARIABLE locates an optional area containing a key to identify which field is to be selected. If no key is specified, all fields in the given virtual device will become selected.

KEYTYPE = {COORD|FLDID|NONE} specifies the keytype to be used on this function.

RRR = {YES|NO} YES specifies Return Receipt Request. This option specifies that the function is not complete until it reaches its destination correctly. Functions to a user terminal are normally considered complete when the data leaves the user buffer. There is no way to know if the data reached the terminal correctly or if any errors occurred performing the function. If RRR is set, the function is not considered complete until the data reaches the terminal and any error status has been returned. The default is NO.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

M$RLSFLD - Release a Field

The M$RLSFLD service is used to release a field within a given virtual device. If no key is specified, all selected fields are released.

The form of the call is as follows:

CALL M$RLSFLD (FPT_RLSFLD) [ALTRET (label)];

Parameters for the M$RLSFLD service are as follows:

DCB = DCBNAME associated with the device. The DCB must have ORG=FORM and be assigned to a timesharing terminal device.

DCB must be specified.

KEY = VARIABLE locates an optional area containing a key to identify which field is to be released. If no key is specified, all selected fields are released.

KEYTYPE = {COORD|FLDID|NONE} specifies the keytype to be used on this function.

RRR = {YES|NO} YES specifies Return Receipt Request. This option specifies that the function is not complete until it reaches its destination correctly. Functions to a user terminal are normally considered complete when the data leaves the user buffer. There is no way to know if the data reached the terminal correctly or if any errors occurred performing the function. If RRR is set, the function is not considered complete until the data reaches the terminal and any error status has been returned. The default is NO.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

M$ERASE - Erase a Field

The M$ERASE service is used to erase a given field. If no key is specified, all selected fields will be erased, subject to the PRT and CNS options.

The form of the call is as follows:

CALL M$ERASE (FPT_ERASE) [ALTRET (label)];

Parameters for the M$ERASE service are as follows:

CNS = {YES|NO} specifies whether or not constant fields are to be erased. Erasing unprotected fields is implicit.

DCB = DCBNAME associated with the device. The DCB must have ORG=FORM and be assigned to a timesharing terminal device.

DCB must be specified.

KEY = VARIABLE locates an optional area containing a key to identify which field is to be erased. If no key is specified, all selected fields are erased.

KEYTYPE = {COORD|FLDID|NONE} specifies the keytype to be used on this function.

PRT = {YES|NO} specifies whether or not protected fields are to be erased. Erasing unprotected fields is implicit.

RRR = {YES|NO} YES specifies Return Receipt Request. This option specifies that the function is not complete until it reaches its destination correctly. Functions to a user terminal are normally considered complete when the data leaves the user buffer. There is no way to know if the data reached the terminal correctly or if any errors occurred performing the function. If RRR is set, the function is not considered complete until the data reaches the terminal and any error status has been returned. The default is NO.

STATION = VARIABLE locates an area containing the name of the station to which this service is to be applied when used on COMGROUPs. If the DCB specified is assigned to a comgroup this parameter may be specified; if not it is ignored. The VLP_STATION macro is used to generate the station name. The default is NIL.

B$TERMINAL_ID - Structure Defining a Terminal ID

The B$TERMINAL_ID macro may be used to define a structure containing terminal identification information. The subfields which may be specified are:

LEV - UBIN BYTE.

SUBDEVICE The subdevice trying to logon (if applicable).

TERM.CHANNEL - UBIN.

TERM.SUBSUBDEVICE - UBIN.

TERM.TERM_NAME - CHAR(8).

TTYP - UBIN BYTE.

Section 6

Exceptional Condition Services

INTRODUCTION

The monitor services described in this section allow the user program to take control at a specific location for processing whenever an exceptional condition occurs. Also described in this section are services that allow the program to return after processing the exceptional condition and to simulate and test certain of the conditions.

An exceptional condition is defined as an event that causes an interruption of the normal flow of program execution, such as program exit or abort, timer run-out, no-wait I/O completion, monitor service request errors, etc.

ESTABLISHING EXCEPTIONAL CONDITION CONTROL

Exceptional condition control may be established for each of these four domains: Command Processor (CP), Debugger (DELTA), Alternate Shared Library (ASL) and user. A domain establishes its exceptional condition controls by issuing the following service requests:

Any monitor service call with ALTRET	To take control when the monitor detects an error on any monitor service request.
M$TRAP	To take control of machine traps and errors on monitor service calls when ALTRET is not specified.
M$STIMER	To take control after a time interval.
M$INT	To take control of a terminal "Break" key-in.
M$EVENT	To take control on completion of an event, such as no-wait I/O completion.
M$XCON	To take control of program exits or aborts.

These services simply establish the entry point to receive control when the condition occurs. The services listed first take precedence over the M$XCON request for control. For instance, if the user requests control at a specific trap condition, the M$TRAP procedure takes control; otherwise, the system passes control to the exit procedure if the same fault occurs. The exit procedure receives control at a variety of other conditions as well: exceeding LIMIT card specifications, operator aborts, line hangups, and program exits.

The monitor maintains an Exceptional Condition Control Block (ECCB) for each of the domains in the user's Read-Only Segment. The ECCB contains the entry points of the procedure to be entered when one of the Exceptional Condition occurs. It also contains a word of control flags.

The %B$ECCB macro may be used to generate a BASED structure of the ECCB. The pointer to this structure (B$ECCB$) may be obtained by SYMREFing B$ECCB$ and LINKing with B_USRPTRS_D in :LIBRARY. The format of this structure can be found in Appendix A. The sub-fields within B$ECCB have the following meaning:

B$ECCB.ARITH contains the address of the procedure to be entered on machine faults of the ARITHMETIC class.

B$ECCB.BRK contains the address of the procedure to be entered on break key interrupt.

B$ECCB.ERR contains the address of the procedure to be entered on machine faults of the ERROR class.

B$ECCB.EVENT contains the address of the procedure to be entered on event completion.

B$ECCB.FLAGS is a 9 bit field that contains bit settings used to indicate the user's request for Exceptional Condition control as follows:

B$ECCB.FLAGS.ARTHSET = VALUE-BIT(1). When set indicates that control over the Arithmetic class of faults has been established via M$TRAP.

B$ECCB.FLAGS.BRKSET = VALUE-BIT(1). When set indicates that control over the break key interrupt has been established via M$INT.

B$ECCB.FLAGS.ERRSET = VALUE-BIT(1). When set indicates that control over the Error class of faults has been established via M$TRAP.

B$ECCB.FLAGS.EVTSET = VALUE-BIT(1). When set indicates that control at event completion has been established via M$EVENT.

B$ECCB.FLAGS.PMMESET = VALUE-BIT(1). When set indicates that control over monitor service errors has been established via M$TRAP.

B$ECCB.FLAGS.PROGSET = VALUE-BIT(1). When set indicates that control over the Programmed class of faults has been established via M$TRAP.

B$ECCB.FLAGS.TMRSET = VALUE-BIT(1). When set indicates control after a timer interval has been established via M$STIMER.

B$ECCB.FLAGS.XCONSET = VALUE-BIT(1). When set indicates that Exit Control has been established via M$XCON.

B$ECCB.FLAGS.YCSET = VALUE-BIT(1). When set indicates that control over the Control-Y GO sequence has been established via M$EVENT.

B$ECCB.FLTFLG contains bit settings to indicate which of the hardware faults are under the control of the user.

B$ECCB.FLTFLG.COMM = VALUE-BIT(1). When set indicates that control over Command faults has been established.

B$ECCB.FLTFLG.DCHK = VALUE-BIT(1). When set indicates that control over Divide-check faults has been established.

B$ECCB.FLTFLG.DERL = VALUE-BIT(1). When set indicates that control over Derail faults has been established.

B$ECCB.FLTFLG.FALT = VALUE-BIT(1). When set indicates that control over Fault-tag faults has been established.

B$ECCB.FLTFLG.HDWR = VALUE-BIT(1). When set indicates that control over Hardware faults has been established.

B$ECCB.FLTFLG.HREG = VALUE-BIT(1). When set indicates that the History Registers are to be placed in the user's Task Control Block on entry to the fault handler procedure.

B$ECCB.FLTFLG.IPR = VALUE-BIT(1). When set indicates that control over Illegal Procedure faults has been established.

B$ECCB.FLTFLG.LOCK = VALUE-BIT(1). When set indicates that control over Lock-up faults has been established.

B$ECCB.FLTFLG.MEM = VALUE-BIT(1). When set indicates that control over Memory faults has been established.

B$ECCB.FLTFLG.MME = VALUE-BIT(1). When set indicates that control over MME faults has been established.

B$ECCB.FLTFLG.MPG = VALUE-BIT(1). When set indicates that control over Missing Page faults has been established.

B$ECCB.FLTFLG.MSEG = VALUE-BIT(1). When set indicates that control over Missing Segment faults has been established.

B$ECCB.FLTFLG.OVFL = VALUE-BIT(1). When set indicates that control over Overflow faults has been established.

B$ECCB.FLTFLG.SEC1 = VALUE-BIT(1). When set indicates that control over Security faults, Class 1, has been established.

B$ECCB.FLTFLG.SEC2 = VALUE-BIT(1). When set indicates that control over Security faults, Class 2, has been established.

B$ECCB.PMME contains the address of the procedure to be entered on Monitor Service Request errors when ALTRET is not specified on the Service Request call.

B$ECCB.PROG contains the address of the procedure to be entered on machine faults of the PROGRAMMED class.

B$ECCB.STIMER contains the address of the procedure to be entered after a time interval.

B$ECCB.XCON contains the address of the procedure to be entered for exit control processing.

B$ECCB.XCONF is used to indicate the Exit Control conditions on a domain basis as follows:

B$ECCB.XCONF.CL3 = VALUE-BIT(1). When set indicates that a Class 3 Exit Control condition is in progress.

B$ECCB.XCONF.CPSAVE = VALUE-BIT(1). When set indicates that exit control is in progress for SAVE from a Command Processor.

B$ECCB.XCONF.GET = VALUE BIT(1). When set indicates that exit control is in progress for GET.

B$ECCB.XCONF.LIMIT = VALUE BIT(1). When set indicates that limits have been incremented for exit control processing.

B$ECCB.XCONF.LNK = VALUE-BIT(1). When set indicates that exit control is in progress for M$LDTRC.

B$ECCB.XCONF.NOTCB = VALUE-BIT(1). When set indicates that the Task Control Block is full.

B$ECCB.XCONF.PROG = VALUE-BIT(1). When set indicates that the executing program is in the exit status.

B$ECCB.XCONF.SAVE = VALUE-BIT(1). When set indicates that exit control is in progress for M$SAVE.

Saving the Exceptional Condition Environment

When an exceptional condition occurs over which the user requested control, the system saves the environment of the interrupted program. This action enables the user to return to the point of interruption and resume normal processing, if the user determines that this is possible.

When an exceptional condition processing procedure is entered, the environment of the interrupted program as well as information specific to the exceptional condition is stored in the program Task Control Block (TCB). The TCB for the user program or standard shared processor is in the read-only-segment. The TCB consists of two parts:

1. A fixed area for the program environment when an ALTRET code sequence is to be entered following a monitor service request.

2. An Exceptional Condition Stack containing one or more frames. A frame is allocated, filled with the program environment from the Safe-Store Stack, and pushed onto the top of the stack for each exceptional condition. The stack is "popped" (that is, the top frame is released) by the user after the exceptional condition is processed.

The environment includes an Exceptional Condition Code (ECC) and an Exceptional Condition Sub-code. The values for the ECCs and Sub-codes are defined in the %SUB_EXC macro. The ECC indicates the type of condition that occurred. (See Table 6-1).

Table 6-1. Exceptional Condition Codes

ECC	Value	Meaning
%ECC_TIMER#	0	Timer run-out
%ECC_EVENT#	1	Event Completion
%ECC_INT#	2	Time-sharing terminal BRK key-in
%ECC_XCON#	3	Exit condition, normal or abnormal
%ECC_PMME#	4	Monitor service error without ALTRET (a)
%ECC_ARITH#	5	Arithmetic fault
%ECC_PROG#	6	Programmed fault
%ECC_ERROR#	7	Errors fault
%ECC_TCBFULL#	98	Insufficient space in TCB stack to allocate an Exceptional Condition frame (b)
%ECC_ALT#	99	Monitor services error with ALTRET specified

a Differs from ALTRET processing in that the environment is saved in an Exceptional Condition Stack frame in this case. Otherwise the exit procedure takes control if the user requested exit control.

b Environment is saved in the ALTRET frame of the TCB.

The Exceptional Condition Sub-codes defined in the %SUB_EXC macro will be discussed later in this section.

Figure 6-1 illustrates the TCB, the position of the exceptional condition stack and the layout of the exceptional condition frame and the ALTRET frame. (The minimum size of the exceptional condition stack is 68 words; the minimum size of the TCB is 140 words.)

In addition to the 64-word environment from the Safe-Store Stack, the system stores four or more words of information describing the condition that occurred. The system then transfers control to the exceptional condition procedures.

NOTE: If there is insufficient space in the exceptional condition stack to contain the environment when a condition occurs, the monitor places the environment in the TCB ALTRET frame. The monitor then performs a program exit; exit control processing (if established) is entered.

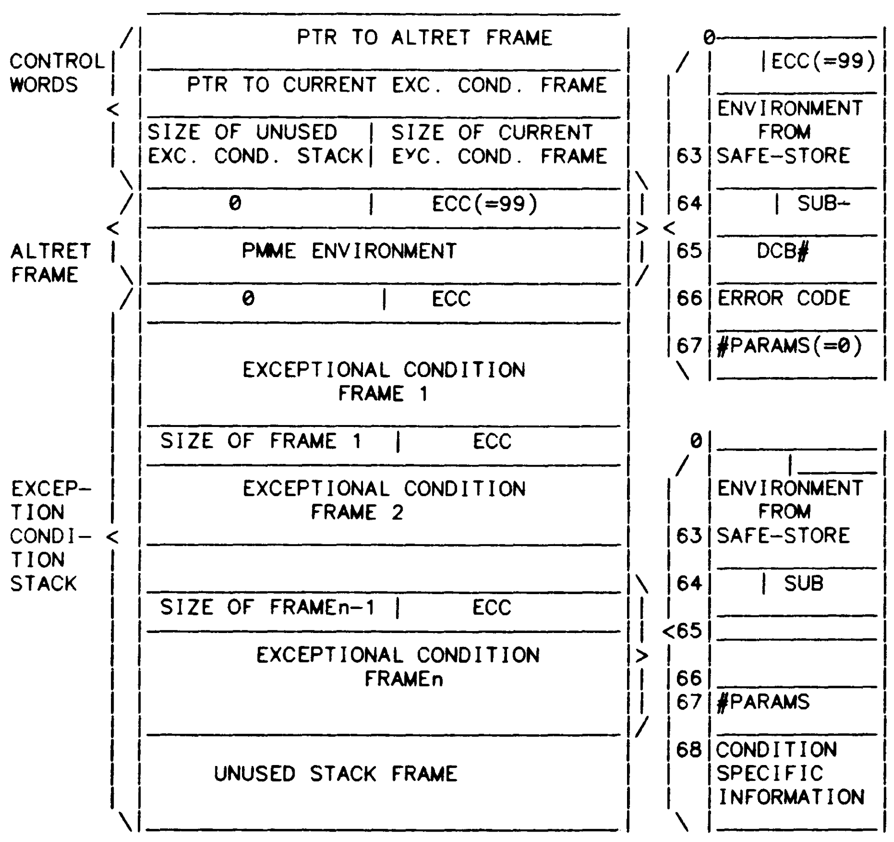
CONTROL WORDS
PTR TO ALTRET FRAME
PTR TO CURRENT EXC. COND. FRAME
SIZE OF UNUSED EXC. COND. STACK
SIZE OF CURRENT EXC. COND. FRAME
0
ECC(=99)
ALTRET FRAME
PMME ENVIRONMENT
0
ECC
EXCEPTIONAL CONDITION FRAME 1
SIZE OF FRAME 1
ECC
EXCEPTION CONDITION STACK
EXCEPTIONAL CONDITION FRAME 2
SIZE OF FRAMEn-1
ECC
EXCEPTIONAL CONDITION FRAMEn
UNUSED STACK FRAME
0
ECC(=99)
ENVIRONMENT FROM SAFE-STORE
63
64
SUB-
65
DCB#
66
ERROR CODE
67
#PARAMS(=0)
0
ENVIRONMENT FROM SAFE-STORE
63
64
SUB
65
66
67
#PARAMS
68
CONDITION SPECIFIC INFORMATION

Figure 6-1. TCB, ALTRET Frame, Exceptional Condition Stack Frame

B$TCB

The %B$TCB macro may be used to generate a based structure for accessing data on the TCB. Sub-fields in this structure are as follows:

B$TCB.ALT$ = PTR. Contains a pointer to the ALTRET frame in the TCB.

B$TCB.CURRSZ = VALUE-UBIN(18). Contains the size of the top frame on the TCB Exceptional Condition Stack.

B$TCB.STK$ = PTR. Contains a pointer to the top frame in the TCB Exceptional Condition Stack.

B$TCB.TCBAVSZ = VALUE-UBIN(18). Contains the number of unused words remaining in the TCB Exceptional Condition Stack.

B$EXCFR

B$EXCFR is a based structure which may be used to access the fixed portion on any TCB frame. The structure describes in detail the Safe-Store portion of the environment and the fixed words of information found in any Exceptional Condition frame. The format of this structure can be found in Appendix A. The sub-fields in B$EXCFR have the following meaning:

B$EXCFR.ALTEMPTY = VALUE-SBIN HALF. Contains a non-zero value when the ALTRET frame is empty. The ALTRET frame is marked empty on initial entry to a run-unit and after execution of an M$MERC, M$RETRY or M$SENV monitor service. This field has meaning only in an ALTRET frame and is redefined as PREVSZ in a TCB Stack frame.

B$EXCFR.ASR = VALUE-BIT(72). Contains the contents of the Argument Stack Register at the time of the exceptional condition. B$EXCFR.ASR contains the same sub-fields as does B$EXCFR.ISR.

B$EXCFR.BRKCNT is the number of terminal breaks received since the last read. This is for break condition only.

B$EXCFR.CODE = VALUE-UBIN(18). For ALTRET frames, contains the Service Request Code. This field may also be referenced by B$EXCFR.FPT.

B$EXCFR.DCB# = VALUE-UBIN(36). For ALTRET frames, contains the number of the DCB on which the error occurred.

B$EXCFR.DRS = ARRAY(0:7)-BIT(72). Contains the contents of the Descriptor Registers at the time of the exceptional condition.

B$EXCFR.ECC = VALUE-UBIN(18). Contains the Exceptional Condition Code. (See %SUB_EXC).

B$EXCFR.EIS = ARRAY(0:7)-UBIN(36). Contains the EIS Pointer and Length Registers when the exceptional condition caused an EIS multiword instruction to be interrupted or aborted in the middle of the instruction.

B$EXCFR.ERR is the 36 bit error code. The error code contains the following:

B$EXCFR.ERR.ERR# = VALUE-UBIN(14). Contains the error code (see file B_ERRORS_C).

B$EXCFR.ERR.FCG = VALUE-BIT(12). Contains the Functional Code Group portion of the error code.

B$EXCFR.ERR.MID = VALUE-BIT(6). Contains the Module ID portion of the error code.

B$EXCFR.ERR.MON = VALUE-BIT(1). Indicates that the error code was generated by the monitor.

B$EXCFR.ERR.SEV = VALUE-UBIN(3). Contains the severity of the error.

B$EXCFR.EVID = VALUE-UBIN(36). For event frames, contains the event ID.

B$EXCFR.EVSC = VALUE-UBIN(18). Contains the event sub-code if this is an event frame.

B$EXCFR.FCODE = VALUE-BIT(5). Contains the fault code.

B$EXCFR.FPT.FCG = VALUE-UBIN(6). Contains the Service Request Functional Code Group.

B$EXCFR.FPT.PMME = VALUE-UBIN(12). Contains the Service Request Code. See Appendix A for the meaning of this code.

B$EXCFR.IC = VALUE-UBIN(18). Contains the Instruction Counter at the time of the exceptional condition.

B$EXCFR.IR = VALUE-BIT(18). Contains the value of the Indicator Register at the time of the exceptional condition. This field is REDEFined as follows:

B$EXCFR.IRBIT.CARRY = VALUE-BIT(1). Carry Indicator.

B$EXCFR.IRBIT.EXOF = VALUE-BIT(1). Exponent Overflow Indicator.

B$EXCFR.IRBIT.EXUF = VALUE-BIT(1). Exponent Underflow Indicator.

B$EXCFR.IRBIT.HEX = VALUE-BIT(1). Hex Floating Point Indicator.

B$EXCFR.IRBIT.MIR = VALUE-BIT(1). Multiword Instruction Interrupt Indicator.

B$EXCFR.IRBIT.MM = VALUE-BIT(1). Master Mode Indicator.

B$EXCFR.IRBIT.NEG = VALUE-BIT(1). Negative Indicator.

B$EXCFR.IRBIT.OF = VALUE-BIT(1). Overflow Indicator.

B$EXCFR.IRBIT.OFM = VALUE-BIT(1). Overflow Mask Indicator.

B$EXCFR.IRBIT.PE = VALUE-BIT(1). Parity Error Indicator.

B$EXCFR.IRBIT.PM = VALUE-BIT(1). Parity Mask Indicator.

B$EXCFR.IRBIT.TR = VALUE-BIT(1). Tally Runout Indicator.

B$EXCFR.IRBIT.TRC = VALUE-BIT(1). Truncation Indicator.

B$EXCFR.IRBIT.ZERO = VALUE-BIT(1). Zero Indicator.

B$EXCFR.ISR = VALUE-BIT(72). Contains the contents of the Instruction Segment Register at the time of the exceptional condition. B$EXCFR.ISR contains the following sub-fields:

B$EXCFR.ISR.BASE = VALUE-SBIN WORD. Contains the virtual byte address which is relative to the working space defined in the Working Space Register defined in the WSR field.

B$EXCFR.ISR.BOUND = VALUE-UBIN(20). Contains the maximum valid byte address within the segment. BOUND is relative to the BASE.

B$EXCFR.ISR.FLAGS = VALUE-BIT(9). Contains the segment descriptor flags, as follows:

B$EXCFR.ISR.FLAGS.BOUND = VALUE-BIT(1). If set, the BOUND field is the maximum effective address. If reset, the Segment is empty; BOUND is irrelevant.

B$EXCFR.ISR.FLAGS.CACHE = VALUE-BIT(1). When set Cache is utilized on memory references using this descriptor.

B$EXCFR.ISR.FLAGS.EXU = VALUE-BIT(1). If set, execute allowed.

B$EXCFR.ISR.FLAGS.MON = VALUE-BIT(1). Flag is reserved for use by the software.

B$EXCFR.ISR.FLAGS.PRIV = VALUE-BIT(1). If set, privileged mode.

B$EXCFR.ISR.FLAGS.READ = VALUE-BIT(1). If set, read allowed.

B$EXCFR.ISR.FLAGS.SAVE = VALUE-BIT(1). If set, the Descriptor may be stored in a Descriptor Segment by the STDn instruction.

B$EXCFR.ISR.FLAGS.SEGMENT = VALUE-BIT(1). If set the Segment is present. If reset use of this descriptor will cause a Missing Segment Fault.

B$EXCFR.ISR.FLAGS.WRITE = VALUE-BIT(1). If set, write allowed.

B$EXCFR.ISR.TYPE = VALUE-UBIN(4). Defines the descriptor type. 0 means that the descriptor frames instruction/operand storage. 1 means that the descriptor frames descriptor storage.

B$EXCFR.ISR.WSR = VALUE-UBIN(3). Contains the Working Space Register to be used with this descriptor.

B$EXCFR.LSR = VALUE-BIT(72). Contains the contents of the Linkage Segment Register at the time of the exceptional condition. B$EXCFR.LSR contains the same sub-fields as does B$EXCFR.ISR.

B$EXCFR.P# = VALUE-UBIN(36). Contains the number of additional words on this stack frame.

B$EXCFR.PREVSZ = VALUE-UBIN(18). Contains the number of words in the previous Stack frame. If this is the first frame on the Exceptional Stack the value will be zero. This field is redefined as ALTEMPTY in an ALTRET frame.

B$EXCFR.PRS = ARRAY(0:7) PTR. Contains the contents of Pointer Registers 0-7 at the time of the exceptional condition.

B$EXCFR.PSR = VALUE-BIT(72). Contains the contents of the Parameter Stack Register at the time of the exceptional condition. B$EXCFR.PSR contains the same sub-fields as does B$EXCFR.ISR.

B$EXCFR.REGS.A = VALUE-UBIN(36). Contains the contents of the Accumulator Register at the time of the Exceptional Condition.

B$EXCFR.REGS.E = VALUE-BIT(8). Contains the contents of the Exponent Register at the time of the exceptional condition.

B$EXCFR.REGS.Q = VALUE-UBIN(36). Contains the contents of the Quotient Register at the time of the exceptional condition.

B$EXCFR.REGS.XREG = ARRAY(0:7)-UBIN(18). Contains the contents of Index Registers at the time of the exceptional condition.

B$EXCFR.SUBC = VALUE-UBIN(18). Contains the exceptional condition sub-code. The contents of this field are dependent on the type of condition that has occurred.

The remainder of the frame is dependent on the type of condition that has occurred.

Accessing Exceptional Condition Information

The system macro library, CP_6, contains a set of macros which may be used to generate based structures for accessing the TCB. The structures B$TCB and B$EXCFR may be used to access data from the initial portion of any frame in the TCB. In addition, every program contains a linker-built pointer, B$TCB$, which can be used to access the TCB.

The following PL-6 statements access the Instruction Counter from an exceptional condition frame or ALTRET frame of the TCB:

```
%INCLUDE CP_6;              —Which provides the user access to
                              macros to generate Exceptional
                              Condition based structures.

%B$TCB;                     —Which generates the BASED structure
                              for the TCB, including these pointers:

                              B$TCB.ALT$ pointing to the ALTRET
                              frame.

                              B$TCB.STK$ pointing to the top frame
                              of the exceptional condition stack.

%B$EXCFR;                   —Which generates a BASED structure
                              for any Exceptional Condition frame,
                              including:

                              B$EXCFR.IC - the Instruction Counter
                              at the time of the exceptional condition.

DCL B$TCB$ PTR SYMREF;      —Declares the linker-built pointer
                              for the TCB.

TRAP_IC = B$TCB$->B$TCB.STK$->B$EXCFR.IC;
                            —Accesses the Instruction Counter
                              in the top Exceptional Condition
                              frame.

ALTRET_IC = B$TCB$->B$TCB.ALT$->B$EXCFR.IC;
                            —Accesses the Instruction Counter
                              in the ALTRET frame.
```

Additional structures described later in this section allow access to data stored in the upper portion of the frame. These structures are also available in the system macro library.

ALTRET Condition

The ALTRET option on a monitor service call establishes a statement to receive control in case of errors. The ALTRET label must be in the same scope as the monitor service call. Before transferring to the ALTRET label, the system saves the environment in the fixed ALTRET frame in the TCB.

The ALTRET frame contains the environment and four additional words of information. The user may access the ALTRET frame using the B$TCB$ pointer and the B$ALT or B$EXCFR based structures. The ALTRET frame is illustrated in Appendix A. The ECC (B$EXCFR.ECC) is set to %ECC_ALT#; the ECC subcode contains the functional code group (B$ALT.FPT.FCG) and the monitor service number (B$ALT.FPT.PMME). B$ALT.DCB# contains the DCB number (if a DCB was a required parameter) and B$ALT.ERR contains the error code.

The ALTRET frame is overlaid every time the monitor transfers control to an ALTRET code sequence. In the course of processing the ALTRET condition, the user may call another monitor service, but may first want to save the original ALTRET environment, because the original ALTRET environment is lost if another monitor service error occurs.

NOTE: The ALTRET frame is also overlaid if the "insufficient space in Exceptional Condition Stack" condition occurs at any exceptional condition.

The following list summarizes the monitor services that act indirectly on the TCB ALTRET frame or the ALTRET environment stored in a frame of the Exceptional Condition Stack.

Monitor Service	Function
M$SENV	Save ALTRET environment in the Exceptional Condition Stack
M$RENV	Restore ALTRET environment from top frame of Exceptional Condition Stack to TCB ALTRET frame
M$RETRY	Retry original monitor service (original environment is in ALTRET frame)
M$RETRYS	Retry original monitor service (original environment is in Exceptional Condition Stack top frame)
M$MERC	Request monitor error handling (original environment is in ALTRET frame)
M$MERCS	Request monitor error handling (original environment is in Exceptional Condition Stack top frame)

B$ALT

The %B$ALT macro may be used to generate a based structure of the ALTRET frame of the TCB. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$ALT.DCB# = VALUE-UBIN(36). Contains the number associated with the DCB referenced by this Service Request.

B$ALT.ERR is the 36 bit error code. The error code contains the following:

B$ALT.ERR.ERR# = VALUE-UBIN(14). Contains the error code (see file B_ERRORS_C).

B$ALT.ERR.FCG = VALUE-BIT(12). Contains the Functional Code Group portion of the error code.

B$ALT.ERR.MID = VALUE-BIT(6). Contains the Module ID portion of the error code.

B$ALT.ERR.MON = VALUE-BIT(1). Indicates that the error code was generated by the monitor.

B$ALT.ERR.SEV = VALUE-UBIN(3). Contains the severity of the error.

B$ALT.FPT = VALUE-UBIN(18). Contains the Service Request code as follows:

B$ALT.FPT.FCG = VALUE-UBIN(6). Contains the Service Request Functional Code Group.

B$ALT.FPT.PMME = VALUE-UBIN(12). Contains the unique Service Request code for this Functional Code Group. These codes are listed in Appendix A.

B$ALT.SSFRAME contains the copy of the program's environment at the time of the ALTRET condition. The based structure B$EXCFR should be used to access the environment.

M$TRAP - Set Trap Control

TRAP CONTROL

The user can request control at a number of hardware-detected faults and at an error on a monitor service call without ALTRET. The user can also request various simulated faults to test the trap procedure(s).

M$TRAP

The M$TRAP service identifies particular hardware detected faults at which the user wants control and specifies trap procedures. The user can change fault control and specify different trap procedures during program execution.

The user can identify procedures for three categories of faults on the ARITHMETIC, ERRORS, and PROGRAMMED parameters. For specific faults in each category the user codes a fault parameter specifying TRAP or omits the parameter to take the ABORT default. For faults in the ERRORS category, the user can request that the history registers be saved along with the environment and other data in the Exceptional Condition Stack. The enabling and disabling of this function is done via the HISTORY_REGS keyword. The user may also specify with the PMME parameter a procedure that processes errors on Monitor service calls without the ALTRET option.

It is possible to specify an entry-address for a catsgory of faults while all the faults within that category are placed in the ABORT state. However, it is an error to specify that a specific fault be put in the TRAP state if no procedure is specified or exists for that category of faults.

A call to M$TRAP may request that the current settings of the Program Trap Condition (PTC) be returned to the user. The information returned includes the entry-address for each of the four possible error conditions (ARITHMETIC, PROGRAMMED, ERRORS and PMME) and the current state of each of the hardware faults (TRAP or ABORT). The structure of the PTC area can be generated by the VLP_PTC macro. (No initial values are provided in this macro because the values are supplied at run time by a call to M$TRAP requesting that the PTC be made available to the user.) The M$TRAP FPT may specify that the PTC be returned to the user and that the current Program Trap Condition be replaced with those specified in the FPT. The Program Trap Conditions may later be reset to their previous values by calling M$TRAP specifying the PTC area that was returned on the original call and RESTORE=YES. The PTC and RESTORE parameters allow any procedure within a program to handle its own trap conditions and to restore the trap conditions to their original states before returning.

The form of the call for this service is:

CALL M$TRAP (FPT_TRAP) [ALTRET (label)];

Parameters for the M$TRAP service are as follows:

ARITHMETIC = ENTRY locates the entry-address to which control is transferred in the event of an OVERFLOW or DIVIDE_CHECK trap. Default is = NIL.

DIVIDE_CHECK = {ABORT|TRAP} specifies the action to take if illegal division is attempted. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

OVERFLOW = {ABORT|TRAP} specifies the action to take at an ARITHMETIC overflow, exponent overflow, or exponent underflow. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

ERRORS = ENTRY locates an entry-address to which control is transferred in the event of MEMORY, COMMAND, LOCKUP, IPR, MISSING_SEG, MISSING_PAGE, SECURITY_1, SECURITY_2, or HARDWARE traps. Default = NIL.

COMMAND = {ABORT|TRAP} specifies the action to take when a program issues a privileged instruction. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

HARDWARE = {ABORT|TRAP} specifies the action to take at a parity error fault, operation not complete fault, or other hardware fault such as memory fault or command fault caused by a system controller illegal action. Only jobs with Test and Diagnostic authorization can set HARDWARE = TRAP. (The M$TRAP service takes the alternate return if an unauthorized user requests control at HARDWARE faults.) The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

IPR = {ABORT|TRAP} specifies the action to take at an illegal procedure which occurs as a result of an illegal operation code, an illegal address, an illegal modifier, or an illegal instruction sequence. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

LOCKUP = {ABORT|TRAP} specifies the action to take at a condition that inhibits the recognition of an execute interrupt or interrupt type fault for a time greater than the lockup time. Lockup time is variable with a minimum of 4 milliseconds on current CPUs. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

MEMORY = {ABORT|TRAP} specifies the action to take at detection of an address that is outside the segment boundary. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

MISSING_PAGE = {ABORT|TRAP} specifies the action to take at a fault that occurs during mapping from virtual to real memory, when the obtained Page Table Word has bit 30=0 meaning "page not in memory". The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

MISSING_SEG = {ABORT|TRAP} specifies the action to take at an attempt to reference memory with a descriptor that has flag bit 28=0, meaning "segment not present". The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

SECURITY_1 = {ABORT|TRAP} specifies the action to take on one of the following conditions:

1. An attempt to execute an instruction in an illegal processor mode.

2. An attempt to modify a housekeeping page in Master Mode.

3. An attempt to modify or fetch instructions or operands from a housekeeping page while in Slave Mode.

4. Attempt to reference non-housekeeping pages Type 1 or Type 3 descriptor.

   The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

SECURITY_2 = {ABORT|TRAP} specifies the action to take at one of the following violations:

1. An attempt to violate read, write, or execute permission for a segment descriptor; or to store a descriptor into a segment for which the descriptor does not permit descriptor storage.

2. An attempt to violate write permission control for a page table word

3. An attempt to violate working space quarter isolation control for a Working Space Page Table Directory word.

The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

HISTORY_REGS = {YES|NO} specifies, if YES, that the history registers are to be placed in the Exceptional Condition Stack for faults in the ERRORS category. Default = NO.

PMME = ENTRY locates a procedure to be entered as a result of a monitor service error, if no ALTRET option is specified in the service call. Default = NIL.

PROGRAMMED = ENTRY locates the entry-address to which control is transferred in the event of an MME, DERAIL, or FAULT_TAG trap. Default = NIL.

DERAIL = {ABORT|TRAP} specifies the action to take at the execution of a DERAIL (DRL) instruction. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

FAULT_TAG = {ABORT|TRAP} specifies the action to take at the recognition of the address modifier IT, whenever T=F. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

MME = {ABORT|TRAP} specifies the action to take at the execution of a Master Mode Entry (MME) instruction. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

PTC = VARIABLE specifies, if RESTORE = NO, the area where the current settings of the Program Trap Conditions are to be returned to the user. If RESTORE=YES, the Program Trap Conditions are to be restored. The structure of the area in which the Program Trap Conditions are returned may be generated by the VLP_PTC macro. If RESTORE = YES, all other FPT parameters are ignored. Default = NIL.

RESTORE = {YES|NO} specifies the action to be taken when PTC is specified. It is an error to specify RESTORE=YES and not specify a PTC area. The default causes this parameter to be ignored; thus the fault flag in the ECCB is not altered.

B$FLT

In the ARITHMETIC,ERRORS, or PROGRAMMED procedure, the user may access the top frame of the Exceptional Condition Stack via the B$EXCFR or B$FLT based structures. In the PMME procedure, the user may access this frame via the B$EXCFR or B$ALT based structures. The %B$FLT macro may be used to generate a based structure of a hardware fault exceptional condition frame. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$FLT.BRANCH_MODE = VALUE-BIT(1). If set, indicates that History Register data (if any) which follows was gathered in a mode that strobes data only for a branch instruction which branches.

B$FLT.CPU_TYPE = VALUE-UBIN(4). Contains a code to indicate the CPU Type. Note that the number of words that will be required to contain the History Register data (P#) is dependent on CPU Type. This value is set as follows:

CPU Type	CPU_TYPE	P#
L66	0	128
ELS1	2	32
DPSE	1	512

B$FLT.ERR is the 36 bit error code. The error code contains the following:

B$FLT.ERR.ERR# = VALUE-UBIN(14). Contains the error code (see file B_ERRORS_C).

B$FLT.ERR.FCG = VALUE-BIT(12). Contains the Functional Code Group portion of the error code.

B$FLT.ERR.MID = VALUE-BIT(6). Contains the Module ID portion of the error code.

B$FLT.ERR.MON = VALUE-BIT(1). Indicates that the error code was generated by the monitor.

B$FLT.ERR.SEV = VALUE-UBIN(3). Contains the severity of the error.

B$FLT.FLTREG = VALUE-BIT(36). Contains the contents of the Fault Register at the time of the fault.

B$FLT.P# = VALUE-UBIN(36). Contains the size of the History Register data that follows. No History Register information will be included in the B$FLT frame unless specifically requested via the M$TRAP Service Request. The size of the History Register information is dependent on the CPU Type. One of the following structures may be used to access the History Register data:

B$HR_ELS1
B$HR_L66
B$HR_DPSE

Please refer to B$FLT.CPU_TYPE.

B$FLT.PORT# = VALUE-UBIN(9). Contains the processor port number on the base system controller (low memory).

B$FLT.SSFRAME contains the c04y of the environment at the time of the fault. The based structure B$EXCFR should be used to access the environment.

B$FLT.SUBC = VALUE-UBIN(18). Contains the Fault Sub-code. Values (from the %SUB_EXC macro) are as follows:

Sub-code	Value	Meaning
When ECC = %ECC_ARITH# (5)		
%SUBC_OVRF#	0	Overflow Fault
%SUBC_DVCHK#	1	Divide-check Fault
When ECC = %ECC_PROG# (6)		
%SUBC_MME#	0	MME Fault
%SUBC_DRAIL#	1	Derail Fault
%SUBC_FLTAG#	2	Fault-tag Fault
When ECC = %ERR_ERROR# (7)		
%SUBC_MEMORY#	0	Memory Fault
%SUBC_CMMD#	1	Command Fault
%SUBC_LOCKUP#	2	Lockup Fault
%SUBC_IPR#	3	Illegal Procedure Fault
%SUBC_MSEG#	4	Missing Segment Fault
%SUBC_MPAGE#	5	Missing Page Fault
%SUBC_SEC2#	6	Security Fault, Class 2
%SUBC_PRTY#	7	Parity Fault
%SUBC_OPNC#	8	Operation Not Complete Fault
%SUBC_SEC1#	9	Security Fault, Class 1
%SUBC_DLNK#	10	Dynamic Linking Fault
%SUBC_MWSP#	11	Missing Working Space Fault

VLP_PTC

The VLP_PTC macro defines the structure of the Program Trap Conditions returned by M$TRAP.

ARITHMETIC = ENTRY Specifies the procedure to receive control if an arithmetic fault occurs.

ERRORS = ENTRY Specifies the procedure to receive control if a fault in the ERRORS class occurs.

FLAGS Specifies the action to take in each condition listed below. For each flag, the value '10'B means TRAP, '00'B means ABORT, '11'B is the default which means the fault flag in the ECCB is not to be changed.

FLAGS.COMMAND = VALUE-BIT(2).

FLAGS.DERAIL = VALUE-BIT(2).

FLAGS.DIVIDE_CHECK = VALUE-BIT(2).

FLAGS.FAULT_TAG = VALUE-BIT(2).

FLAGS.HARDWARE = VALUE-BIT(2).

FLAGS.HISTORY_REGS = VALUE-BIT(2).

FLAGS.IPR = VALUE-BIT(2).

FLAGS.LOCKUP = VALUE=BIT(2).

FLAGS.MEMORY = VALUE-BIT(2).

FLAGS.MISSING_PAGE = VALUE-BIT(2).

FLAGS.MISSING_SEG = VALUE-BIT(2).

FLAGS.MME = VALUE-BIT(2).

FLAGS.OVERFLOW = VALUE-BIT(2).

FLAGS.RESTORE = VALUE-BIT(2).

FLAGS.SECURITY_1 = VALUE-BIT(2).

FLAGS.SECURITY_2 = VALUE-BIT(2).

PMME = ENTRY Specifies the procedure to receive control if a monitor service error occurs on a call without the ALTRET option.

PROGRAMMED = ENTRY Specifies the procedure to receive control if a fault in the PROGRAMMED class occurs.

B$HR_L66

The B$HR_L66 macro may used to generate a based structure defining the History Register information that is appended to the standard B$FLT frame when the CPU type is L66. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$HR_L66.CU_HR = ARRAY(0:15) BIT(72). Contains the contents of the Control Unit History Registers at the time of the fault.

B$HR_L66.DU_HR = ARRAY(0:15) BIT(72). Contains the contents of the Decimal Unit History Registers at the time of the fault.

B$HR_L66.FLTFRAME = ARRAY(0:67) BIT(36). This area contains the standard B$FLT information. Use the B$FLT macro to generate a structure that may be used to access the information in this area.

B$HR_L66.OU_HR = ARRAY(0:15) BIT(72). Contains the contents of the Operations Unit History Registers at the time of the fault.

B$HR_L66.VU_HR = ARRAY(0:15) BIT(72). Contains the contents of the Virtual Unit History Registers at the time of the fault.

B$HR_DPSE

The B$HR_DPSE macro may used to generate a based structure defining the History Register information that is appended to the standard B$FLT frame when the CPU type is DPSE. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$HR_DPSE.CU_HR = ARRAY(0:63) BIT(72). Contains the contents of the Control Unit History Registers at the time of the fault.

B$HR_DPSE.FLTFRAME = ARRAY(0:67) BIT(36). This area contains the standard B$FLT information. Use the B$FLT macro to generate a structure that may be used to access the information in this area.

B$HR_DPSE.OUDU_HR = ARRAY(0:63) BIT(72). Contains the contents of the Operations Unit and Decimal Unit History Registers at the time of the fault.

B$HR_DPSE.VU1_HR = ARRAY(0:63) BIT(72). Contains the contents of the Virtual Unit 1 History Registers at the time of the fault.

B$HR_DPSE.VU2_HR = ARRAY(0:15) BIT(72). Contains the contents of the Virtual Unit 2 History Registers at the time of the fault.

B$HR_ELS1

The B$HR_ELS1 macro may used to generate a based structure defining the History Register information that is appended to the standard B$FLT frame when the CPU type is ELS1. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$HR_ELS1.FLTFRAME = ARRAY(0:67) BIT(36). This area contains the standard B$FLT information. Use the B$FLT macro to generate a structure that may be used to access the information in this area.

B$HR_ELS1.HR = ARRAY(O:15) BIT(72). Contains the contents of the History Registers at the time of the fault.

M$STRAP - Simulate Trap or Break

The M$STRAP monitor service may be used to simulate the occurrence of a hardware trap or the break of a time-sharing terminal.

The M$STRAP service simulates a trap by imitating the occurrence of a fault in the ARITHMETIC, ERRORS, or PROGRAMMED category. Any trap conditions and procedures previously established by the M$TRAP service are in effect at a simulated trap. Saving of the environment and transfer of control to the user's trap procedure are identical to the functions described in the discussion of the M$TRAP service.

When used to simulate the occurrence of the break, the user must first have established break control via the M$INT monitor service. The environment will be saved and the user's interrupt routine entered as described in the discussion of the M$INT service.

The form of the call for this service is:

CALL M$STRAP (FPT_STRAP) [ALTRET (label)];

The parameter for the M$STRAP service is as follows:

BRK ={YES|NO} specifies, when YES, that a break is to be simulated. Default = NO.

STRAP = OPTION specifies the trap to simulate. The options are listed below.

MEMORY
MME
FAULT_TAG
COMMAND
DERAIL
LOCKUP
IPR
OVERFLOW
DIVIDE_CHECK
MISSING_SEGMENT
MISSING_PAGE
SECURITY_1
SECURITY_2

Default = no simulated trap condition.

M$STIMER - Set Timer

INTERVAL TIMER CONTROL

The CP-6 interval timer is a software timer service provided by the monitor for each user that requests it. The monitor measures the user's execution and monitor service processing time, in millisecond units, and reports the conclusion of the specified time interval to the user through the Exceptional Condition Processing mechanism. The time measurement does not include time spent executing other users or monitor overhead procedures. The time measurement is made each time the monitor receives control from the user, after each monitor service, time slice quantum, and I/O interrupt when the user is executing. Thus the user receives control at the Exceptional Condition procedure no sooner than the requested interval and no later than the interval plus one user quantum time.

The M$STIMER and M$TTIMER services permit the user to request the interval timer service, obtain the unelapsed time within the current interval, or cancel the current interval.

M$STIMER

The M$STIMER service requests a time interval and specifies the Exception Condition processing procedure to receive control at the conclusion of the interval. Only one interval may be in progress at a time. If an earlier interval is in progress when the M$STIMER service is requested, it is forgotten.

NOTE: Special shared processors may not call the M$STIMER service.

The form of the call for this service is:

CALL M$STIMER (FPT_STIMER) [ALTRET(label)];

The parameters for M$STIMER are as follows:

UENTRY = ENTRY specifies an entry point to the user timer run-out procedure. If UENTRY is not specified, no timing will occur. The default is NIL.

UNITS = VALUE-DEC specifies the time interval. It must be specified. The maximum permissible value of UNITS is:

```
   value < 2**26 if UTYPE = TUN
   value < 2**16 if UTYPE = SEC
   value < 2**10 if UTYPE = MIN
```

Default=0.

UTYPE = OPTION specifies the kind of time that UNITS is expressed in. Valid options are:

TUN	milliseconds
SEC	seconds
MIN	minutes

The default is TUN.

M$TTIMER - Test Timer

The M$TTIMER service returns the current unelapsed time and optionally allows the current interval to be cancelled. If the current interval is cancelled, the user specified timer run-out procedure is not entered.

NOTE: Special shared processors may not call the M$TTIMER service.

The form of the call for this service is:

CALL M$TTIMER (FPT_TTIMER) [ALTRET (label)];

The parameters for this service are as follows:

CANCEL = {YES|NO} specifies, if YES, that the current running interval is to be cancelled. The default is NO.

UNITS = VARIABLE locates an 1-word area where the unelapsed time value is returned. The default is NIL.

UTYPE = OPTION specifies how the current value of the interval timer is to be expressed. Valid options are:

TUN	milliseconds
SEC	seconds
MIN	minutes

The default is TUN.

M$INT - Set Break Control

The M$INT service establishes the procedure to process an interrupt generated by depression of the BRK key at a time-sharing terminal. The user's interrupt procedure takes control immediately, if the user domain is in control when the interrupt occurs. Figure 6-2 summarizes the action taken if a domain other than the user is in control when the interrupt occurs.

When the interrupt procedure takes control, the environment is available in the Exception Condition Stack top frame. The user can use the based structure B$EXCFR to access the Exception Condition Stack frame. Exit from the interrupt procedure (to return to the point of interruption) is discussed under "Exiting Exceptional Condition Procedure".

A number of interrupts may accumulate for a given user; thus the interrupt procedure may be reentered. The user may enclose short portions of code with the DO INHIBIT...END statements which ensures up to two milliseconds free of interruption. (DO INHIBIT does not inhibit interruption during any monitor service call included in the DO group, however.)

A Control-Y sequence is simulated if the user hits four consecutive breaks. Following the Control-Y sequence, the Command Processor always takes control. A count of breaks is accumulated and can be reset to zero by calling M$TRMPRG with RSTBRK=YES. This will avoid the simulation of the Control-Y sequence.

There is no user control over the Control-Y sequence; the Command Processor will always be given control. However, the user may request that control be given to its event procedure prior to continuing normal program execution following the "GO" command to the Command Processor. (Refer to M$EVENT in this Section.)

The form of the call for this service is:

```
CALL M$INT (FPT_INT) [ALTRET (label)];
```

The parameter for this service is as follows:

UENTRY = ENTRY locates the procedure that control is transferred to when the break key is depressed (or 'B:' is entered from console if activity to be interrupted is running as a console ghost). A zero address or NIL resets a previous request for interrupt control. The default is NIL.

Domain in Control	Break Control				Debug	Action Taken
	CP	Debug	ASL	USER	Control	
CP	YES	–	–	–	–	Enter CP M$INT handler.
CP	NO	–	–	–	–	Interrupt ignored.
Debug	–	YES	–	–	–	Enter Debugger M$INT handler.
Debug	–	NO	–	–	–	Interrupt ignored.
ASL	–	–	YES	–	–	Enter ASL M$INT handler.
ASL	–	–	NO	–	YES	Enter Debug at start address.
ASL	–	–	NO	YES	NO	Defer interrupt until ASL relinquishes control.
ASL	–	–	NO	NO	NO	Enter CP at start address.
USER	–	–	–	–	YES	Enter Debug at start address.
USER	–	–	–	YES	NO	Enter User M$INT handler.
USER	–	–	–	NO	NO	Enter CP at start address.

Figure 6–2. Entry to M$INT Procedure

B$BRK

The B$BRK macro may be used to generate a based structure to access the top frame on the Exceptional Condition Stack from within a break control procedure. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$BRK.BRKCNT is the number of terminal breaks received since the last read.

B$BRK.ERR is the 36 bit error code. The error code contains the following:

B$BRK.ERR.ERR# = VALUE-UBIN(14). Contains the error code (see file B_ERRORS_C).

B$BRK.ERR.FCG = VALUE-BIT(12). Contains the Functional Code Group portion c the error code.

B$BRK.ERR.MID = VALUE-BIT(6). Contains the Module ID portion of the error code.

B$BRK.ERR.MON = VALUE-BIT(1). Indicates that the error code was generated by the monitor.

B$BRK.ERR.SEV = VALUE-UBIN(3). Contains the severity of the error.

B$BRK.P# = VALUE-UBIN(36). Contains the number of words of parameters that follow. For B$BRK, this field is always zero.

B$BRK.SSFRAME contains the copy of the program's environment at the time of the interrupt condition. The based structure B$EXCFR should be used to access the environment.

M$EVENT - Set Event Control

The M$EVENT service works in conjunction with the M$READ, M$WRITE, M$ENQ, and other services with the no-wait option specified (WAIT=NO). The user calls M$EVENT to establish the entry-address of an event processing procedure that receives control when an I/O completion occurs (for M$READ or M$WRITE), when a resource becomes available (as a result of M$ENQ), or when the user has been interrupted by the Command Processor (see the YC option). The call to the M$READ, M$WRITE, or M$ENQ service may optionally specify an EVENT value. This value is passed to the event processing procedure to identify the event that occurred. If an EVENT value is not specified or is zero for a non-wait M$READ or M$WRITE operation, the event procedure is not entered on read or write completion.

The asynchronous event reporting facility for no-wait IO completion is provided by either the M$EVENT monitor service or the M$CHECK monitor service discussed in Section 3. These services allow the user to call a monitor service, continue processing while the monitor service is being performed, and be notified of its completion.

Entry to the event procedure occurs at one of these points: following a monitor service, at an I/O interrupt, or at completion of a time slice. If a program has been suspended by the M$WAIT service when an event completion occurs, the suspension is terminated and the event completion is reported.

The user may request as many concurrent no-wait operations as desired. The order in which events are processed is not necessarily the order in which they were requested. The event procedure processes events in the order of event completion.

The form of the call for this service is:

CALL M$EVENT (FPT_EVENT) [ALTRET (label)];

The parameter for this service is as follows:

UENTRY = {ENTRY|NIL} specifies the event procedure entry-address. An address of zero or NIL resets a previous request and specifies that event completions of no-wait services are not to be reported.

YC = {YES|NO}. Specifies, if YES, that the procedure specified via the UENTRY option is also to be entered following a Control-Y then GO sequence from the user terminal. Default = NO.

Environment for M$EVENT

Refer to Appendix A for the layout of the top frame in the Exceptional Condition Stack upon entry to the event procedure. The B$EXCFR based structure is available to access the common portion of the frame. The value in B$EXCFR.SUBC uniquely identifies the type of event. Values (from the %SUB_EXC macro) are as follows:

Sub-code	Value	Meaning
%SUBC_ENQ#	0	Enqueue Event Completion
%SUBC_IO#	1	No-wait I/O Completion
%SUBC_COOP#	2	Reserved by the Monitor
%SUBC_JRNLERR#	3	Journal requires error handling by owner
%SUBC_YC#	4	Continuing from Command Command Processor (YC-GO)
%SUBC_FPRG#	5	A FEP user program has just either exited, aborted or issued M$LDTRC
%SUBC_COMIO#	6	Communications IO event

The BENQ, BNWIO, B$JRNLERR, B$FPRG and B$COMIO based structures are available to access information pertinent to enqueueing, no-wait I/O operations, journal errors and FEP user program state changes.

Reentry Considerations

Once entered, the user's event procedure may be re-entered to report subsequent event completions. For example, a second outstanding no-wait I/O may complete while the user is asynchronously processing a prior no-wait I/O completion event. Also, the M$EVENT routine could be interrupted for entry to the M$STIMER specified routine if the specified time expires while in event processing. The DO INHIBIT statement may be used to protect against reentry to the event procedure, if necessary. The DO INHIBIT statement inhibits interrupts and timer run-out faults within a section of the program. The DO INHIBIT statement inhibits for up to two milliseconds, but after that time a lockup fault will occur. Within the code protected by DO INHIBIT, monitor service calls are not inhibited; thus event reporting can take place in the course of their execution.

Inhibiting event reporting via the DO INHIBIT...END statements can be useful under various conditions. Two sample cases are discussed below.

1. A program tests a memory location to determine if an event has been reported and, if it has not been reported, calls M$WAIT to suspend the program until the event completes. If DO INHIBIT is not used prior to the test of the memory location, the event could be reported and the flag could be set between the test and the call to M$WAIT; in this case M$WAIT would suspend for the full wait time. If the DO INHIBIT statement is used just prior to the test, event reporting is delayed until the M$WAIT service is called and the program is not suspended.

2. A program contains a data structure in which data modified by the event completion procedure and data modified by the main program are contained in the same memory word. If the main program is interrupted while manipulating the data and the event procedure takes control and modifies its data, these modifications may be lost when control is returned to the main program and execution of the interrupted statement causes main program data to be restored to memory. Placing DO INHIBIT...END statements in the main program can protect against such problems.

Event Completion Processing and Domains

The monitor no-wait services and control via the M$EVENT YC option are available to each of the domains: user, Alternate Shared Library, Debugger, and Command Processor.

It is not necessary for the domain that called the no-wait service to be in control when the service completes. If a special shared processor calls the no-wait service and then returns control to the user, the user is interrupted and the processor's event procedure is entered when the no-wait operation completes. If the user calls the no-wait service and then enters a special shared processor, entry to the user's event procedure is delayed until control returns to the user .

B$NWIO

The %B$NWIO macro may be used to generate a based structure of a frame placed on the TCB stack for a No-wait I/O event. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$NWIO.ARS = VALUE-UBIN(36). Contains the actual record size. See M$DCB.ARS.

B$NWIO.DVE.DVBYTE is returned on reads and contains information about the nature of the item read. See M$DCB. This field contains the following sub-fields:

B$NWIO.DVE.DVBYTE.BIN = VALUE-BIT(1).

B$NWIO.DVE.DVBYTE.TRANS = VALUE-BIT(1).

B$NWIO.DVE.DVBYTE.VFC = VALUE-BIT(1).

B$NWIO.DVE.EOMCHAR = CHAR(1). Contains the activation character of reads.

B$NWIO.ERR is the 36 bit error code, containing the following:

B$NWIO.ERR.ERR# = VALUE-UBIN(14). Contains the error code (see file B_ERRORS_C).

B$NWIO.ERR.FCG = VALUE-BIT(12). Contains the Functional Code Group portion of the error code.

B$NWIO.ERR.MID = VALUE-BIT(6). Contains the Module ID portion of the error code.

B$NWIO.ERR.MON = VALUE-BIT(1). Set if the error was generated by the monitor.

B$NWIO.ERR.SEV = VALUE-UBIN(3). Contains the severity of the error.

B$NWIO.EVID = VALUE-UBIN(36). Contains the event ID as was specified on the M$READ or M$WRITE Service Request.

B$NWIO.IOERRCODE is the error code that occurred on the I/O. This is only valid on I/O connected directly to the FEP.

B$NWIO.P# = VALUE-UBIN(36). Contains the number of words of parameters that follow.

B$NWIO.SSFRAME contains the copy of the program's environment at the time of the event. The based structure B$EXCFR should be used to access this area.

B$NWIO.SUBC = VALUE-UBIN(18). Contains the event sub-code. For No-wait I/O completion this field will be set to %SUBC_IO#.

B$NWIO.TYC = VALUE-BIT(36). Contains the type of completion. See M$DCB.TYC.

B$NWIO.VLP_STATION is described under VLP_STATION.

B$COMIO

The B$COMIO macro may be used to generate a based structure of a frame placed on the TCB stack for a communications I/O event. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$COMIO.ARS = VALUE-UBIN(36). Contains the actual record size. This is the size of the record available for a %SUBC2_DATA_AVL# event or the size of the read buffer for a %SUBC2_DATA_RQS# event.

This word is not present if SUBC2 is set to %SUBC2_CLS_DCB#.

B$COMIO.EVID = VALUE-UBIN(36). Contains the event ID as was specified on the M$OPEN Service Request for this DCB.

B$COMIO.P# = VALUE-UBIN(36). Contains the number of words of parameters that follow.

B$COMIO.SSFRAME contains the copy of the program's environment at the time of the event. The based structure B$EXCFR should be used to access this area.

B$COMIO.SUBC = VALUE-UBIN(18). Contains the event sub-code. For communications I/O events, this field will be set to %SUBC_COMIO#.

B$COMIO.SUBC2 = VALUE-UBIN(9). Contains the type of communications I/O event. It is either %SUBC2_DATA_RQS# or %SUBC2_DATA_AVL# if this is a user event.

%SUBC2_DATA_RQS# specifies that a read is being performed by the process attached to the specified DCB.

%SUBC2_DATA_AVL# specifies that a write has been performed by the process attached to the specified DCB and that the specified DCB is not currently reading.

If the event is for a debugger, SUBC2 could also be set to %SUBC2_CLS_DCB# indicating the user has closed his FPRG DCB.

FLAGS.DELTA = VALUE-BIT(1) specifies, if set, that this B$COMIO event is intended for the debugger only; users will never see this flag set.

B$ENQ

The %B$ENQ macro may be used to generate a based structure of a frame placed on the TCB for an M$ENQ event. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$ENQ.ERR is the 36 bit error code, containing the following:

B$ENQ.ERR.ERR# = VALUE-UBIN(14). Contains the error code (see file B_ERRORS_C).

B$ENQ.ERR.FCG = VALUE-BIT(12). Contains the Functional Code Group portion of the error code.

B$ENQ.ERR.MID = VALUE-BIT(6). Contains the Module ID portion of the error code.

B$ENQ.ERR.MON = VALUE-BIT(1). Set if the error was generated by the monitor.

B$ENQ.ERR.SEV = VALUE-UBIN(3). Contains the severity of the error.

B$ENQ.EVID = VALUE-UBIN(36). Contains the event ID as was specified on the M$ENQ Service Request.

B$ENQ.MESSAGE = VALUE-BIT(36). Contains the message from the user who last issued a DEQ for the resource.

B$ENQ.P# = VALUE-UBIN(36). Contains the count of the number of words of pa.ameters that follow. For B$ENQ this field is always 1.

B$ENQ.SSFRAME contains the copy of the program's environment at the time of the event. The based structure B$EXCFR should be used to access this area.

B$ENQ.SUBC = VALUE-UBIN(18). Contains the event sub-code. For enqueue event completion this field will be set to %SUBC_ENQ#.

B$JRNLERR

The %B$JRNLERR macro may be used to generate a based structure of a frame placed on the TCB stack for a journal error event. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$FIXED.DCB# = VALUE-UBIN(36). For ALTRET frames, contains the number of the DCB on which the error occurred.

B$JRNLERR.ERR is the 36 bit error code, containing the following:

B$JRNLERR.ERR.ERR# = VALUE-UBIN(14). Contains the error code (see file B_ERRORS_C).

B$JRNLERR.ERR.FCG = VALUE-BIT(12). Contains the Functional Code Group portion of the error code.

B$JRNLERR.ERR.MID = VALUE-BIT(6). Contains the Module ID portion of the error code.

B$JRNLERR.ERR.MON = VALUE-BIT(1). Set if the error was generated by the monitor.

B$JRNLERR.ERR.SEV = VALUE-UBIN(3). Contains the severity of the error.

B$JRNLERR.P# = VALUE-UBIN(36). Contains the number of words of parameters that follow.

B$JRNLERR.SSFRAME contains the copy of the program's environment at the time of the event. The based structure B$EXCFR should be used to access this area.

B$JRNLERR.SUBC = VALUE-UBIN(18). Contains the event sub-code. For journal errors this field will be set to %SUBC_JRNLERR#.

B$JRNLERR.VOL = VALUE-UBIN(18). Contains the index into the JRNL owner's serial number table of the volume on which the error occurred.

B$FPRG

The B$FPRG macro may be used to generate a based structure of a frame placed on the TCB stack for an FPRG state-change event. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$EVNT.B$FIXED contains the fixed initial portion of the event frame. See B$FIXED for details.

B$EVNT.P# = VALUE-UBIN(36). Contains the count of the number of parameters that follow.

B$FPRG.DCBNO = VALUE-UBIN(18). Contains the DCB number for which this FPRG event occurred.

B$FPRG.ERR is the 36 bit error code meaningful only if B$FPRG.SUBC2 = %SUBC2_ABORT#. It contains the following:

B$FPRG.ERR.ERR# = VALUE-UBIN(14). Contains the error code (see file B_ERRORS_C).

B$FPRG.ERR.FCG = VALUE-BIT(12). Contains the Functional Code Group portion of the error code.

B$FPRG.ERR.MID = VALUE-BIT(6). Contains the Module ID portion of the error code.

B$FPRG.ERR.MON = VALUE-BIT(1). Set if the error was generated by the monitor.

B$FPRG.ERR.SEV = VALUE-UBIN(3). Contains the severity of the error.

B$FPRG.EVID = VALUE-UBIN(36). Contains the event ID as was specified on the M$OPEN Service Request for this DCB.

B$FPRG.FPRG contains the description of the FPRG just loaded via an M$LDTRC. This section is valid only for B$FPRG.SUBC2 = %SUBC2_FPLDTRC#. See VLP_FPRG.

B$FPRG.P# = VALUE-UBIN(36). Contains the number of words of parameters that follow.

B$FPRG.SSFRAME contains the copy of the program's environment at the time of the event. The based structure B$EXCFR should be used to access this area.

B$FPRG.SUBC = VALUE-UBIN(18). Contains the event sub-code. For FPRG state changes this field will be set to %SUBC_FPRG#.

B$FPRG.SUBC2 = VALUE-UBIN(9). Contains the reason for this FPRG state change event. It is either %SUBC2_FPEXIT#, %SUBC2_FPABORT# or %SUBC2_FPLDTRC#.

M$XCON - Set Exit Control

EXIT CONTROL

Conditions causing a program exit are listed below. The conditions are classified by severity of impact, as described later.

Class	Condition
1	Normal exit at M$EXIT
1	Abnormal exit at M$ERR or M$XXX
1	Machine trap and no user trap control requested
1	Monitor service error not processed by ALTRET code sequence or M$TRAP (PMME) procedure
1	Transfer of control to another load module (M$LDTRC)
1	Save program image (M$SAVE or Control-Y SAVE)
1	Restore program image (GET)
1	Exceptional condition occurs but procedure cannot be entered because the stack is full
1	Operator "errored the user"
2	Resources limit exceeded
3	Operator abort
3	Line disconnect

If the user requests exit control, control is transferred to the user at any of these conditions. The exit procedure may perform various clean-up funtions and also process unexpected conditions. The reason for the exit is passed to the specified procedures. Limits on output and time are reestablished to control the exit procedure.

M$XCON

The M$XCON service requests exit control and specifies the entry-address for the exit procedure. When this procedure takes control, the program is in "exit" status; a flag in the ECCB is set to reflect "exit control in progress".

The user can also issue the M$XCON service for these purposes: nesting exit entries and removing the program from exit status. These two uses are discussed below.

1. The user can control nesting of exit entries. At the initial call to M$XCON, an exit entry is established. To supply a different exit entry the user reissues M$XCON specifying the entry and the PXCON parameter which saves the previous exit entry so that it may be used later to reestablish exit control to its previous entry point. Thus each level of nested exit procedures retains the entry-address of the next higher level. The currently effective exit control procedure can issue the M$XCON service to reestablish a previous exit entry.

2. Executing the M$XCON service within an exit procedure removes the program from "exit" status.

The form of the call for this service is:

CALL M$XCON (FPT_XCON) [ALTRET (label)];

The parameters for the M$XCON service are as follows:

PXCON = VARIABLE locates a 1-word area where an EPTR to the previously set exit procedure address is to be returned. Default = NIL.

UENTRY = {ENTRY|NIL} locates the procedure to be entered upon exit of the current program whether normal or abnormal. A zero address or NIL reverses the user's previous request for exit control. Default = NIL.

Entry to Exit Control Procedure

The user should be aware of several conditions that exist on entry to the exit procedures:

- o The exit procedure is entered in Slave Mode.
- o Previously established timer and break controls are deferred. (These controls are reimposed if the program "exit" status is reversed.)
- o Standard system exit control limit increments are established by the system manager at system definition. Table 6-2 summarizes the effect of various classes of exit events on the user's current limits. The user's current limits are incremented by exit control limits, if a class 2 or a class 3 exit occurs. An exception is made for the time limit which is not treated as an increment. If maximum time must be set (see Table 6-2), the exit control default is stored as the new maximum run-time limit. The system is protected from looping in an exit procedure because some limit will eventually be exceeded.

Another exit condition may occur while the user's exit control procedure has control. Table 6-2 also explains what occurs in this case and describes the effect on the exit procedure already in progress.

Table 6-2. Entry to Exit Control

Condition	Monitor Action at Initial Exit	Monitor Action While Processing Previous Exit
Class 1		
Normal exit from user program (M$EXIT) Abnormal exit from user program (M$ERR or M$XXX) User program trap Monitor service error not handled by ALTRET or M$TRAP (PMME) Operator errored the user Load and transfer to another program (M$LDTRC) SAVE, M$SAVE, or GET if exit control requested Normal or abnormal exit from Special Shared Processor	User's current limits are not modified. No time limit is imposed.	While processing Class 1, 2, or 3: If M$EXIT, M$ERR, or M$XXX, then run-down the job step. In all other cases, enter the currently effective exit procedure without establishing processing limits.
Resource limit exceeded	User's current limits are incremented by a fixed amount specified at system definition. Time limit is imposed for batch jobs, but not for online jobs.	While processing Class 1, enter the currently effective exit procedure and take action described for Class 2 initial entry. While processing Class 2, batch users are logged off; online user's Command Processsor takes control (the user exhausted the extended processing capabilities granted by the system). While processing Class 3, both batch and online users are logged off.

Table 6-2. Entry to Exit Control (cont.)

Class 3		
Operator abort Line disconnect Mon/CP cancel	User's current limits are incremented by a fixed amount (specified at system definition). Time limit is imposed for both batch and online jobs.	While processing Class 1, enter the currently effective exit procedure and take action described for initial Class 3 entry. While processing Class 2, user's current limits are not modified. Time limit is set for batch and online jobs. While processing Class 3, both batch and online users are logged off.

On entry to the Exit Control procedure, the environment is placed on the Exceptional Condition Stack (see Figure 6-1). The BASED structure, B$XCON, can be used to access the additional words of information stored following the environment.

If there is insufficient room on the Exceptional Condition stack for the exceptional condition environment, as indicated by the the XCONF.NOTCB bit set in the ECCB, the exit control environment will be in the ALTRET frame with the ECC set to indicate that the TCB is full. The format of the frame is the same as the exit control frame that is placed on the stack with the exception of the value of the ECC. The error code in the exit control frame is not replaced if the exit condition is abnormal. If the program is being aborted because of a stack full condition when a break, timer interval expired or event completion occurs, the error code is set to indicate that the TCB is full. The user should always inspect the ECCB upon entry to the exit control procedure to determine if this condition exists. When this condition occurs, the exit control information must be accessed via the ALTRET frame pointer in the TCB. For example, on entry to an exit control procedure:

```
IF B$ECCB$->B$ECCB.XCONF.NOTCB
THEN MYPTR$ = B$TCB$->B$TCB.ALT$;
ELSE MYPTR$ = B$TCB$->B$TCB.STK$;
```

B$XCON

The %B$XCON macro may be used to generate a based structure defining an exit control frame. The format of this structure can be found in Appendix A. The sub-fields in this structure have the following meanings:

B$XCON.CECCB = VALUE-BIT(9). The bit settings in B$XCON.CECCB reflect the ECCB.XCONF bits that have been set on this entry to exit control. For example, if a program simply issues an M$EXIT, B$XCON.CECCB.PROG will be set and all other bits in B$XCON.CECCB will be reset. If the program then exceeds the time limit while in exit control processing, a second exit control frame will be pushed on the TCB stack and the program will be entered at its exit control address. The top stack frame will now have B$XCON.CEECB.LIMIT set. B$XCON.ECCB.PROG will not be set in this frame indicating that exit control was already in progress when this exit condition occurred.

Level 3 Name	Bit	Meaning
PROG	0	Exit control was not in progress when the exit condition occurred.
NOTCB	1	The exit condition occurred when the Exceptional Condition Stack was full.
LIMIT	2	Limits have been incremented, i.e., Class 2 exit control in progress.
CL3	3	Class 3 exit control in progress.
SGET	4	Exit control for simulated GET. (GET will also be set.)
LNK	5	Exit control for M$LDTRC.
SAVE	6	Exit control for M$SAVE.
CPSAVE	7	Exit control Control-Y SAVE.
GET	8	Exit control for GET.

B$XCON.ECCB = VALUE-BIT(9). The bit settings in B$XCON.ECCB reflect the exit control conditions that existed prior to this entry to exit control. For example, a program that has entered exit control processing for M$LDTRC and is then aborted by the operator will be re-entered at its exit control address with B$XCON.CECCB.CL3 set, and B$XCON.CECCB.LNK reset. B$XCON.ECCB.PROG and B$XCON.ECCB.LNK would both be set.

Level 3 Name	Bit	Meaning
PROG	0	Exit control was previously in progress when the exit condition occurred.
NOTCB	1	The exit condition occurred when the Exceptional Condition Stack was full.
LIMIT	2	Limits have been incremented, i.e., Class 2 exit control in progress.
CL3	3	Class 3 exit control in progress.
	4	Unused.
LNK	5	Exit control for M$LDTRC.
SAVE	6	Exit control for M$SAVE.
CPSAVE	7	Exit control Control-Y SAVE.
GET	8	Exit control for GET.

B$XCON.ECSC = VALUE-UBIN(18). The bit settings for B$XCON.ECSC reflect the exit conditions as they apply to domains other than the currently active domain.

Sub-code	Meaning
0	The exit condition occurred while the currently executing domain was in control. This will always be zero for a user level program. If a user program exit control procedure is entered following either a normal or abnormal exit from a special shared processor, the special shared processor abort bit is set in the current run status (B$XCON.RNST.SSABRT). The environment in the exit control frame is that which existed in the user program at the last call to the Alternate Shared Library or upon entry to the debugger or to the Command Processor.
1	Exit control occurred while a lower priority domain was in control. M$TRTN is not allowed. This code is meaningful only to special shared processors.

B$XCON.ERR is the 36 bit error code. The error code contains the following:

B$XCON.ERR.ERR# = VALUE-UBIN(14). Contains the error code (see file B_ERRORS_C).

B$XCON.ERR.FCG = VALUE-BIT(12). Contains the Functional Code Group portion of the error code.

B$XCON.ERR.MID = VALUE-BIT(6). Contains the Module ID portion of the error code.

B$XCON.ERR.MON = VALUE-BIT(1). Indicates that the error code was generated by the monitor.

B$XCON.ERR.SEV = VALUE-UBIN(3). Contains the severity of the error.

B$XCON.LIMIT = VALUE-BIT(9). The bit settings for B$XCON.LIMIT have the following meaning:

Level 3 Name	Bit	Meaning
PO	0	Punched cards.
MEM	1	Memory (M$LINK return).
LO	2	User pages(LO).
DGO	3	Diagnostic pages.
PDISK	4	Permanent disk granules.
TDISK	5	Temporary disk granules.
TAPE	6	Scratch Tapes.
STACK	7	Safe-Store Stack limit has been exceeded.
TIME	8	Time Limit Exceeded.

B$XCON.P# = VALUE-UBIN(36). Contains the number of words following the standard B$XCON Exceptional Condition frame. This will be zero unless the exit condition is for SAVE or LDTRC.

B$XCON.RNST = VALUE-BIT(9). The bit settings for B$XCON.RNST reflect the current Run Status and indicate the reason for this entry to exit control:

Level 3 Name	Bit	Meaning
OPXXX	0	Operator aborted the job.
HANGUP	1	Line disconnect.
LIMIT	2	Limit exceeded.
MONXXX	3	Monitor or Command Processor is logging off the user.
OPERR	4	Operator errored the job or user errored the job via the Control-Y QUIT sequence.
MABRT	5	Program aborted by the monitor.
SSPABRT	6	Program aborted by a special shared processor.
XXX	7	M$XXX was issued.
ERR	8	M$ERR was issued.

This field is set to zero if M$EXIT, M$LDTRC, M$SAVE or Control-Y SAVE or GET caused the exit. The bit settings in B$XCON.CECCB differentiate between these conditions. In all other cases the exit was due to an error condition.

B$XCON.XCONF = VALUE-BIT(9). The bit settings in B$XCON.XCONF have meaning on multiple entries to exit control either at the same level or for multiple levels of exit control.

Level 3 Name	Bit	Meaning
	0-5	These bits reflect the previous Run Status. For example, if a program that is in exit control for limits exceeded were to be aborted by the operator, that program's exit control procedure would be re-entered with B$XCON.RNST.OPXXX set and B$XCON.XCONF.LIMIT also set.
OPXXX	0	Operator aborted the job.
HANGUP	1	Line disconnect.
LIMIT	2	Limit exceeded.
MONXXX	3	Monitor or Command Processor is logging off the user.
OPERR	4	Operator errored the job or user errored the job via the Control-Y QUIT sequence.
MABRT	5	Program aborted by the monitor.

	6-8	These bits reflect the exit condition of another domain
URND	6	The user level exit has completed. This code is meaningful only to special shared processors. ECC Subcode will be set to 1.
ASL	7	The ASL level exit has completed and the ASL has been run down. This code is meaningful only to the user level.
QUIT	8	Control-Y QUIT sequence (operator abort) initiated the abort process.

Exit from Exit Control Procedure

To resume normal execution of a program after receiving exit control, the user can call the M$TRTN service. This service removes the exit control frame from the TCB stack and removes the program from the "exit" status if this call to M$TRTN is not from within a nested exit control procedure; i.e. the 'PROG' bit is set in B$XCON.CECCB.

Note that the M$TRTN service cannot be used by a special shared processor if a lower level domain was in control when the exit condition occurred, as indicated by the ECC subcode.

The user can also issue the M$XCON service within the exit procedure to remove the program from the "exit" status. The user may call this service with a previously acquired PXCON parameter specified as the exit control entry address to reestablish a previous exit control entry. If this is followed by an M$EXIT, M$ERR or M$XXX, that exit control entry will then be entered.

To unconditionally exit the current job step, the user can call the M$EXIT, M$ERR, or M$XXX service. The user may communicate with other job steps in the same job by use of the Step Condition Code, which may be set via a parameter of these services.

Please note that a program is removed from the "exit" status any time an exit control frame is removed from the Exceptional Condition Stack and the frame does not indicate a nested exit control entry; again, when the 'PROG' bit in B$XCON.CECCB is set. This may be as a result of an M$CLRSTK service request or an M$TRTN service request.

Exit Control and Domains

At an exit condition, control passes to the exit control procedure of the domain currently in control. The CP-6 system permits a separate level of exit control for the user domain, the Alternate Shared Library (ASL), the Debugger and the Command Processor. Each level of control is distinct from the other; there may be nesting of exit procedures for each domain. The system maintains separate limits for each domain.

If both the user and ASL request exit control and the ASL is in control when the exit occurs, its exit procedure is entered first. If the Alternate Shared Library can recover from the exit condition, it can call the M$TRTN service to continue normal execution. If the ASL cannot continue normal execution, it can defer exit control processing until the user's exit control has been completed (see M$XCONRTN), or it can close files and issue the M$XXX service which causes the monitor to disassociate the ASL and to enter the user's exit procedure. In this case, the special shared processor abort bit is set (in B$XCON.RNST.SSPABRT). The environment in the Exceptional Condition Stack is the environment that existed in the user program at the last call to the Alternate Shared Library. After the ASL is disassociated, no further calls to the ASL are allowed.

If the user is in control at the exit condition, the user's exit procedure takes control immediately. If the user continues the abort process by issuing M$XXX, the ASL exit procedure is entered, allowing the ASL to clean up its files. In this case the environment on the Safe-Store Stack is the initial ASL environment. The ASL must exit its exit control procedure by issuing the M$EXIT, M$ERR, or M$XXX service; a M$TRTN request or a return to the user causes reentry to the ASL exit procedure.

If DELTA is associated with the user, and the user is in control when the exit condition occurs, DELTA is entered prior to entering the exit control procedure for the user or ASL. The "XCON" command to DELTA then causes the user's exit control procedure to be entered. When the user issues an M$EXIT, M$ERR or M$XXX service request in his exit control procedure, control passes to the Special Shared Processors that have requested exit control from lowest to highest priority.

Exit Control and M$LDTRC or Save

If exit control is requested in a program that calls another load module, the M$LDTRC service passes control to the effective exit procedure. An M$EXIT or M$TRTN request when the top stack frame is the M$LDTRC exit control frame will then cause the M$LDTRC to occur.

The SAVE process is similar when the user has asked for exit control on SAVEs via the M$SCON monitor service. The SAVE process takes place when the user issues an M$EXIT or M$TRTN and the top TCB stack frame is the SAVE exit control frame.

M$SENV and M$RENV - Save/Restore ALTRET Environment

SAVE/RESTORE ALTRET ENVIRONMENT

The services to save and restore the environment are typically used in conjunction with an ALTRET code sequence. Saving the ALTRET environment also permits esoteric manipulations. For example, using M$SENV/M$RENV allows recursion into the ALTRET code sequence. In another case, using M$SENV/M$RENV protects the ALTRET environment of the main procedure of a program if any ASYNC procedure also issues monitor service requests. In addition, using the M$SENV service ensures the integrity of the ALTRET environment in case of an "insufficient space in TCB stack" condition which can occur if an exceptional condition is detected while the ALTRET code sequence has control.

M$SENV

The M$SENV service allocates a frame in the Exceptional Condition Stack and moves the environment to the frame from the TCB ALTRET frame.

The M$SENV service takes the alternate return if there is no environment in the ALTRET frame or if there is insufficient space to store the environment in the Exceptional Condition Stack.

The form of the call for this service is:

```
CALL M$SENV [ALTRET (label)];
```

Because no parameters are associated with the monitor service call, there is no FPT.

M$RENV

The M$RENV service removes the top frame of the exceptional condition stack and restores the environment to the ALTRET frame in the TCB.

The M$RENV service takes the alternate return if the exceptional condition stack is empty or if the top frame of the stack is not an ALTRET frame.

The form of the call for this service is:

CALL M$RENV [ALTRET (label)];

Because no parameters are associated with the monitor service call, there is no FPT.

Exiting Exceptional Condition Procedure

The user's exceptional condition procedure or ALTRET code sequence determines whether to exit, to abort, to pass control to the monitor for error processing, or to resume normal program execution. The services described here allow the user to transfer control, to manipulate the TCB Exception Condition Stack, and to restore a prior environment to the Safe-Store Stack. These services are categorized below by the type of procedure in which they may appear.

Routine Type	Monitor Services to Exit the Routine
ALTRET code sequence	M$MERC M$MERCS (a) M$RETRY M$RETRYS (a)
Exceptional Condition Procedures	M$TRTN M$CLRSTK M$MERCS M$RETRYS (b)

a This service is appropriate if the original monitor service environment is in the top frame of the Exceptional Condition Stack.

b This service may be used to exit the PMME procedure specified on the M$TRAP service.

PL-6 CONSIDERATIONS

In a PL-6 program, the M$TRAP, M$STIMER, M$EVENT, M$INT, and M$XCON services establish the entry (ENTADDR) to procedures to be entered asynchronously when exceptional conditions occur. These procedures should be coded as external ASYNC procedures.

Exit from these procedures can take advantage of the PL-6 RETURN and ALTRETURN statements. In particular, exit from a PL-6 ASYNC procedure can occur as follows:

- o RETURN - Generates an M$TRTN monitor service request (with no FPT) which returns to the point of interruption with the original environment restored to the Safe-Store Stack.
- o ALTRETURN - Generates an M$MERCS monitor service request which gives control to the monitor for error processing.
- o UNWIND - Restores the AUTO pointer to the REMEMBERed value and transfers control to the REMEMBERed label. UNWIND should normally be preceded by M$CLRSTK to release the top frame on the Exceptional Condition Stack.

If it is necessary to execute an M$TRTN or M$MERCS procedure specifying an FPT, this must be done by calling X66_TRTN or X66_MERCS respectively, passing the FPT to be used. Automatic storage will be properly maintained followed by the execution of the desired monitor service request. In no case should M$TRTN or M$MERCS monitor service, with or without an FPT, be requested from an PL6 procedure.

M$MERC and M$MERCS - Monitor Error Cntl

The M$MERC service returns control from the user ALTRET code sequence to the monitor. This service is provided in case the ALTRET procedure cannot process all possible error codes. The original environment from the error on a monitor service call is assumed to be in the ALTRET frame of the TCB at the call to the M$MERC service.

The monitor responds by taking appropriate action based on the severity of the error:

Severity Level	Monitor Action
0	Ignore the error. Return to the statement following the monitor service call, with the Safe-Store Stack reflecting the original environment. If the MERC is from exit control the monitor performs M$EXIT.
1-3	Abort the job step. (The monitor performs M$ERR.)
>3	Abort the job. (The monitor performs M$XXX.)

The STEPCC will be set to the value of the severity of the error code. The M$MERC service takes the alternate return if there is no environment in the ALTRET frame of the TCB. No FPT is required by M$MERC but one may be specified to provide for explicit setting of the severity of the error.

The form of the call for this service is:

```
CALL M$MERC [ALTRET(label)];
```

or

```
CALL M$MERC(FPT_MERC) [ALTRET(label)];
```

The M$MERCS service performs the same function as the M$MERC service, except that the environment is assumed to be in the top frame of the Exceptional Condition Stack. (The M$MERCS service releases this top frame.) Thus this service is appropriate for use in an ALTRET code sequence that previously executed M$SENV, in the PMME procedure specified on the M$TRAP service, or in an exit control procedure that receives control because the program provides no ALTRET procedure or M$TRAP procedure.

The M$MERCS service takes the alternate return if the Exceptional Condition Stack frame is empty. No FPT is required by M$MERCS but one may be specified to provide for explicit setting of the severity of the error. Generally, a PL-6 programmer will not use M$MERCS directly. Please refer to the comments about PL-6 considerations under 'Exiting Exceptional Condition Procedures' in this section.

The form of the call for this service is:

```
CALL M$MERCS [ALTRET(label)];
```

or

```
CALL M$MERCS(FPT_MERC) [ALTRET(label)];
```

Note that the same FPT is used for both the M$MERC and M$MERCS services. Th^ parameter is also identical:

JERR=VARIABLE Locates the word that is to be used to replace the error code. When specified the value of the Severity will be used by the monitor to determine how to proceed.

M$RETRY and M$RETRYS - Retry Service

The M$RETRY service returns control from the user ALTRET code sequence to the original monitor service call and retries the service call. The environment for the original monitor service request is assumed to be in the ALTRET frame of the TCB. The M$RETRY service takes the alternate return if there is no environment in the ALTRET frame.

Before retrying the call, the user can alter the environment in the ALTRET frame. If no changes are to be made to the environment, the form of the call for this service is as follows:

CALL M$RETRY [ALTRET (label)];

If changes are to be made to the environment, the form of the call is as follows:

CALL M$RETRY (FPT_RETRY) [ALTRET (label)];

The M$RETRYS service retries a monitor service call. Its function is the same as the M$RETRY service except that the environment of the original monitor service call is assumed to be in the top frame of the Exceptional Condition Stack. The M$RETRYS service removes this frame from the stack and restores it to the Safe-Store Stack. The M$RETRYS service takes the alternate return if the Exceptional Condition Stack is empty or if the top frame contains an exceptional condition code (ECC) other than %ECC_ALT# or %ECC_PMME#.

If no changes are to be made to the environment prior to retrying the original monitor service call, the form of the call for this service is as follows:

CALL M$RETRYS [ALTRET (label)];

If changes are to be made to the environment, the form of the call is as follows:

CALL M$RETRYS (FPT_RETRY)[ALTRET (label)];

Note that the same FPT is used for both the M$RETRY and M$RETRYS services. The parameters are also identical:

DRS = VARIABLE locates an 8-word area that contains the pointers that are used to load the Pointer Registers. The SID section of each pointer must reference the Linkage Segment; if not, the alternate return is taken. The structure for the area may be generated by the VLP_PTRS macro. Default = NIL.

REGS = VARIABLE locates an 8-word area that contains the values to replace registers X0-X7, A, Q, and E. The structure of this area may be generated by the VLP_REGS macro. Default = NIL.

M$TRTN - TCB Return

The M$TRTN service returns control to the program from an exceptional condition procedure. The service removes the top frame of the Exceptional Condition Stack and restores that environment to the Safe-Store Stack. The user may specify changes to be made to the environment, including the Instruction Counter. The M$TRTN service transfers control to the user program at the statement following the point of interruption or to the location specified by a change to the IC.

Note that if the M$TRTN service request is issued when the top frame on the Exceptional Condition Stack is an exit control frame, and that frame does not indicate a nested exit control entry, the program is removed from the "exit" status. Please refer to 'EXIT FROM EXIT CONTROL' in this section.

Generally, a PL-6 programmer will not use M$TRTN directly. Please refer to the comments about PL-6 considerations under 'Exiting Exceptional Condition Procedures' in this section.

The M$TRTN service takes the alternate return if there is no environment in the Exceptional Condition Stack.

If there are no changes to the environment, the form of the call for the M$TRTN service is as follows:

CALL M$TRTN [ALTRET (label)];

If there are changes to be made to the environment, the form of the call for this service is as follows:

CALL M$TRTN (FPT_TRTN) [ALTRET (label)];

In this case, the parameters for the M$TRTN service are as follows:

DRS = VARIABLE locates an 8-word area that contains the pointers that are used to load the Pointer Registers. The SID section of each pointer must reference the Linkage Segment; if not, the alternate return is taken. The structure for the area may be generated by the VLP_PTRS macro. Default = NIL.

IC = VALUE-DEC(0-n) specifies the location of a procedure to be entered following the call to M$TRTN. A value of zero indicates that the IC is not to be changed. Default = 0.

IR = VALUE-BIT(18) specifies the value to be placed in the Indicator Register when the environment is restored. Default = '000'O.

REGS = VARIABLE locates an 8-word area that contains the values to replace X0-X7, A, Q, and E. The structure of this area may be generated by the VLP_REGS macro. Default = NIL.

SETIR = {YES|NO} indicates whether or not the contents of the Indicator Register is to be replaced in the restore environment. Default = NO.

M$CLRSTK - Clear Stack Frame(s)

The M$CLRSTK service deletes one or more frames from the Exceptional Condition Stack. Following this call, execution continues in-line. If there is no environment in the Exceptional Condition Stack, the alternate return is taken.

Note that on removing an exit control frame from the Exceptional Condition Stack if that frame does not indicate a nested exit control entry, the program is removed from the "exit" status. Please refer to 'EXIT FROM EXIT CONTROL' in this section.

To clear the top frame in the stack, the form of the call to this service is as follows:

CALL M$CLRSTK [ALTRET (label)];

To clear more that one frame, the form of the call for this service is as follows:

CALL M$CLRSTK (FPT_CLRSTK) [ALTRET (label)];

In this case, the parameters for the service is as follows:

FRAMES = VALUE-DEC(1-n) specifies the number of frames to remove from the Exceptional Condition Stack. If the value specified is equal to or greater than the number of frames currently on the stack, all frames are removed and the stack is marked empty. Default = 1.

VLP_REGS

Three monitor services that may be used to exit Exceptional Condition Processing procedures allow for the values of the index registers, the A and Q registers, and the exponent register to be replaced by values specified in the FPT. These monitor services are M$TRTN, M$RETRY and M$RETRYS.

The VLP_REGS macro may be used to define the area to contain these values. No initial values are provided for in this macro as the values must always be supplied at run time. Generally, the data area defined by the VLP_REGS macro will be initialized from the TCB frame (B$EXCFR.REGS) and selected values replaced. The monitor service FPT can then be modified to point to this area (or can be initialized at compile time to point to this area). For example, the following PL-6 statements would cause the contents of Index Register 7 and the Q register to be replaced on returning to the point of interruption:

```
%INCLUDE CP_6;
%B$TCB;
%B$EXCFR;
%VLP_REGS;
%FPT_TRTN;

DCL B$TCB$ PTR SYMREF;

VLP_REGS = B$TCB$->B$TCB.STK$->B$EXCFR.REGS;
VLP_REGS.REG.X7 = 0;
VLP_REGS.Q = 0;
FPT_TRTN.REGS_ = VECTOR(VLP_REGS);
CALL M$TRTN(FPT_TRTN);
```

A = VALUE-UBIN(36). Contains the value to be used to replace the A register.

E = VALUE-BIT(8). Contains the value to be used to replace the contents of the Exponent Register.

Q = VALUE-UBIN(36). Contains the value to be used to replace the Q register.

REG.Xn = VALUE-UBIN(18) where n= 0 through 7. Redefines the Array XREGS, and may be used to address the Index Registers individually.

XREGS = ARRAY(0:7)-UBIN(18). Contains the values to be used to replace the Index Registers X0 through X7.

VLP_PTRS

The M$TRTN, M$RETRY and M$RETRYS monitor services also allow the user to specify that the values of the pointer registers are to be replaced with values supplied in the FPT.

The VLP_PTRS macro may be used to generate the area to contain the pointers that are to be used to reload the Descriptor Registers and corresponding Address Registers. No initial values are provided for in this macro as the values must be supplied at run time. Generally, the data area will be initialized from the TCB frame (B$EXCFR.PRS) and selected values replaced. The monitor service FPT can then be modified to point to this area (or can be initialized at compile time to point to this area).

P.PRn$ = VALUE-PTR where n= 0 through 7. Redefines the Array PR$, and may be used to address the Pointer Registers individually.

PR$ = ARRAY(0:7)-PTR. Contains the values to be used to replace the Pointer Registers PR0 through PR7.

Section 7

Resource Control Services

This section explains all monitor services for resource management. The services provided for general use are presented first; restrictions on the use of the remaining services are clearly stated.

M$ENQ - Enqueue for Resource

The enqueue/dequeue services permit users to coordinate shared use of a resource. These services are used to coordinate sharing of a disk file by several jobs that run simultaneously.

All users of a resource must refer to it by the same name. The resource names for M$ENQ and M$DEQ are in no way related to physical resource names. Instead the resource names are defined by users of the resource to suit their own requirements. For example, in a random file called DATA FILE, its granules might be considered as resources and referred to as 000,001, 002, etc.

When a user enqueues on a resource name, he is effectively put in a queue to wait for availability of the resource. He remains in the queue until he specifically dequeues naming the resource or until the monitor automatically dequeues him at the end of the job step.

M$ENQ

The M$ENQ service enters the user into the appropriate queue for the resource specified by the RNAME parameter. Either the user is put to sleep until the resource is available to him (WAIT=YES), or an event is reported when the resource becomes available to that user (WAIT=NO). Once the resource becomes available to the user, it remains available to him until he dequeues or is dequeued by the monitor (as described above).

The user specifies conditions for his use of the resource by specifying the SHARE parameter and the LIMITED_COM parameter. The simplest use of enqueue/dequeue allows one group of users to have exclusive use of the resource. For users of a disk file, the SHARE parameter can be assigned as follows. (The LIMITED_COM parameter can be allowed to default.)

SHARE=ALL	Readers of the resource who want to share the file with other readers but not with updaters.
SHARE=NONE	Updaters who do not want any other users.

A more complex use of M$ENQ parameters gives greater flexibility in coordinating users that share a resource. For instance, for users of a disk file the SHARE and LIMITED_COM parameters could be assigned as follows:

SHARE=ALL
LIMITED_COM=YES
	Readers who do not care if the file is being updated as they read (statistical read).

SHARE=LIMITED
LIMITED_COM=YES
	Readers who do not want updates (exact read).

SHARE=LIMITED	Updaters who do not care that there may be statistical readers.
SHARE=NONE	Updaters who do not want any other users.

The following table fully defines the parameter combinations for simultaneous (Y) and exclusive (N) use of the resource. In the table, the SHARE parameter is shown as simply the value ALL, LIMITED, or NONE; the LIMITED_COM parameter is represented by YES or NO. All requests for the use of a resource are satisfied in the order submitted, even if it means delaying a request that is compatible with all current users of the resource.

	ALL YES	ALL NO	LIMITED YES	LIMITED NO	NONE
ALL YES	Y	Y	Y	Y	N
ALL NO	Y	Y	N	N	N
LIMITED YES	Y	N	Y	N	N
LIMITED NO	Y	N	N	N	N
NONE	N	N	N	N	N

Two types of deadlock are possible with enqueue/dequeue: multi-queue and single queue. A multi-queue deadlock occurs if two or more users have at least one resource and are waiting for at least one resource such that none of the users involved can ever get the resource for which they are waiting. This deadlock condition causes an alternate return from M$ENQ. The user receiving the error is responsible for unblocking the deadlock.

The single queue deadlock occurs when two or more users have been granted access to a resource and two of these users are attempting to upgrade their access to SHARE=NONE (by re-issuing M$ENQ without first calling M$DEQ). Since they cannot get exclusive access so long as there are SHARE=ALL or SHARE=LIMITED users of the resource, and since they have not relinquished their existing SHARE positions in the queue, deadlock is created and M$ENQ returns an error.

The form of the call for this service is:

CALL M$ENQ (FPT_ENQ) [ALTRET (label)];

The parameters for this service are as follows:

DCB = DCBNAME is a required parameter. This parameter must specify the DCB for an open disk file.

EVENT = VALUE-DEC(0-n) specifies an event identification to be reported back to the user when a WAIT=NO resource request is allocated. It is also used in conjunction with the SELECTIVE option of M$DEQ to identify sets of resources. The default is 0.

LIMITED_COM = {YES|NO} specifies, if YES, that the user can share with other users who specified SHARE=LIMITED. The default is NO.

MESSAGE = VARIABLE specifies, if WAIT=YES, the location of a 1-word area where a message is to be placed from the user who last issued a DEQ from the resource. If WAIT=NO, this message will be in the TCB. The default is NIL.

NO_DOWNGRADE = {YES|NO} specifies, if YES, that if RNAME was named in a previous enqueue request, and this enqueue request would cause a downgrade of the user's access or no change in the user's access, the alternate return should be taken. If NO_DOWNGRADE = NO is specified, this enqueue request could cause a downgrade. The default is NO.

NO_UPGRADE = {YES|NO} specifies, if YES, that if RNAME was named in a previous enqueue request, and this enqueue request would cause an upgrade of the user's access, the alternate return should be taken. If NO_UPGRADE = NO is specified, this enqueue request could cause an upgrade. The default is NO.

RNAME = VARIABLE specifies an area containing the TEXTC name of the resource that is to be shared. This area may be generated by invoking the VLP_NAME macro. This parameter must be specified.

SHARE= OPTION Three options exist for the SHARE parameter:

ALL specifies that the user will share (1) with any other user who has specified SHARE=ALL, and (2) with any other user who has specified SHARE=LIMITED, if LIMITED_COM=YES on the current call.

LIMITED specifies that the user will share with any user who has specified LIMITED_COM=YES.

NONE specifies that the user desires exclusive use of the resource.

WAIT = {YES|NO} specifies, if YES, that the program is to wait for an unavailable resource for a period of time designated by the WAIT_TIME parameter. If NO, the user is placed in the queue, control is returned to the user, and when the request for the resource can be granted, an event is posted that causes an asynchronous entry into the user. The default is YES.

WAIT_TIME = VALUE-DEC(0-n) specifies, in seconds, the maximum time that a user who specified WAIT=YES is willing to wait before the request is automatically cancelled. The upper range of this time is a system parameter determined at system definition. WAIT_TIME=-1 indicates that the user wants the maximum time available. The default is -1.

M$DEQ - Dequeue for Resource

The M$DEQ service allows a user to dequeue for a particular resource or group of resources.

The form of the call for this service is:

CALL M$DEQ (FPT_DEQ) [ALTRET (label)];

The parameters for this service are as follows:

CANCEL = {YES|NO} specifies, if YES, that any of the caller's unsatisfied requests for the designated resource element(s) are to be canceled. The default is NO.

DCB = DCBNAME specifies a DCB that is open to a disk file.

MESSAGE = VARIABLE specifies the location of a 36-bit message that can be passed to the next user of the resource. This parameter is only applicable if RELRES = YES. The default is NIL.

RELRES = {YES|NO} specifies, if YES, that designated resource element(s) are to be de-allocated. The default is YES.

RNAME = VARIABLE specifies an area containing the name of the resource that is to be affected by the DEQ. This area may be generated by invoking the VLP_NAME macro. If RNAME is not specified and SELECTIVE = NO, the DEQ applies to all resources associated with the DCB.

SELECTIVE = {YES|NO} specifies, if YES and RNAME is NIL, that the DEQ applies to all resources associated with the DCB that were enqueued with EVENT the same as this DEQ's SELECT_ID. The default is NO.

SELECT_ID = VALUE-DEC(0-n) specifies the set of resources to be dequeued if SELECTIVE = YES and RNAME = NIL. The resources selected are those that were enqueued with EVENT = SELECT_ID. The default is 0.

M$DISPRES - Return Currently Owned RES

The M$DISPRES service may be used by any program to obtain a list of the resources it currently owns.

The form of the call for this service is as follows:

CALL M$DISPRES (FPT_RESCTL) [ALTRET(label)];

The parameters for this service are as follows:

PRES = VARIABLE locates an area where the list of physical resources owned is returned. This area can be generated by invoking the VLP_PRES macro. The default is NIL.

PSEUDO = VARIABLE locates an area where the description of currently owned pseudo resources is returned. The area can be generated by invoking the VLP_PSEUDO macro. The default is NIL.

RPSN = VARIABLE locates an area where the description of pack sets is returned. This area can be generated by invoking the VLP_RPSN macro. The default is NIL.

M$RELRES - Release Resources

The M$RELRES service may be used by any program to dispose of resources it currently owns. If resources cannot be released because they are in use, their names are returned as in M$DISPRES.

The form of the call for this service is as follows:

CALL M$RELRES (FPT_RESCTL) [ALTRET (label)];

The parameters for this service are as follows:

PRES = VARIABLE locates an area which describes the physical resources to be manipulated. This area can be generated by invoking the VLP_PRES macro. The default is NIL.

PSEUDO = VARIABLE locates an 8-byte array specifying how many of each pseudo resource are to be manipulated. The area can be generated by invoking the VLP_PSEUDO macro. The default is NIL.

RELALL = {YES|NO} YES specifies that all resources owned by this user are to be released. The default is NO.

RPSN = VARIABLE locates an area which describes the pack sets to be manipulated. This area can be generated by invoking the VLP_RPSN macro. The default is NIL.

M$REQUIRE - Require Pseudo Resources

The M$REQUIRE service is used typically by processors which require the user to have certain pseudo resources. If the resources are not owned by the user, a severe error is given and the processor may abort.

The form of the call for this service is as follows:

CALL M$REQUIRE (FPT_RESCTL) [ALTRET (label)];

The parameter for this service is as follows:

PSEUDO = VARIABLE locates an area which describes the pseudo resources to be manipulated. This area can be generated by invoking the VLP_PSEUDO macro. The default is NIL.

M$LIMIT - Get RES for Current User

The M$LIMIT service is usable only by command processors to obtain resources for a running user, usually as a result of an online or job-step !RESOURCE command.

The form of the call for this service is as follows:

CALL M$LIMIT (FPT_RESCTL) [ALTRET (label)];

The parameters for this service are as follows:

PRES = VARIABLE locates an area which describes the physical resources to be manipulated. This area can be generated by invoking the VLP_PRES macro. The default is NIL.

PSEUDO = VARIABLE locates an area which describes the pseudo resources to be manipulated. This area can be generated by invoking the VLP_PSEUDO macro. The default is NIL.

RPSN = VARIABLE locates an area which describes the pack sets to be manipulated. This area can be generated by invoking the VLP_RPSN macro. The default is NIL.

M$MBS - Get RES for New Batch User

The M$MBS service is used only by the Multi-Batch Scheduler (MBS). This service obtains the specified resources if MBSDISP=NO. If MBSDISP=YES, available resources are returned in the areas specified by PRES, PSEUDO, and RPSN for the specified system.

The form of the call for this service is:

CALL M$MBS (FPT_RESCTL) [ALTRET(label)];

The parameters for this service are as follows:

MBSDISP = {YES|NO} YES specifies that available resources are to be returned. NO specifies that the named resources are to be obtained. The default is NO.

PRES = VARIABLE locates an area which describes the physical resources to be manipulated. This area can be generated by invoking the VLP_PRES macro. The default is NIL.

PSEUDO = VARIABLE locates an 8-byte array specifying how many of each pseudo resource are to be manipulated. This area can be generated by invoking the VLP_PSEUDO macro. The default is NIL.

RPSN = VARIABLE locates an area which describes the pack sets to be manipulated. This area can be generated by invoking the VLP_RPSN macro. The default is NIL.

SYSID = VALUE-DEC(0-n) specifies the SYSID for which to reserve resources. The default is 0.

VLP_PSEUDO

The VLP_PSEUDO structure is used to specify pseudo resources to be manipulated by the resource management monitor services. Pseudo resources are defined via the TIGR processor and manipulated via the CONTROL processor. For information on TIGR and CONTROL, refer to the CP-6 System Support Reference Manual. On M$DISPRES this structure is filled with the information on currently owned pseudo resources. On M$RELRES this VLP returns information on resources which could not be released.

The area generated by VLP_PSEUDO often contains a single pseudo resource name, but may contain a list of up to eight names. Each entry in the list includes the pseudo resource name and the number of that type of pseudo resource to manipulate. For multiple list entries in the macro invocation, the form of the macro invocation is as shown in the following example. Double quotation marks are required when multiple names and values are specified. In this example, the list generated contains these entries: 4 of pseudo resource A, 3 of B, 1 of C, and 1 of D.

Example:

```
%VLP_PSEUDO (N=4,PSEUDO="'A','B','C','D'",NUM="4,3,1,1");
```

The contents of the VLP_PSEUDO structure are as follows:

ENQBLOCKS = VALUE-DEC(0-n) specifies the user's ENQ/DEQ memory requirement in increments of 4-words. This facility is not yet implemented. On M$DISPRES this value is returned. The default is 0.

FAILX = VALUE-DEC(0-7) contains the index of a list entry, when an error is returned on a monitor service involving an area generated by VLP_PSEUDO. If it is MEMORY that cannot be allocated, FAILX is set to -1.

MEMORY = VALUE-DEC(0-n) specifies the user's resource memory requirement in increments of 1K words. On M$DISPRES this value is returned. The default is 0.

N = VALUE-DEC(1-8) specifies the number of pseudo resources contained in VLP_PSEUDO.PSLIST. The fully qualified name of this field is VLP_PSEUDO.NN#. The default is 1.

NUM = 1 to 8 VALUES-DEC(1-n) Represents how how many of the pseudo resources are to be obtained (MMBS, MLIMIT), released (M$RELRES), or required (M$REQUIRE). For M$DISPRES the number currently owned is returned here. The fully qualified name of this field is VLP_PSEUDO.PSLIST.NUM#. The default is 1.

PSEUDO = 1 to 8 VALUES-CHAR(8) specifies the name(s) of the pseudo resource(s) being manipulated. These names are established via the TIGR processor by the system manager. The fully qualified name of this field is VLP_PSEUDO.PSLIST.PSEUDO#. The default is blank.

VLP_PRES

This VLP is used to specify physical resources to be manipulated by the resource management monitor services. On M$DISPRES this structure is filled with the information on currently owned physical resources. This VLP must not be in CONSTANT storage since the monitor may store into this area (see RAT$ below).

The VLP_PRES area often contains a single physical resource name, but may contain a list of physical resources. Each entry in the list includes the the physical resource name, attributes for the resource, and for M$LIMIT and M$MBS a pointer. For a list, the form of the macro invocation is as shown in the following example. Double quotation marks are required when multiple values are specified. In this example, the list generated contains these entries: MT03, LP02, PL01.

Example:

```
%VLP_PRES (N=3,RNAME="'MT03','LP02','PL01'");
```

The contents of the VLP_PRES structure is as follows:

ATTR = 1 to n VALUES-BIT specify the attributes required on the physical device. Examples are DENSITY for tapes and lower-case print capability for printers. The bit strings for attributes are defined in the file CP_6_SUBS. This field has meaning only for M$LIMIT and M$MBS as a parameter and for M$DISPRES as a returned value. The fully qualified name of this field is VLP_PRES.PRLIST.ATTR#. The default='0'B.

FAILX = VALUE-DEC(0-n) contains the index of a list entry when an error is returned on a monitor service call as a result of this VLP.

N = VALUE-DEC(1-n) specifies the number of physical resource descriptions contained in VLP_PRES.PRLIST. The fully qualified name of this field is VLP_PRES.NN#. The default is 1.

RAT$ is a pointer used by the monitor on M$LIMIT and M$MBS. Because the monitor stores a value into this PTR variable, VLP_PRES must not be in CONSTANT storage. The contents of this area are not significant to the caller of the monitor service.

RNAME = 1 to n VALUES-CHAR(4) specifies the name(s) of the physical resource(s) to be manipulated. The name(s) must be in text form: 'dvnn' where dv is a device type (e.g., MT, LP) and nn is a 2-digit number from 1 to 99. This name is used later to access the resource (C TO ME LP37#). The fully qualified name of this field is VLP_PRES.PRLIST.RNAME#. The default is blank.

VLP_RPSN

This structure is used to describe pack sets to be manipulated on resource management monitor services. On M$DISPRES the structure is filled with information on the pack sets currently reserved.

The VLP_RPSN area often contains a single pack set name, but may contain a list of pack sets. Each entry in the list includes the pack set name and a usage field set to indicate that the pack set is either shared or for exclusive use.

Example:

```
%VLP_RPSN (N=1,PSN='PAC',USAGE=EXCLUSIVE);
```

The contents of the VLP_RPSN structure are as follows:

DISP = OPTION applies to M$RELRES only. This parameter specifies the disposition for the specified resource(s) or set name(s). A tape volume set or pack set is always dismounted if it is in exclusive use. In addition, the options specify that these actions are to be taken:

REM	Release the set name(s) if the user has no files open.
RELEASE	Release the physical resource(s) if the set dismounts.
REL	Release the set name(s) if the user has no files open, and release the physical resource(s) if the set dismounts.
NOREL	No additional action.

The default is RELEASE.

FAILX = VALUE-DEC(0-7) contains the index of a list entry when an error is returned on a resource management monitor service as a result of this VLP.

N = VALUE-DEC(1-n) specifies the number of pack sets described in VLP_PRLIST. The fully qualified name of this field is VLP_RPSN.NN#. The default is 1.

PSN = 1 to n VALUES-CHAR(6) specifies the pack set name(s) of the pack set(s) to be manipulated. The fully qualified name of this field is VLP_RPSN.RPLIST.PSN#. The default is blank.

USAGE = {SHARED|EXCLUSIVE} Is specified as a parameter for M$LIMIT or M$MBS and is returned on M$DISPRES. USAGE defines whether the pack set is shared or for exclusive use only. USAGE cannot be specified at compile time if the N parameter is greater than 1 and must be specified at run time instead. The default is %SHARED#.

Section 8

Special Services

The monitor services in this section are intended for use solely by the advanced system programmer and therefore, they are restricted. In many cases, the user must have special privileges before these services can be executed within the user domain.

M$GJOB - START GHOST OR TP JOB

The M$GJOB monitor service allows users to start GHOST or TP jobs by logging them onto the system. The user being started must be defined in :USERS by SUPER as a valid logon for the mode desired, GHOST or TP, unless FPT_GJOB.V.SLEAZE# is set. This bit, honored for system ghosts only, specifies that the user is not to go through logon. System ghosts may also specify FPT_GJOB.V.PRIINC# as a base priority increment for the ghost being started. In general, the user may specify ACCT, NAME, PASS of the user to be logged on. A free command, CCBUF, is also allowed.

The call is of the form:

CALL M$GJOB (FPT_GJOB) [ALTRET (label)];

The parameters for this service are as follows:

ACCT = VARIABLE. Locates a VLP_ACCT area containing the account of the user to be logged on. If NIL, the caller's ACCT will be used.

CCBUF = VARIABLE. Locates a text string to be placed into JIT.CCBUF for the user. This is analogous to the "first command" option to SUPER. If CCBUF is NIL, the first command specified by SUPER for this user will be used; if not NIL, this will override any first command specified by SUPER. The CCBUF string is limited to 256 characters.

NAME = VARIABLE. Locates a VLP_NAME area containing the user name for the user to be started. If NIL, the caller's NAME will be used. Names are limited to twelve characters.

PASS = VARIABLE. Locates a VLP_PASS area containing the password of the user to be started. If NIL, a password of all blanks will be used. All blanks is legal if the user to be started is the same ACCT,NAME as the caller or if the user to be started has no password.

PRIINC = VALUE-DEC. Specifies the boost to base scheduling priority to give the ghost being started. This option is honored only if the caller is a system ghost.

RESULTS = VARIABLE. Locates a VLR_GJOB area into which the sysid of the started user is returned.

SLEAZE = {YES|NO}. YES specifies that the ghost being started is not to go through logon. It does not need to be defined by SUPER. This option is honored only if the caller is a system ghost.

TPU = {YES|NO}. YES specifies that a transaction processing user rather than a ghost is to be started.

VLR_GJOB

This macro generates a VLR area definition for RESULTS parameter of the M$GJOB service.

SYSID = VALUE-DEC(0-n) Contains the SYSID of the ghost started by a successful M$GJOB monitor service.

M$FSUSER - Find Suspended User

This monitor procedure is available only to the host log-on processor for the purpose of finding whether or not a newly logged-on user has a suspended program as a result of an earlier line disconnect.

M$FSUSER returns normally if a suspended program was found for the current user; otherwise it takes the alternate return.

The form of the call for this service is:

```
CALL M$FSUSER [ALTRET (label)];
```

There is no FPT for M$FSUSER. This service obtains the user's account and name from the JIT.

M$ASUSER - Associate Suspended User

This monitor routine is available only to the host log-on processor for the purpose of associating a newly logged-on user with a suspended program saved as a result of an earlier line disconnect. It is intended to be used following a successful return from M$FSUSER.

M$ASUSER, upon finding the user's suspended program, "attaches" it to the communication line owned by the newly logged on user and awakens the program. After the user is reconnected to the saved program, the skeletal user program formed to accomplish the new log-on is disconnected from any communication line. Thus, upon a successful return from M$ASUSER, the user will be communicating with his previously saved program.

In the unlikely event that the user's suspended program timed out between the time the M$FSUSER and M$ASUSER calls were made, the alternate return is taken.

The form of the call for this service is:

```
CALL M$ASUSER [ALTRET (label)];
```

There is no FPT for M$ASUSER. This service obtains the user's account and name from the JIT, and assumes that the M$UC DCB is open.

M$DSUSER - Delete Suspended User

This monitor routine is available only to the host log-on processor for the purpose of deleting the current user's suspended program. In the event that a suspended program for the user does not exist, the alternate return is taken.

The form of the call for this service is:

```
CALL M$DSUSER [ALTRET(label)];
```

There is no FPT for M$DSUSER. This service obtains the user's account and name from the JIT.

M$CPEXIT - Exit from Command Processor

A Command Processor becomes associated with a user whenever:

1. The Command Processor has been specified via the SUPER user authorization CPROC option.

2. One Command Processor issues an M$CPEXIT Service Request specifying another Command Processor be associated with the user.

Once associated with a user, a command processor will be entered at its start address under any of the following conditions:

1. The user is at Job Step (as will be the case on the initial entry to the Command Processor).

2. The user has aborted and is about to be rundown.

3. A time-sharing user has typed a Control-Y sequence on his terminal.

4. A user program has issued an M$YC service request.

5. A user program has issued an M$LINK or M$LDTRC monitor service request using the CP_CMD option.

The CP-6 monitor communicates the reason for entry to the command processor via bit settings in B$JIT.CPFLAGS1 as follows:

CP_JSTEP#	the user is at Job Step.
CP_RUND#	the user is about to be rundown.
CP_YC#	the time-sharing user has typed a Control-Y sequence.
CP_YCPMME#	the user program has issued an M$YC.
CP_LNKPMME#	the user run unit has issued an M$LINK or M$LDTRC with a command to be parsed.

Another interesting bit in B$JIT.CPFLAGS1 is CP_LOGOFF#. This bit is set whenever the system has detected a line hang-up of a time-sharing terminal or an operator abort of a user. This bit may be set in conjunction with any of the other bits mentioned above. When set it indicates to the command processor that no more job steps are allowed.

The Command Processor communicates the action to be taken for this user via the various options of M$CPEXIT. This monitor service is used to:

- Initiate execution of a user program or shared processor.

- Resume execution of an interrupted program (following Control-Y or an M$YC service call).

- Associate a debugger with a (possibly interrupted) user program.

- Remove a user from the system.

- Save the user environment on disk storage

Before a Command Processor issues an M$CPEXIT, it must have closed all its DCBs and, in general, have cleaned up. Data Segments that have been obtained via M$GDS will be released unless CP_KEEPDS# has been set in B$JIT.CPFLAGS1.

The form of the call for this service is:

CALL M$CPEXIT (FPT_CPEXIT) [ALTRET (label)];

The parameters for this service are as follows:

ACCT = VARIABLE specifies location of a TEXT string of eight characters, designating the account in which the run unit resides (:SYS is specified for shared processors). The default is NIL.

ALTRTN = {YES|NO} specifies, if YES, that an ALTRET to the user's M$YC service is to be taken. When ALTRTN is specified, the error code to be returned to the user may also be specified via the FCG, MID, CODE and SEV options. This option is ignored if CONT is also specified. The default is NO.

CODE = VALUE-DEC(0-16383) specifies the value of the error number in the error code. This parameter is only valid if CPOFF=YES or ALTRTN=YES, and allows the Command Processor to generate a specific error code.

CONT = {YES|NO} specifies, if YES, that the currently interrupted program is to be resumed, possibly under control of a debugger. If both CONT and ALTRTN are specified, the ALTRTN option is ignored. The default is NO.

CPOFF = {YES|NO} specifies, if YES, that the Operator Abort sequence is to be initiated. When CPOFF = YES is specified, the error code to be returned to the user may also be specified via the FCG, MID, CODE and SEV options. The default is NO.

DEBUG = VARIABLE specifies the VLP containing a TEXTC string of up to 11 characters, designating a debugger to be associated with the user. The default is NIL.

FCG = VALUE-BIT(12). specifies the value for the function code group in the error code. This parameter is only valid if CPOFF=YES or ALTRTN=YES, and allows the Command Processor to generate a specific error code.

GETFLG = {YES|NO} specifies, if YES, that the Command Processor is requesting the monitor to restore (GET) a previously saved program, using the FID specified in the M$CPEXIT FPT. The default is NO.

LINK = {YES|NO} specifies, if YES, that the environment of the current run-unit is to be saved on disk storage in the *N file; i.e. the M$LINK process is to be simulated. After the image is saved control will return to the command processor at the normal return of the M$CPEXIT. Default = NO.

MID = VALUE-BIT(6). specifies the value for the module ID in the error code. This parameter is only valid if CPOFF=YES or ALTRTN=YES, and allows the Command Processor to generate a specific error code.

NAME = VARIABLE specifies the location of a TEXTC string of up to 31 characters, designating the requested run unit or shared processor. The default is NIL.

M$CPEXIT issued with no NAME and all flags specified as NO (the default) constitutes a request to remove the user from the system. This request must never be issued if the user is not at job step.

PASS = VARIABLE specifies the location of a TEXT string of eight characters, designating the password associated with the run unit. The default is NIL.

PSID = VARIABLE specifies the location of a TEXT string of six characters, designating the identification of the pack set on which the run unit is located. The default is NIL.

QUIT = {YES|NO} specifies, if YES, that the currently interrupted program has been errored (Control-Y followed by QUIT). Any Exit Control established by the user program and/or ASL will be honored. The Command Processor will then be re-entered. The default is NO.

SAVEFLG = {YES|NO} specifies, if YES, that the Command Processor is requesting the monitor to perform a SAVE upon the currently running user program, with the FID in the M$CPEXIT FPT used as the FID for the SAVE file. The default is NO.

SEV = VALUE-DEC(0-7) specifies the value for the severity level in the error code. This parameter is only valid if CPOFF=YES or ALTRTN=YES, and allows the user to generate a specific error code.

M$FINDPROC - Find Shared Processor

The M$FINDPROC service allows a program to either:

1. Determine a processor number and/or processor type given a processor name.
2. Determine a processor name given the processor number, and optionally determine the type.

The function that is performed is determined by the contents of the NAME parameter.

The form of the call for this service is:

CALL M$FINDPROC (FPT_FINDPROC) [ALTRET (label)];

The parameters for this service are as follows:

ACCT = VARIABLE specifies the location of an area containing 8 TEXT characters, designating the account of the shared processor.

If the call to M$FINDPROC was to search the shared processor tables for the processor specified by the NAME parameter, this ACCT parameter may be used to qualify NAME. If the ACCT parameter is omitted, :SYS will be assumed.

If the call to M$FINDPROC was to return the processor name of the processor specified by the PNUM parameter, the account will be returned in the area specified by ACCT.

The default is NIL.

NAME = VARIABLE specifies the location of a TEXTC string of up to 31 characters, designating the name of the shared processor.

If the length byte in this TEXTC location is non-zero, the shared processor tables will be searched for a processor from the account specified by the ACCT parameter. If a shared processor of this name cannot be found in the system shared processor tables the ALTRET routine will be entered.

If the length byte in this TEXTC field is zero, the name of the processor specified by the PNUM parameter will be returned in this location.

The default is NIL.

PNUM = VARIABLE specifies the location of an area that contains the processor number of the processor whose name is to be returned in the area specified via the NAME option.

If this is a request to determine the processor type, the processor number will be returned here if this parameter is not NIL.

This area should be defined as UBIN(9) DALIGNED.

Default = NIL.

PTYPE = OPTION specifies the processor type as follows:

P_SP	– Standard Shared Processor
P_LIB	– Run-time Library
P_ASL	– Alternate Shared Library
P_DB	– Debugger
P_CP	– Command Processor

If a shared processor of the name or number specified is found but is not of the type specified, the ALTRET routine is entered. If this parameter is not specified, no check is made of the processor type; this is the default.

RTYPE = VARIABLE specifies the location of an area where the processor type may optionally be returned. This area may be generated using the VLR_RTYPE macro. The bit strings for the processor type are defined in the file CP_6_SUBS. The default is NIL.

VLR_RTYPE

The VLR_RTYPE macro generates an 18-bit area which receives the processor type as a result of a call to M$FINDPROC. For the processor types that may be returned, see the PTYPE parameter of M$FINDPROC.

M$DRTN - Debugger Return

The M$DRTN monitor service provides a way for an Interactive Debugger (hereafter refered to as DELTA) to return control to the user program. A description of the conditions that cause DELTA to receive control and of the environment that exists when DELTA is entered is in order here.

DELTA is entered under any of the following conditions:

1. Initial entry to DELTA:

   a. A program is started under DELTA.
   b. DELTA is invoked at Job-Step with no Run-unit associated.
   c. DELTA is invoked after a user program is in execution by striking Control-Y and asking for DELTA.

2. An overlay of the user's program is loaded.

3. A program is put into execution via M$LINK or M$LDTRC or an M$LINKed to program is restored.

4. An Exceptional Condition occurs other than:

   a. Line hang-up
   b. Operator !X key-in
   c. Bad call, and user has specified ALTRET.

5. A user is exiting an Exceptional Condition processing procedure.

6. DELTA is associated via an M$ALIB service request. If the program making the request is not executing under control of DELTA (see Item #1), DELTA is thereafter entered only on subsequent M$ALIB requests or for requests to be put under control of DELTA. Note that this may or may not be the initial entry to DELTA.

7. A data breakpoint has been hit.

8. The user program has issued an M$SETFP service request or a state change is to be reported for an FPRG that is being debugged.

When DELTA is entered a standard Exceptional Condition frame containing a copy of the user's Safe-Store frame will be placed in DELTA's TCB. The Exceptional Condition Code (B$EXCFR.ECC), Sub-code (B$EXCFR.SUBC) and Event ID (B$EXCFR.EVID) uniquely identify the condition that caused the entry to DELTA. The %SUB_EXC and %SUB_ECCDELTA macros from the system macro library provide string substitutions for the values of these fields as follows:

ECC	SUBC	Reason for entry
DELTA specific ECCs:		
ECC_DELTA#	SC_STARTU#	User program started under DELTA.
	SC_JOBSTEP#	DELTA was invoked at Job-Step time. No Run-unit associated.
	SC_YC#	Post association of DELTA while the user program is in execution. B$EXCFR.EVID will contain one of the following values: EVID_USER# EVID_AUTOS#
ECC_OLAY#	Contains the Node#	Overlay has been loaded. B$EXCFR.EVID will contain one of the following values from the M$OLAY FPT: EVID_CANCEL# EVID_ENTER# EVID_NOPATH#
ECC_LINK#	SC_MLINK#	User program entered via M$LINK.
	SC_MLDTRC#	User program entered via M$LDTRC.
ECC_LRTN#	-	M$LINKed to program has been restored.
ECC_ALIB#	-	Debugger was invoked via M$ALIB. B$ALIB[F].CMDSZ contains the byte size of the command. B$ALIB[F].REPLYSZ contains the byte size of the reply area. B$ALIB[F].WHO is set as follows: SC_AUSR# - User Program SC_AASL# - Alternate Shared Library SC_ASHR# - Standard Shared Processor SC_EXUO# - Execute-only Run unit
ECC_EXCRTN#		Exit from a user's exceptional Condition procedure. B$EXCFR.EVID contains the address of the call.
	SC_TRTN#	M$TRTN
	SC_MERC#	M$MERC
	SC_MERCS#	M$MERCS
	SC_RETRY#	M$RETRY
	SC_RETRYS#	M$RETRYS
	SC_XCONXIT#	Final Exit from Exit Control. In this case B$EXCRTN.TYP has the following values:

		XCON_EXIT# - M$EXIT XCON_ERR# - M$ERR XCON_XXX# - M$XXX
ECC_DBRK#	-	Data Break Point
ECC_EVENT#	SUBC_FPRG#	M$SETFP service request complete. B$FPRG.SUBC = %SUBC2_AFD# B$FPRG.DCBNO will contain the DCB number of the FPRG
	SUBC_COMIO#	FPRG state change message is to be read. B$EXCFR.EVID will be zero. B$COMIO.DCBNO will contain the DCB number of the COMIO

ECC	SUBC	Reason for entry
User Exceptional Condition ECCs:		
ECC_TIMER#	-	M$STIMER specified interval has expired.
ECC_EVENT#	As specified by user	Event over which the user has requested control has occurred.
ECC_INT#	SC_BRK#	Time-sharing terminal break key.
	SC_BYC#	DELTA request from the Command Processor when DELTA is already associated.
ECC_XCON#	Refer to Section 6	User exit condition, normal or abnormal.
ECC_PMME#	Refer to Section 6	Error on Monitor Service request. No ALTRET specified on user's call.
ECC_ARITH#	Refer to Section 6	User has caused an Arithmetic fault.
ECC_PROG#	Refer to Section 6	User has caused a Programmed fault.
ECC_ERROR#	Refer to Section 6	User has caused an Error class fault.

When DELTA is entered because of a user's Exceptional Condition, the ECC and the remainder of the frame will reflect what would have been placed on the user's TCB had the user not been running under DELTA and had established control of the specific condition. No determination has been made as to user specified Exceptional Condition control requests. The Exceptional Condition frame will be moved to the user's TCB and his procedure to handle the condition will be entered only when this action is specified via options of the M$DRTN FPT. (Refer to the description of the SETECC and ECC options below.)

Word 1 of the TCB frame will be non-zero if the ASL was in control upon entry to DELTA. Note that this can happen only if the user has hit Break and the ASL has not requested break control or if the user has hit Control-Y and invoked DELTA while the ASL is in control.

DELTA has access to the user's Working Space through descriptors stored in the Special Descriptor Access descriptor slots in DELTA's Linkage Segment. The following pointers (which are DEFed in B_USRPTRS_D) may be used to access the user's area:

B$SPCL1$ ->	the user's Safe-Store frame
B$SPCL2$ ->	the user's Linkage Segment
B$SPCL3$ ->	the user's Argument Segment
B$SPCL4$ ->	the user's Parameter Segment
B$SPCL5$ ->	the user's Instruction Segment

The first four of these descriptors are type 1; the user's Instruction Segment descriptor is type 0. The Special Access Descriptors 2 through 5 are a copy of those from the user's Safe-Store frame. Unless DELTA is being entered as a result of an M$ALIB from an ASL, shared processor or execute-only run unit, the Page Table write control bit for procedure pages in the user's ISR is set prior to entry to DELTA; it is reset when DELTA returns to the monitor via M$DRTN.

The monitor normally enters DELTA via the LTRAD instruction. However, if DELTA is being entered as a result of an M$ALIB request, the monitor will LTRAS to DELTA making the user's M$ALIB FPT available through DELTA's Parameter Stack:

Descriptor 0 -	frames the Debugger name.
Descriptor 1 -	frames the area containing the command.
Descriptor 2 -	frames the area where DELTA may return a reply.
Descriptor 3 -	frames the V area of the M$ALIB FPT. FPT$ALIB_V may be used to define the based structure of this area.

DELTA returns control to the user program via the M$DRTN service request. The form of the call for this service is as follows:

CALL M$DRTN(FPT_DRTN) [ALTRET(label)];

The parameters for this service are as follows:

ALIB_FD = {YES | NO} specifies, if YES, that the LCP-6 debugger that communicates with the debugger issuing the M$DRTN request is to be associated with the FPRG specified via the FPRG_DCB option and that that debugger is to be entered for communicating with this Host debugger. The debugger making this request will be re-entered at the instruction following the M$DRTN Service Request. Default = NO.

ALTRTN = {YES|NO} specifies, if YES, that the user's M$ALIB service request ALTRET procedure is to be entered. The error code to be returned to the user may be specified via the ECCS option (VLP_ECCS.ERR). This option is valid only when DELTA is entered with B$EXCFR.ECC = ECC_ALIB#. When the ALTRTN option is specified all other options, other than the error code and the DLIB option specified via the ECCS parameter, are ignored. Default = NO.

DBRK ={YES|NO} specifies, if YES, that the entries in the user's Page Table that have the SCDBRK Software Control bit set are to have the write control bit reset. Default = NO.

DLIB = {YES|NO} specifies, if YES, that DELTA is to be disassociated from the user. Valid only when DELTA has been entered with B$EXCFR.ECC = ECC_ALIB#. The disassociation will not take place if the program was started under control of DELTA (as indicated by %JJ_UDELTA# set in B$JIT.JUNK). Default = NO.

DRS = VARIABLE Specifies the location of an 8 word block that contains the pointers to be used to reload the user's Pointer Registers. Note that the SID field must reference the user's Linkage Segment. The structure for this area may be generated by using the VLP_PTRS or VLP_DRS macro. Default = NIL.

ECC = VALUE-DEC(0-n) specifies the value that is to replace the B$EXCFR.ECC value of the Exceptional Condition frame. The value of B$EXCFR.ECC is used by the monitor to determine how to proceed. This option is ignored if SETECC = NO; the value from the TCB frame will be used. Basically, the monitor proceeds as follows:

1. ECC = DELTA specific ECC

   The frame is removed from DELTA's TCB and the user is re-entered at the address specified via the IC=value option.

2. ECC = User Exceptional Condition ECC

   If the user has not established control over the condition DELTA will be re-entered at the M$DRTN ALTRET address. B$ALT.ERR in DELTA's TCB ALTRET frame will be set to E$NOCONTROL.

If the user has established control over the condition the frame is moved from DELTA's TCB to the user's TCB and the user will be re-entered at his Exceptional Condition processing procedure. If there is insufficient room in the user's TCB to contain the frame, the frame is removed from DELTA's TCB and DELTA will be re-entered at DELTA's start address with a user's Exit Control frame on DELTA's TCB.

ECCS = VARIABLE Specifies a 4 word block that contains the data to replace words 64-67 of the Exceptional Condition Frame prior to entering the user's Exceptional Condition procedure. If the SETECC and ECC parameters indicate the original ECC is to be changed, word 67 (B$EXCFR.P#) will be set to 0. The VLP_ECCS macro may be used to generate this area. Default = NIL.

EVENT = {YES|NO} specifies, if YES, that any outstanding events belonging to this user are to be deferred until after the user program has been re-entered. Default = NO.

FPRG_DCB = VALUE-DEC(0-N) specifies the DCB# through which DELTA communicates with its counterpart LCP-6 debugger. This DCB must have been previously opened DEVICE, RES=FE[nn], ORG={DBGDCB | DBGCG | DBGSYSID}. See M$OPEN for more details. Default = 0.

FPRG_FID = VARIABLE Specifies the location of the area where the fid for the FPRG specified by the FPRG_DCB option is to be returned. This option is ignored unless WAKE_DB or ALIB_FD is also specified. The VLR_FPRG_FID macro should be used to generate this area. Default = NIL.

IC = VALUE-DEC(0-n). Specifies the value that is to be used to replace the 18-bit IC in the user's Safe-Store frame. Default = 0. This value should always be specified.

IR = VALUE-BIT(18) Specifies the value to be used to replace the Indicator Register in the user's environment. Ignored if SETIR=NO. Default = '000'O.

KLDELTA = {YES|NO} specifies, if YES, that DELTA is to be disassociated from the user. This option differs from the DLIB option in that the disassociation will take place unconditionally. Default = NO.

KPDELTA = {YES|NO} specifies, if YES, that DELTA is to remain associated with the user program following an M$ALIB Service Request. This option is valid only if DELTA was entered with B$EXCFR.ECC = ECC_ALIB# and then takes precedence over the DLIB option. The KPDELTA option is ignored if the M$ALIB Service Request was made from an Alternate Shared Library. Default = NO.

MRGSS = {YES|NO} specifies, if YES, that the user's Safe-Store frame and the user's environment in DELTA's TCB are to be updated with the data specified via the REGS and DRS parameters. No other changes are made to DELTA's environment or to the user's environment. All options other than MRGSS, REGS and DRS are ignored. DELTA will then be re-entered at the instruction following the M$DRTN Service Request. Default = NO.

QUIT = {YES|NO} specifies, if YES, the user's program is to be run down without entering user level Exit Control or re-entering DELTA. Default = NO.

REGS = VARIABLE Specifies an 8 word block that contains the values to replace X0-X7, A, Q and E of the user's environment. The structure of this area may be generated by using the VLP_REGS macro. Default = NIL.

SETECC = {YES|NO} specifies, if YES, that the value specified via the ECC option is to be used to replace the value of B$EXCFR.ECC . Default = NO.

SETIR = {YES|NO} specifies, if YES, that the value specified by IR=value is to be used to replace the user's Indicator Register. Default = NO.

SETSTEPCC = {YES|NO} specifies, if YES, that the value specified via the STEPCC option is to be used to replace the value of B$JIT.STEPCC . Default = NO.

SPCLD = {YES|NO} specifies, if YES, that the Special Access Descriptors in DELTA's Linkage Segment are to be refreshed to the values that existed upon entry to DELTA. All other options, including MRGSS, are ignored. DELTA will be re-entered at the instruction following the M$DRTN Service Request. Default = NO.

STEPCC = {OK|ERROR|ABORT} specifies how the Step Condition Code is to be set: OK=0, ERROR=4, ABORT=6. The parameter may also be specified as a decimal value (0-511). The default is 0.

WAKE_FD = {YES | NO} specifies, if YES, that the FPRG specified via the FPRG_DCB option is to be suspended and the LCP-6 debugger associated with that FPRG is to be entered to communicate with this Host debugger. The debugger making this request will be re-entered at the instruction following the M$DRTN service request. This option is used when the LCP-6 debugger has been previously associated (see the ALIB_FD option). Default = NO.

VLP_ECCS

The VLP_ECCS macro may be used to generate the area to contain the values to be used to modify the user's TCB frame when calling the M$DRTN service.

No initial values are provided for in this macro, as the values must be supplied at run time.

ECCS.EVSC = VALUE-DEC(0-n) Specifies the ECC subcode.

ERR = VALUE-BIT(36) Specifies the error code.

EVID = VALUE-DEC(0-n) Specifies the event ID.

P# = VALUE-DEC(0-n) Specifies the number of parameters which follows in this frame. This value will be forced to zero if there is a change to the ECC.

SUBC = VALUE-DEC(0-n) Specifies the ECC subcode which is redefined as ECCS.EVSC.

VLR_FPRG_FID

A host debugger may use the VLR_FPRG_FID macro to generate an area where the fid of the specified FPRG is to be returned on an M$DRTN monitor service request which has the ALIB_FD option specified.

ACCT = CHAR(8). Contains the ACCounT from where the FPRG was fetched.

MODTIME = VALUE-UBIN(36). Contains the MODification TIME of the FPRG to be associated.

NAME = TEXTC(31). Contains the FPRG NAME of the file.

PASS = CHAR(8). Contains the PASSword associated with the FPRG.

PSN = CHAR(8). Contains the Pack Set Name of the pack set on which the FPRG resides.

VLP_DRS

The M$DRTN monitor service allows the user to specify that the values of the pointer registers are to be replaced with values supplied via the FPT. The VLP_DRS macro generates an area to contain the pointers that are to be used to reload the Descriptor Registers and corresponding Address Registers. No initial values are provided for in this macro.

PR0 - PR7 = 8 POINTERS Specifies the pointer contents. Each pointer is defined with a 24-bit AR field (VALUE-DEC) followed by an 18-bit SID field (VALUE-BIT).

SUB_ECCDELTA

The %SUB_ECCDELTA macro generates string substitutions of the ECC and ECC Sub-code that is passed to DELTA in the user environment frame on DELTA's TCB.

B$ALIBF and B$ALIB

The B$ALIBF macro may be used by an Interactive Debugger to generate a based structure of the frame placed on the debugger's TCB as a result of a M$ALIB Monitor Service request. The B$ALIB macro may be used to generate a based structure of only the last 4 words of this frame.

The fully qualified name of fields in these structures begins with the name B$ALIB or B$ALIBF. The sub-fields in these structures are as follows:

B$ALIBF.CMDSZ = VALUE-UBIN(20). Contains the byte size of the command that has been supplied by the user via Parameter Stack Descriptor 1.

B$ALIBF.ERR = BIT(36). Contains the error code (will be zero).

B$ALIBF.P# = VALUE-UBIN. Contains a count of the number of additional words in the Exceptional Condition frame. For an M$ALIB frame this value will always be zero.

B$ALIBF.REPLYSZ = VALUE-UBIN(20). Contains the byte size of the reply area that has been supplied by the user via Parameter Stack Descriptor 2.

B$ALIBF.SSFRAME = ARRAY(0:63) BIT(36). Contains a copy of the environment at the time of the M$ALIB Service Request. The based structure B$EXCFR may be used to access the environment. This field is present only in the B$ALIBF macro.

B$ALIBF.WHO = VALUE-UBIN(9). Contains the ECC_ALIB# sub-code, which indicates the type of Run-unit that has made the M$ALIB request.

B$EXCRTNF and B$EXCRTN

The B$EXCRTNF macro may be used by an interactive Debugger to generate a based structure of the frame placed on the debugger's TCB when the ECC = %ECC_EXCRTN#. The B$EXCRTN macro may be used to generate a based structure for only the last 4 words of this frame.

The fully qualified name of fields in these structures begins with the name B$EXCRTN or B$EXCRTNF. The sub-fields in these structures are as follows:

B$EXCRTNF.ECCS = UBIN(18). Contains the %ECC_EXCRTN# Sub-code, which indicates which monitor service the user has executed to exit an exceptional condition processing procedure. Refer to the description of %SUB_ECCDELTA.

B$EXCRTNF.ERR = BIT(36). Contains the error code (if any).

B$EXCRTNF.EVID = UBIN(36). Contains the address of the exceptional condition return service request.

B$EXCRTNF.P# = VALUE-UBIN. Contains a count of the number of additional words in the Exceptional Condition frame.

B$EXCRTNF.SSFRAME = ARRAY(0:63) BIT(36). Contains a copy of the environment at the time of the exit from the Exceptional Condition. The based structure B$EXCFR may be used to access the environment. This field is present only in the B$EXCRTNF macro.

B$EXCRTNF.SVL = BIT(9). This field is non-zero only when the user is exiting an Exit Control procedure after being entered for SAVE or M$LDTRC. This field will contain one of the following values (from %EQU's found in B$JIT):

%SVL_EXIT#	User has issued an M$EXIT
%SVL_TRTN#	User has issued an M$TRTN

B$EXCRTNF.TYP = UBIN(9). This field has meaning only when B$EXCFR.SUBC is set to %SC_XCONXIT#. See description of %SUB_ECCDELTA.

M$INTRTN - Interrupt Return

The M$INTRTN monitor service allows an Alternate Shared Library to pass break control down to the user level domain. This service is designed for use by Alternate Shared Libraries that have no special use for the Break-key interrupt but wish to establish Break Control (M$INT) to allow a program running under DELTA to be responsive to user requests for DELTA while ensuring that the library itself will not be interrupted while accessing user supplied data.

The Alternate Shared Library relinquishes control with this Monitor Service; i.e., the monitor will perform the return form of the CLIMB instruction for the library. Note that the Alternate Shared Library should have removed the interrupt frame from his TCB (M$TRTN or M$CLRSTK) prior to issuing the M$INTRTN service request.

There is no FPT associated with the M$INTRTN service. The alternate return is taken if the request is made from a domain other than that of the ASL.

The form of the call for this service is:

```
CALL M$INTRTN [ALTRET (label)];
```

M$XCONRTN - Exit Control Return

The M$XCONRTN monitor service allows a Special Shared Processor to defer exit control processing until after all user exit control processing is completed. This service is designed to be used by a Special Shared Processor that has been interrupted for exit control (B$XCON.ECSC=0) for conditions other then those caused by the processor itself; i.e., Operator abort, limit exceeded, QUIT or SAVE commands to the Command Processor, etc.

Prior to issuing the M$XCONRTN request the processor should have removed the exit control frame from its TCB (M$TRTN or M$CLRSTK).

Any exit control limit increments that had been given to the Special Shared Processor will be removed and the user's exit control logic (if any) will be entered with limits incremented for the user if appropriate. When the user's exit control processing is complete the Special Shared Processor will again be entered at its exit control procedure (B$XCON.ECSC=1) with a new set of exit control limit increments.

The M$XCONRTN procedure call is of the form:

CALL M$XCONRTN(FPT_XCONRTN) [ALTRET(label)];

The parameters for this service are as follows:

JERR = VARIABLE Locates the word that contains the error code to be passed to the user. Generally this should be the error code from the Special Shared Processor's exit control frame, i.e. the limit exceeded, operator abort, etc. error code. The default is NIL.

NOENTRY= {YES|NO} Specifies, if YES, that no more entries to the Special Shared Processor are to be allowed prior to entry for deferred exit control processing. The default is NIL.

M$ACCT - Gathering Accounting Statistics on a User

The purpose of M$ACCT is to gather and place in a standard form, accounting statistics on a user.

The M$ACCT monitor service is for use by the CP-6 monitor, CP-6 recovery and Command Processors for the purpose of gathering and presenting in a standard format, accounting statistics on a user at various times throughout his job. The CP-6 monitor uses M$MACCT to produce and place into the *S file an accounting record at each of the following times:

1) At the beginning of each step. This record is produced if (B$JIT.CPFLAGS1 & %CP_STEPACCT#) is set or if B$JIT.SSLEV~=%AZ_NONE#.

2) At the end of each step. This record is produced if (B$JIT.CPFLAGS1 & %CP_STEPACCT#) is set or if B$JIT.SSLEV~=%AZ_NONE#. Both the step-start and the step-end records if used together may be used to produce step statistics on the user.

3) At the beginning of any proprietary accounting period.

4) At the end of any proprietary accounting period. A proprietary accounting period is defined in the CP-6 system as that period of time during which a proprietarily charged processor FETCHed from account :SYS is in control during a job step. A proprietary processor is a processor which has been designated (in it's HEAD record) as such. This may be done by LINK or CONTROL. Due to the effects of that processor calling M$LINK and the subsequent return to the proprietary processor from the linked-to processor, there may be several proprietary-start and proprietary-end records for a proprietary processor during a job step. These proprietary-start and proprietary-stop records may be used to obtain and charge for accounting statistics caused by that proprietary processor. Whenever M$ACCT writes a proprietary-start record, (B$JIT.CPFLAGS1 & %CP_STARPROC#) is set to so indicate. This may be used by a Command Processor as an indication that some proprietary accounting records exist in the *S file.

M$ACCT is also used by CP-6 recovery in order to do accounting for jobs caught in a system SCREECH.

Finally, M$ACCT may be called by Command Processors at JOBEND in order to gather accounting statistics for the running user.

As stated before, M$ACCT when called by the CP-6 monitor will write the accounting information record into the *S file. These records are in the format described further by the CP6_JSP Macro in AZ_MACRO_C and keyed by the ACCT_KEY Macro also in AZ_MACRO_C. Records produced for CP-6 recovery are also presented in CP6_JSP format but are returned to Recovery in a BUFfer specified in the FPT. For Command Processors calling M$ACCT to obtain JOBEND statistics, the record, as always, is in CP6_JSP format. The record itself may be either returned to the Command Processor via the specified BUFfer as for Recovery or, if desired, be written to the *S file for later processing. For information on macros in AZ_MACRO_C, see the System Manager Handbook.

The form of the call for this service is as follows:

CALL M$ACCT (FPT_ACCT) [ALTRET (label)];

The parameters for this service are as follows:

BUF = VARIABLE. Locates a buffer in which to return the information accumulated by this service if STAROUT=NO. The CP6_JSP macro from AZ_MACRO_C should be used to examine this buffer.

JIT = VARIABLE. Locates the JIT to use to gather the information needed for this service if TYP=RCVRY only. For all other TYPs, the caller's JIT will be used.

PNAME = VARIABLE. Locates a VLP_NAME structure containing the name of the processor running during the accounting period this call is to describe. It is used only for TYP = PSTART and SSTART. If unspecified in these cases, JSP.PROCNAME will be empty.

STAROUT = {YES|NO}. If YES, specifies that the JSP record generated is to be written to the *S file. If so, the key is described as %ACCT_KEY macro in AZ_MACRO_C. If NO, the record will be written into the caller's BUF with the data's length written into LEN.

TYP = OPTION. Specifies type of M$ACCT call. Possible values are PSTART, PSTOP, SSTART, SSTOP, JOBEND and RCVRY. This value also is used in the KEY for the record written to the *S file. PSTART and PSTOP are for start and stop of a charge period for a proprietary processor. SSTART and SSTOP are for start and stop of a job step. JOBEND is for use at job end and RCVRY is for use by recovery. JOBEND may be used by Command Processors; all others are for monitor use only.

UTS = VALUE-DEC. Specifies a UTS to be used for current time in all time differencing calculations for TYP=RCVRY only. The value should be the UTS of the crash this recovery is handling. The system UTS is used for all other TYPs.

M$STLPP - Steal Physical Page

The M$STLPP service acquires a physical memory page from the monitor's free page pool without explicitly assigning it to any process. This service can be requested only by a process with Extended MM privilege. The capability to "steal" physical pages from the monitor is a carefully controlled method for obtaining additional memory. Only a limited number of pages can be obtained in this manner (as defined by the STEALPAGES option furnished by the TIGR processor and described in the CP-6 System Support Reference Manual).

Another mode of M$STLPP is available to any user having the Get Physical Page privilege. If this privilege is active in the calling program, then physical pages are allocated directly from the system free page pool with no limit set on the number which can be obtained other than that imposed by the number of free memory pages in the system. Pages allocated in this mode are truly lost to the system; it is entirely the user's responsibility to keep track of their usage. The only method of returning pages acquired in this mode is to call M$RSPP with the Get Physical Page privilege active. This mode of M$STLPP is intended solely for the use of certain system maintenance processors.

If a "stealable" physical page is available, the BASE field of the returned vector will contain the page number of the allocated page. (The M$CVM service must be called in order to access the page.) If no physical pages are available, the returned vector will be VECTOR(NIL).

The form of the call for this service is:

CALL M$STLPP (FPT_STLPP) [ALTRET (label)];

The parameter for M$STLPP is:

RESULTS = VARIABLE specifies the location of a two-word area in which the stolen physical page number is returned, right-justified, in the second word. The default is NIL.

M$RSPP - Release Stolen Page

The M$RSPP service returns stolen physical pages to the monitor. It returns the page to the monitor's free page pool, ensuring first that the returned page was, in fact, a stolen page. The physical page number of the page being released is passed to M$RSPP in the BASE field of the vector in the area named by the RESULTS parameter.

The form of the call for this service is:

CALL M$RSPP (FPT_RSPP) [ALTRET (label)];

The parameter for this service is as follows:

RESULTS = VARIABLE specifies the location of a 2-word area in which the stolen physical page number is passed, right-justified, in the second word. The default is NIL.

M$CVM - Change Virtual Map

The M$CVM service allows a privileged processor or program to examine, display or modify a real physical memory page after mapping it into its working space quarter (WSQ). Users with Special MM privilege can map real pages into the Instruction Segment. Users with Extended MM privilege can map pages anywhere in their virtual space and thus can effect a change in their overall virtual structure unknown to the monitor. M$CVM must be used with exceedingly great care.

A typical case of M$CVM involves mapping a physical memory page onto a previously unallocated virtual page in the user's Instruction Segment. In this case, the WSQFLAG bit in the FPT is off. The number of the specified physical page is placed in the specified Instruction Segment-relative virtual page entry in the user's Page Table. Access to the page is set to read-only for a user with the Special MM Privilege, and to full read/write access for a user with the Extended MM Privilege.

By setting the WSQFLAG, the user (assuming Extended MM Privilege) can map a particular page of physical memory to any previously unassigned virtual page within the user's 512 page WSQ. The virtual page number in this case is interpreted as WSQ-relative, and access to the page will always be set to read/write.

Physical pages made accessible in this way are not considered as being assigned to the user, unless this was initially the case.

Specifying a physical page number of -1 results in effectively removing the specified virtual page from the user's working space. Its access is set to "not in memory", provided that the page was established as a result of an M$CVM operation.
NOTE: M$CVM may be used to change the physical page previously mapped to a virtual page via M$CVM without first freeing the virtual page.

The form of the call for this service is:

CALL M$CVM (FPT_CVM) [ALTRET (label)];

Parameters for M$CVM are:

PPNO = VALUE-DEC(0-n)   specifies the physical page to map from. The default is 0.

VPNO = VALUE-DEC(0-n)   specifies the virtual page to map onto. The default is 0.

WSQ = (0-511). This specifies the working space quarter in which the page is to be mapped. Zero specifies the caller's WSQ; otherwise WSQ should contain a WSQ that was returned on an open of a Virtual Segment.

WSQFLG = {YES|NO} specifies, if YES, that the virtual page number is relative to the start of the caller's working space. If WSQFLG=NO, the virtual page number is relative to the start of the Instruction Segment. The default is NO.

M$SMPRT - Set Memory Protection

The M$SMPRT service sets or resets the Page Table word control field bits for a specified range of virtual pages, based upon the privilege available to the program making the request. The six control flag bits have the following significance:

FLAGS-Binary Value	PTW Bit	Meaning
'100000'	30	Page in memory (read access)
'010000'	31	Page may be written
'001000'	32	Housekeeping page
'000100'	33	Page available for I/O
'000010'	34	Page has been modified (set by hardware)
'000001'	35	Page has been accessed (set by hardware)

For the user without the Extended MM Privileges, this procedure call cannot be used to reduce the amount of protection on a given page or pages from their initial value, and these pages must lie within the user's Instruction Segment (i.e., the WSQ-FLAG bit may not be set, and the virtual page numbers affected are interpreted as Instruction Segment-relative). Thus a user can mark his data pages as "read-only" or even "no access", and can mark his procedure pages "no access". He can also restore these pages to their initial protection values.

In order to affect the other Page Table control bits in any manner, the user program or processor must have the Extended MM Privilege. Such a privileged process has almost complete control over the setting and resetting of the Page Table control information, and one must therefore exercise exceedingly great caution when attempting any such manipulation.

The form of the call for this service is:

CALL M$SMPRT (FPT_SMPRT) [ALTRET (label)];

The parameters for this service are as follows:

FLAGS = VALUE-BIT(6) specifies a 6-bit value that is used to set or reset the Page Table word control field bits. The significance of these flags is shown above. The default is '000000'B.

NUMPGS = VALUE-DEC(0-n) specifies number of pages. The default is 0.

VPNO = VALUE-DEC(0-n) specifies starting page number. The default is 0.

WSQFLG = {YES|NO} specifies, if YES, that VPNO is relative to the work space; if NO, that VPNO is relative to the Instruction Segment. The default is NO.

M$SSC - Set Software Control Flags

The M$SSC service allows a processor with Extended MM privilege to alter the setting of the 10-bit software control flags field in the Page Table for a specified range of virtual pages. Several of these software control flags are reserved for internal MM use only, while the others are available for other software controlled functions, as indicated in the following table.

FLAGS-Binary Value	PTW Bit	Meaning
'1000000000'	18	Page was write-protected for data breakpoint
'0100000000'	19	Page is in recovery dump
'0011111000'	20-24	Reserved
'0000000100'	25	Virtual page is used as FPOOL buffer
'0000000010'	26	Page was established via M$CVM
'0000000001'	27	Page is owned by the user

The form of the call for this service is:

CALL M$SSC (FPT_SSC) [ALTRET (label)];

The parameters for this service are as follows:

FLAGS = VALUE-BIT(10) specifies a 10-bit value that is used to alter the setting of certain bits in the software control flags field. The default is '0000000000'B.

NUMPGS = VALUE-DEC(1-n) specifies number of pages. The default is 0.

SEGFLG = {YES|NO}. Specifies, if YES, that VPNO is relative to the SEGID; if NO, that WSQFLG specifies what VPNO is relative to. The default is NO.

SEGID = VALUE-BIT(12) specifies the 12-bit user's SEGID that the VPNO is relative to if SEGFLG=yes. The default is '0'B.

VPNO = VALUE-DEC(0-n) specifies starting page number. The default is 0.

WSQFLG = {YES|NO} specifies, if YES, that VPNO is relative to the work space; if NO, that VPNO is relative to the Instruction Segment. The default is NO.

M$SAD - Store Access Descriptor

The M$SAD service fills in any of the user's eight special access descriptors with any of the descriptors of the monitor linkage segment, allowing the user to view anything visible to the monitor. The M$SAD service is only available to users with Special or Extended MM privilege. Write access is granted to users with Extended MM privilege. Segment identifiers are available in the file B_SEGIDS_C in the system account.

The form of the call for this service is:

CALL M$SAD (FPT_SAD) [ALTRET (label)];

The parameters for this service are as follows:

MONSID = VALUE-BIT(12) specifies the SEGID of any monitor linkage segment descriptor. The default is '0'B.

USERSID = VALUE-BIT(12) specifies the user's SEGID of the special access descriptor that is to be filled. The default is '0'B.

M$OCMSG - Special Write to Console

The M$OCMSG service is used by command processors to write messages on consoles without getting the user M$KEYIN output template. The procedure call is of the form:

CALL M$OCMSG (FPT_OCMSG) [ALTRET (label)];

For use in KEYIN templates the parameters passed to KEYIN are:

1,T - ACCOUNT FROM JIT
2,T - USER NAME FROM JIT
3,D - MODE FROM JIT (CONSOLE GHOST = 5)
4,D - USER NUMBER
5,D - SUBC FROM M$OCMSG
6,D - FEP NUMBER FROM JIT (IF MODE IS T.S.)
7,4H -PORT NUMBER FROM JIT (IF MODE IS T.S.)
8,T - NODE NAME FOR FEP (IF MODE IS T.S.)
9,D - LEVEL OF PORT INFORMATION (IF MODE IS T.S.)
10,D - TERMINAL TYPE (IF MODE IS T.S.)
11,D - SUB DEVICE (IF MODE IS T.S.)
12,D - SUB SUB DEVICE (IF MODE IS T.S.)
13,T - User-specified text #1
14,T - User-specified text #2
S,D - SYSID FROM JIT

The parameters for this service are as follows:

MSG = VALUE-DEC(0-?) Specifies the output template message number of the message to be printed.

OCTYPE = OPTION Specifies the type of console at which the message is to be printed. The options are as follows:

DEVICE specifies a console which receives messages about devices (e.g., ERROR, MOUNTs) and is allowed to perform device keyins (e.g., RETRY, MOUNT).

SYSTEM specifies a console which receives messages about central site operations (e.g., :SYS,LJS LOGGED ON), and is allowed to perform keyins pertaining to central site operations (e.g., ZAP, START CPU 1).

ADMIN specifies a console that controls a workstation. ADMIN consoles can display and alter (e.g., PRIO, DELETE) only their own workstation's jobs and can control users whose workstation of origin matches this console's WSN.

TAPE specifies console attributes which presently have the same meaning as DEVICE.

DISK specifies console attributes which presently have the same meaning as DEVICE.

UNITREC specifies console attributes which presently have the same meaning as DEVICE.

COMM specifies a console which is able to control and display the status of communications equipment connected to the CP-6 system, in particular the Front-End Processor(s).

ADCOM specifies a console that performs ADMIN and COMM functions.

TP specifies a console that deals with transaction processing related functions. For example, sending messages to TPAs, and receiving messages about TP stations.

The default is ADMIN.

SUBC = VALUE-DEC (0-511) Used to select message modifications of the MSG template.

USERTEXT1 = VARIABLE Locates a text area which contains the 1-31 characters used to fill in the 13th parameter in the output message block.

USERTEXT2 = VARIABLE Locates a text area which contains the 1-31 characters used to fill in the 14th parameter in the output message block.

M$MADMUCK - Read or Change MAD

The MAD (Master Account Directory) is used by the CP-6 system to associate an account with the packset on which it resides. M$MADMUCK may be used to read or update the MAD. Update functions require FMDIAG privilege.

The form of the call for this service is as follows:

CALL M$MADMUCK (FPT_MADMUCK) [ALTRET (label)];

The parameters for this service are as follows:

ACCT = VARIABLE Locates an area containing the account. This area can be generated by invoking the VLP_ACCT macro.

DCB= DCBNAME Specifies, if present, the DCB to use (into which the PSN is returned if no SN is specified). If DCB is not specified, M$* is used.

DELETE = {YES|NO} Specifies that an existing MAD entry is to be deleted. READ occurs before DELETE if both are present.

READ = {YES|NO} Specifies whether a PSN is to be returned to the SN area (if present) or to the DCB specified by the DCB parameter.

SN = VARIABLE Locates an area for the packset name. This area can be generated by invoking the VLP_SN macro.

WRITE = {YES|NO} YES specifies that a new MAD entry is to be made. The alternate return occurs if the entry already exists.

M$IOQ - Input or Output Queueing

The M$IOQ service perform a direct I/O request. It performs its I/O through NIO$QUE and thus is not for T&D I/O. The PR_IOQ# privilege is required to use M$IOQ. The PR_IOQW# privilege is also required for write operations.

The form of the call for this service is:

CALL M$IOQ (FPT_IOQ) [ALTRET (label)];

The parameters for this service are as follows:

BUF = VARIABLE Locates a data buffer or DCW list followed by a data buffer. The default is NIL.

DCB = VALUE-DEC(0-n). Specifies a DCB number for synchronizing I/O. The default is NIL.

DCTX = VALUE-DEC(0-2**15-1). Specifies the DCT index portion of the device logical address. This value is stored in the field DLA.DCTX#. The default is 0.

DRELADDR = VALUE-DEC(0-2**21-1). Specifies the device relative address (e.g., granule number) portion of the device logical address. This value is stored in the field DLA.DRELADDR#. The default is 0.

EVENT = VALUE-DEC(0-n). Specifies the event completion code to be reported when I/O completes. EVENT is used only when WAIT=NO. The default is 0.

FC = VALUE-DEC(0-n). Specifies a logical Function code. The default is 0.

WAIT = {YES|NO} WAIT=YES specifies that the operation is to be completed before control is returned to the user program. If WAIT=YES, the DCB must be specified. WAIT=NO requests the monitor to transfer control to the next user statement after the I/O operation is started. The default is YES.

M$SCREECH - Recovery

The M$SCREECH monitor service allows the CP-6 System Ghosts and Special Shared Processors and users with Extended MM Privilege to cause entry to recovery to create a CP-6 dump file.

The highest Screech Severity allowed to Special Shared Processors is 6 (SUA). Privileged users may only specify a severity of 5 (SNAP).

The M$SCREECH procedure call is of the form:

CALL M$SCREECH(FPT_SCREECH) [ALTRET(label)];

The ALTRET is taken if the user is not authorized to call M$SCREECH or if the Dump Area on the system disk is busy when calling Screech to take a Snapshot Dump.

The parameters for this service are as follows:

JERR = VARIABLE Locates the word that contains the error code to be placed in the JIT. The VLP_ERRCODE macro should be used to generate the errcode. The default is NIL.

SCODE = VARIABLE Locates the doubleword Recovery Code. The Recovery Code contains an identifier to be output on the operator's console. This code also contains bit settings indicating what portions of memory are to be dumped.

The VLP_SCODE macro should be used to generate the recovery code. The default is NIL.

SCREECHID = VARIABLE Locates the area where the TEXTC name of the snapshot dump file is to be returned. This area must be at least 11 bytes in length. The VLP_NAME macro may be used to generate this area. The default is NIL.

VLP_SCREECH

The VLP_SCREECH macro generates a CP-6 recovery code that is passed to the monitor on an M$SCREECH request.

The only difference between the VLP_SCODE and VLP_SCREECH macros is in the defaults for the various options. Please refer the description of VLP_SCODE for the names and meanings of the options. The default for all {YES|NO} options when using the VLP_SCREECH macro is YES.

VLP_SCODE

The VLP_SCODE macro generates the structure describing a standard CP-6 recovery code. This is the code passed to the monitor on a call to M$SCREECH.

The recovery code is either two or four words in length. The four word format is used only when portions of a user (other than the current user) are to be dumped.

The first word contains an identifier in VLP_ERRCODE format. This code will be reported on the operator console by M$SCREECH. The FCG, MID, MON, CODE and SEV options allow initialization of this word.

The second word contains bit settings indicating what portions of memory are to be dumped.

The Monitor Page Table, JIT, HJIT, TSTACK, ISR data and TIGR data are always dumped by default. The LRM, CFUS, ASAVE, ENQ, LDCTS, PPUT, SHJIT, CWSPT, CGDATA, CURCG, PMST, and PMDATA options may be used to specify that other portions of the monitor are to be written to the dump file. These options are generally of no interest to the M$SCREECH user.

The CUSR, CUSRS, IOUSRS, or ALLUSR option may be used to select what users are to be dumped. The SPBUF, FMBUF, ROSEG, ISEG, DDSEG, DBSEG, ASLSEG, and CPSEG options then select what portions of the selected user(s) are to be dumped.

If a selected user (other than the current user) is to be dumped, the AUSER option is used to indicate this. Specifying AUSER=YES causes a four word recovery code to be generated. Word three is to contain the user number of the user to be dumped in bits 0-8. The user number may be specified by the USER=value option. Generally, this user number would not be known at compile time and would be specified at run time by setting fpt_name.USER# to the desired user's number.

The default values for VLP_SCODE are all null. For convenience the recovery code structure may be generated using the VLP_SCREECH macro which has a default SEV of SEV_SCREECH and defaults to everything being dumped. The VLP_SUA macro also generates a recovery code. This macro has the default SEV set to SEV_SUA and defaults to dumping all parts of the current user.

The following options allow initialization of VLP_SCODE:

ALLUSR = {YES|NO}. Specifies that all users are to be dumped. Default=NO.

ASAVE = {YES|NO}. Specifies that the Auto-save Tables (Monitor Data Segment #2) are to be written to the dump file. Default=NO.

ASLSEG = {YES|NO}. Specifies that the Alternate Shared Library Dynamic Data Segments are to be written to the dump file. Default=NO.

AUSER = {YES|NO}. Specifies that the user selected by the USER option is to be dumped. Default=NO.

CFUS = {YES|NO}. Specifies that the CFUs (Monitor Data Segment #1) are to be written to the dump file. Default=NO.

CGDATA = {YES|NO}. Specifies that the Comgroup WSQ Page Tables and context area are to be written to the dump file. Default=NO.

CODE = VALUE-DEC(0-16383). This field contains the number that identifies a particular recovery condition. The file B_SCODE_C contains a list of these CODEs for the monitor.

CPSEG = {YES|NO}. Specifies that the Command Processor Dynamic Data Segments are to be written to the dump file. Default=NO.

CURCG = {YES|NO}. Specifies that the data for the current Comgroup is to be written to the dump file. Default=NO.

CUSR = {YES|NO}. Specifies that only the current user is to be dumped. Redundant if CUSRS or ALLUSR is also specified. Default=NO.

CUSRS = {YES|NO}. Specifies that the current user on all CPUs are to be dumped. Redundant if ALLUSR is specified. Default=NO.

CWSPT = {YES|NO}. Specifies that the Communication WSQ Page Tables are to be written to the dump file. Default=NO.

DBSEG = {YES|NO}. Specifies that the Debugger Dynamic Data Segments are to be written to the dump file. Default=NO.

DDSEG = {YES|NO}. Specifies that the user's Dynamic Data Segments are to be written to the dump file. Default=NO.

ENQ = {YES|NO}. Specifies that the ENQ-DEQ Tables (Monitor Data Segment #3) are to be written to the dump file. Default=NO.

FCG = VALUE-BIT(12). This field contains the two special six-bit characters that identify the functional code group that is initiating recovery. For convenience, it may be specified as text or as a quote string (e.g., FCG=FM).

FMBUF = {YES|NO}. Specifies that the File Management Buffers are to be written to the dump file. Default=NO.

IOUSRS = {YES|NO}. Specifies that all IO Busy users are to be dumped. Redundant if ALLUSR is specified. Default=NO.

ISEG = {YES|NO}. Specifies that the users Instruction Segment is to be written to the dump file. Default=NO.

LDCTS = {YES|NO}. Specifies that the LDCTs, logical device control table. (Monitor Data Segment #8) are to be written to the dump file. Default=NO.

LRM = {YES|NO}. Specifies that low real memory (Pages 0-3) is to be written to the dump file. Default=NO.

MID = VALUE-BIT(6). This field contains the special six bit character that identifies which module in the functional code group is initiating recovery. For convenience, it may be specified as text or as a quote string (e.g., MID='M').

MON = {YES|NO}. Specifies that Recovery was called by the Monitor if set to 1 or by a Ghost or Special Shared Processor if set to 0. Default = YES.

PMDATA = {YES|NO}. Specifies that the Performance Monitor Data is to be written to the dump file. Default=NO.

PMST = {YES|NO}. Specifies that the Performance Monitor State Tables are to be written to the dump file. Default=NO.

PPUT = {YES|NO}. Specifies that the Monitor Page Table Chains (MM_PPUT) are to be written to the dump file. Default=NO.

ROSEG = {YES|NO}. Specifies that the users Read-Only Segment is to be written to the dump file. Default=NO.

SEV = VALUE-DEC(0-7). Specifies the severity of the Recovery code:

- 4 = Zap
- 5 = Snapshot Dump
- 6 = Single User Abort
- 7 = Full Recovery

One of the following should be specified:

```
SEV_ZAP
SEV_SNAP
SEV_SUA
SEV_SCREECH
SEV_FESNAP
```

Default = SEV_SNAP.

SHJIT = {YES|NO}. Specifies that the Monitor Page Table, JIT, HJIT and TSTACK for the slave CPUs are to be written to the dump file. Default=NO.

SPBUF = {YES|NO}. Specifies that the STEP Special Buffers are to be written to the dump file. Default=NO.

USER - VALUE(0-?). Specifies the user number of the user that is to be dumped. This option is ignored if AUSER is not also specified. Default = 0.

VLP_SUA

The VLP_SUA macro generates a CP-6 recovery code that is passed to the monitor on an M$SCREECH request.

The only difference between the VLP_SCODE and VLP_SUA macros is in the defaults for the various options. Please refer the description of VLP_SCODE for the names and meanings of the options. Those options that default to YES when using the VLP_SUA macro are:

```
CUSR
SPBUF
FMBUF
ROSEG
ISEG
DDSEG
DBSEG
ASLSEG
CPSEG
```

M$FEBOOT - Boot Front-End Processor

The M$FEBOOT service is used by privileged processes to re-boot Front-End Processors. The procedure call is of the form:

CALL M$FEBOOT (FPT_FECTL) [ALTRET (label)];

See M$FECTL for a description of parameters for this service.

M$FEDUMP - Dump Front-End Processor

The M$FEDUMP service is used by privileged processes to dump the memory of a Front-End Processor. The procedure call is of the form:

CALL M$FEDUMP (FPT_FECTL) [ALTRET (label)];

See M$FECTL for a description of parameters for this service.

M$FECTL - Control Front-End Processor

The M$FECTL service is used by privileged processes to control FEP operations. The procedure call is of the form:

CALL M$FECTL (FPT_FECTL) [ALTRET (label)];

The parameters for FPT_FECTL are as follows:

BUF = VARIABLE Locates the user's buffer from which the FEP boot image comes on M$FEBOOT or to which the FEP dump is sent on M$FEDUMP.

The default is NIL.

DMPSTART = VALUE-DEC(0-?). Specifies the byte address within FEP memory at which to start the dump on an M$FEDUMP. DMPSTART is unused on other services. The bottom 9 bits are ignored, thus dumps may only begin on multiples of 512 bytes.

The default is 0.

FEP = VALUE-DEC(1-?). Specifies the node number of the FEP to be booted or dumped.

The default is 0.

HALT = {YES|NO}. On M$FECTL YES specifies that the FEP is to be halted, that is, all users logged off, halted and masked.

The default is NO.

MLCPDUMP = {YES|NO}. On an M$FEBOOT request, YES specifies that the FEP MLCPs are to be dumped into the buffer used to boot the FEP. If NO is specified the MLCP dump is not performed.

The default is NO.

NOTIME = {YES|NO}. Specifies whether the FEP is to be timed out in the running state. NOTIME=YES is used for a debug FEP that it is not to be timed out. The default is NO.

RESULTS = VARIABLE Locates a DALIGNED doubleword into which the results of an M$FEBOOT or M$FEDUMP operation are stored. The first word is the error if any that occurred (zero = no error), and the second word contains the size of the memory or MLCP dump performed.

The default is NIL.

START = {YES|NO}. On M$FECTL YES specifies that the FEP is to be started. The FEP is unmasked. If it is not already running, the FEP is re-booted. This bit is inspected after HALT and STOP.

The default is NO.

STOP = {YES|NO}. On M$FECTL YES specifies that the FEP is to be stopped, that is, all users are to be logged off and masked. This bit is inspected after the HALT bit. Note that this leaves the FEP running but not in contact with the host.

The default is NO.

M$CONFIG - Get Channel Configuration

The M$CONFIG service returns the configuration and status of a channel and its controller on a specified FEP.

The form of the call is as follows:

CALL M$CONFIG (FPT_CONFIG) [ALTRET (label)];

Parameters for the M$CONFIG service are as follows:

CHANNEL = VALUE-DEC(0-n) specifies the channel number on which this M$CONFIG call is to report.

Valid channel numbers must be multiples of 128.

The default is 0.

FEP = VALUE-DEC(1-n) specifies the number of the FEP. If FEPNAME is specified, it will override any value specified for FEP.

The default is 0.

FEPNAME = VARIABLE locates a VLP_FEPNAME structure which contains the name of the FEP.

The default is NIL.

RESULTS = VARIABLE locates a VLR_CONFIG structure to contain the results of this M$CONFIG. See VLR_CONFIG for more information about this field.

The default is NIL.

VLR_CONFIG

This macro generates an area which receives the results of M$CONFIG.

CTLR_STATUS = VALUE-DEC(0-n) is set to the status of the controller for the requested channel. The same status values will be used for CTLR_STATUS as for STATUS. The value reported for CTLR_STATUS will be the highest priority status found for all channels on this controller. The priority of the status is equal to its value, i.e. CHNST_ENB# has the highest priority while CHNST_NONE# has the lowest.

DVCID = VALUE-DEC(0-n) is set to the device ID of the requested channel.

EXDVCID = VALUE-DEC(0-n) is set to the extended device ID of the requested channel.

STATUS = VALUE-DEC(0-n) is set to the status of the requested channel. STATUS may take on any of the following values to be found in the INCLUDE file, CP_6_SUBS:

STATUS	VALUE	CHANNEL DESCRIPTION
CHNST_NONE#	0	Does not exist
CHNST_PART#	1	Partitioned for diagnostics
CHNST_DSB#	2	Disabled
CHNST_DIAG#	3	Busy with diagnostics
CHNST_ENB#	4	Enabled

VLP_FEPNAME

The VLP_FEPNAME macro may be used to generate a FEP name.

FEPNAME = VALUE-CHAR(8) specifies a FEP name. The default is blanks, which is equivalent to "no fepname".

M$NODEINFO - Get Node Information

The M$NODEINFO monitor service allows the user to obtain information on a particular node in the network. It returns the current state, the type of node, the node name and number, and routing information for the specified node.

The form of the call is:

CALL M$NODEINFO (FPT_NODEINFO) [ALTRET(label)];

Parameters for the M$NODEINFO service are:

NODEINFO = VARIABLE locates an area where the information is to be returned. This area is described under VLR_NODEINFO.

NODENAME = VALUE-CHAR(8) contains the node name of the node for which the information is requested. If the name is nonblank the node name is used to define the node. If the node name is blank the node number is used. The default is blanks.

NODENUM = VALUE-DEC(0-255) specifies the node number for which the information is requested. This field is used, if the node name is blank, to define the node for which the information is requested.

VLR_NODEINFO

The VLR_NODEINFO macro is used to define an area which will contain the results of the M$NODEINFO monitor service. It contains the information about the node. Parameters for the macro are:

ACTIVE = {YES|NO} is the current state of the node.

ALTQOS = VALUE-DEC(0-255) is a value representing the alternate quality of service. 0 is the highest quality.

CURRENTNODE = {YES|NO} specifies if the node is the node for which the monitor service was issued.

HOST = {YES|NO} YES specifies a host node; NO specifies a FEP.

LINKCONNECTED = {YES|NO} specifies if this node is directly connected by a link.

LINKNODENAME = VALUE-CHAR(8) contains the node name to which messages for the specified FEP are routed.

LINKNODENUM = VALUE-DEC(0-255) is the node number to which messages are routed.

NODENAME = VALUE-CHAR(8). Contains the node name for the node specified on the M$NODEINFO monitor service.

NODENUM = VALUE-DEC(0-255) is the node number of the node.

QOS = VALUE-DEC(0-255) is a value representing the quality of service for the node. 0 is the highest quality.

M$SMOUSE - Start PMME Monitoring

The M$SMOUSE service may be used to initiate the PMME monitoring feature. Extended performance monitor privilege is required.

The M$SMOUSE call is of the form:

CALL M$SMOUSE(FPT_SMOUSE) [ALTRET(label)];

The alternate return will be taken if PMME monitoring cannot be initiated for any reason.

The parameters for the M$SMOUSE service are as follows:

SYSID = VALUE-DEC(0-n) Specifies the sysid of a user to whom the PMME monitoring process is to be restricted. If SYSID equals zero (the default), the PMME monitoring process will monitor all users' PMMEs. If SYSID is non-zero but no user with the indicated sysid exists, PMME monitoring will not be initiated and the altreturn will be taken.

M$XMOUSE - End PMME Monitoring

The M$XMOUSE service may be used to terminate the PMME monitoring feature. Extended performance monitor privilege is required.

The M$XMOUSE call is of the form:

CALL M$XMOUSE [ALTRET(label)];

There is no FPT associated with this service. The alternate return is taken if PMME monitoring cannot be terminated for any reason.

M$GETMOUSE - Get PMME Monitor Data

The M$GETMOUSE service may be used to copy the PMME monitoring routines' data segment from the monitor into a buffer provided by the user.

The M$GETMOUSE procedure call is of the form

CALL M$GETMOUSE(FPT_GETMOUSE) [ALTRET(label)];

The parameter for this service is as follows:

RESULTS = VARIABLE Locates the user's buffer into which the PMME monitoring data is to be transferred. The data area should be word-aligned and at least 1000 words in length.

M$GETPM - Get Performance Monitor Data

The M$GETPM service may be used to obtain a copy of the general system performance monitoring data. This service is intended for use primarily by the STATS processor. Performance monitor privilege is required.

The basic format of data in the M$GETPM results area is simple: multiple frames, each with a 2-word header (frame ID number in the first word, entry count and entry word size in the second). However, the frames vary greatly in size, and proper interpretation of the frame contents requires the use of many of the monitor's BASED data structures, quite a few of which are not available in the :LIBRARY account.

The M$GETPM service call is of the form:

CALL M$GETPM (FPT_GETPM) [ALTRET (label)];

The parameter for this service is as follows:

RESULTS = VARIABLE Locates the user's buffer area into which the performance monitoring data is to be transferred. The data area should be word-aligned and at least 1 page in length.

Section 9

Test and Diagnostic Services

INTRODUCTION

The monitor services in this section are provided for advanced system programmers and require special privilege.

The Test and Diagnostics (T&D) interface provides direct access to devices on a running CP-6 system for preventive and emergency maintenance procedures. The T&D interface permits a hardware diagnostic program to be executed as a user job running in slave mode. The interface allows the user to specify:

- o The device to be tested (device number)
- o The path to that device (IOM number, channel number)
- o The PCW (Peripheral Control Word) and channel; program to be issued to that device.

The T&D tests are classified according to their requirements. Peripheral On-Line Test Systems (POLTS) test a peripheral device and run with the application firmware in the associated Micro-Programmed Controller. Micro-coded Device Routines (MDRs) also test a peripheral device, but reside in an overlay that is called into the associated Micro-Programmed Controller. Isolation Test Routines (ITRs) test the actual MPC and completely replace the application firmware in the MPC. The FEP test is used to perform online testing of a FEP. The test software completely replaces the CP-6 front end software.

Access Restrictions

A series of checks made prior to any T&D input or output ensures that the security of the system is not compromised. Those checks are discussed in the following paragraphs.

T&D Service Access

Diagnostic privilege is required to access the T&D services. In other words, the user's account must have diagnostic privilege; or the diagnostic program must be a processor filed in :SYS, the CP-6 system account. In addition, the operator's permission is required for a program to use the T&D monitor services.

Device Access

Generally T&Ds are permitted only on a peripheral device that is idle and partitioned out of the system. The exceptions are the system console, which is not available for T&D, and shared disks. The availability of shared disks is discussed later. For any device to be available for T&D the TDOK bit in the device's Device Control Table (DCT) must be set.

The two types of tests for peripheral devices, POLTS and MDRs, can be run provided the path to the device is available. A Micro-Programmed Controller loaded with the application firmware can support any number of simultaneous POLTS tests on different devices, or can support one MDR at a time.

The monitor honors a request for an MDR, provided the MPC is not running an ITR or another MDR. The monitor also honors a request for an MDR for a disk drive attached to an MPC which is part of a dual MPC subsystem, provided the other MPC is available to the system (that is, not partitioned out of the system and not running an ITR or another MDR).

A shared disk is any disk device that contains a pack that is a member of a CP-6 packset or does not have a pack mounted at the time of the M$TDOPEN. If the pack on the requested drive is a scratch pack or not a CP-6 pack then the shared disk restriction is not in effect. The diagnostic program is restricted to writing only to the diagnostic cylinders of a shared disk.

MPC Access

The MPC need not be partitioned out of the system to be accessed for Test and Diagnostics. The MPC itself can be tested by an ITR, which completely replaces the application firmware. When an ITR is running, normal I/O and POLTS tests are suspended. The application firmware is immediately restored on completion of the ITR; and normal I/O and POLTS tests are resumed.

The monitor honors a request for an ITR, provided an MDR or another ITR is not running in the MPC. The monitor also honors a request for an ITR for an active MPC which is part of a dual MPC disk subsystem, provided the other MPC is available to the system (that is, not partitioned out and not running another ITR or an MDR).

M$TDOPEN - Open DCB for T&D

The M$TDOPEN service prepares for Test and Diagnostic input or output for a specific peripheral device or for a Micro-Programmed Controller. The open procedure consists of a series of checks. If a device is not known to the system, if it cannot be accessed (see "Access Restrictions" explained previously), if the device is already open for T&D, or if the user does not have diagnostic privilege, the monitor stores an error code and takes the alternate return.

If the operator has not granted permission for Test and Diagnostics, the monitor sends a message to the operator. If the operator grants permission, the procedure continues. If the operator responds negatively, the monitor stores an error code and takes the alternate return.

The monitor sets up the DCB, which includes clearing TYC, CONNCT, CHANTIME, FCN, and FCNF. The device is prepared for Test and Diagnostics at this point.

The form of the call for this service is:

CALL M$TDOPEN (FPT_TDOPEN) [ALTRET (label)];

The parameters for this service are:

CHAIN = VARIABLE locates an area to receive the line printer chain image. The default is NIL.

CHAN = VALUE-DEC(0-63) specifies the channel number of the device to be tested. The default is 0.

DCB = DCBNAME must be specified by the time of the call for this service. The default is NIL.

DVN = VALUE-DEC(0-63) specifies the device number of the device to be tested. Device number zero requests direct access to a Micro-Programmed Controller, rather than a peripheral device associated with the controller. If TEST is equal to FEP then DVN is the FEP number of the FEP to be opened. The default for DVN is 0.

IOM = VALUE-DEC(0-7) specifies the IOM number of the device to be tested. The default is 0.

SPEVENT = VALUE-DEC(0-n) specifies a code to be returned for special interrupts on this device or channel. Each special interrupt causes an event completion notice containing the status word to be sent to the user's event completion entry point. This code will be returned on all coupler interrupts for FEP opens. The default is 0.

TEST = OPTION specifies the type of test to be run: POLTS, MDR, ITR, MTR or FEP. This parameter must be specified by the time of the call for this service. The default is 0. The default causes the monitor to take the alternate return.

VFC = VARIABLE locates an area to receive the line printer VFC image. The default is NIL.

WINDOW = VARIABLE locates an area in the user's memory which is to be mapped into the base of the FEP's working space quarter for FEP opens. This area must begin and end on page boundaries. Word zero of this area will be the coupler's L66 mailbox.

M$TDCLOSE - Close DCB for T&D

The M$TDCLOSE service closes a peripheral device Micro-Programmed Controller or FEP for Test and Diagnostics. The close process waits for any outstanding T&D input or output to complete. The monitor also reloads the application firmware in an MPC if an ITR was running and, if necessary, releases the MPC to perform normal I/O if an ITR or MDR was running.

This service is also called by the job step cleanup routine for each open T&D DCB when a job step terminates.

The form of the call for this service is:

CALL M$TDCLOSE (FPT_TDCLOSE) [ALTRET (label)];

The parameter for this service is:

DCB = DCBNAME must be specified by the time of the call for this service. The default is 0.

M$TDIO - Perform T&D I/O

The M$TDIO service requests that T&D input or output be performed. The DCWLIST parameter supplies the channel program to be performed. The user may also specify an alternate path to access the device. The monitor sets up the I/O request packet and calls the appropriate device scheduler to start the I/O, after certain preliminary checks discussed later. The I/O can be requested with or without a wait. The M$CHECK monitor service may be used to await the completion of a T&D I/O. Or, if WAIT=NO, the monitor takes the normal return at this point.

On completion of the I/O, the DCB is set with the hardware status doubleword, the final LPW, and a completion code. The monitor sends notice of event completion, if requested by the EVENT parameter. If WAIT=YES, the monitor takes the normal return.

The monitor takes the alternate return for the service call if any of the following conditions are detected:

- o The specified DCB is not open for T&D.
- o The access restrictions are not met for the alternate path to the device (the same checks performed by M$TDOPEN occur for the alternate path).

- o The device is a shared disk and the diagnostic program attempts to write on other than the diagnostic cylinders.
- o For ITR and MDR tests, the first IDCW in the DCWLIST is not a suspend command and the suspend flag is not set for the MPC in the DCT entry.

If the T&D DCB is open to an FEP then only the DCB and PCW parameters to M$TDIO are significant. Bits 21-23 of PCW are moved to the corresponding bits of the PCW for the connect on the coupler channel. All other bits of PCW are ignored. The caller must have set up the WINDOW area (see M$TDOPEN) prior to the call to M$TDIO. M$TDIO will always return immediately after issuing the connect to the coupler.

NOTE: During T&D I/O no error recovery is performed. If a fault interrupt occurs for the specified channel during the request, the system fault word is returned and the I/O request is terminated.

The form of the call for this service is:

CALL M$TDIO (FPT_TDIO) [ALTRET (label)];

The parameters for this service are:

CHAN = VALUE-DEC(0-63) specifies the channel number if a path other than the one specified in the DCB is to be used. The default is 0.

DATA = VARIABLE locates a buffer area to be used for T&D input or output. All these data buffers must be in the area framed by FPT_TDIO.DATA_; the entire area must be on a doubleword boundary. If it is not, the hardware reports boundary violations in bits 21-23 of the first hardware status word. All T&D I/O is done in NSA paged mode with LPW address and segmented DCW addressing. The lower bound and size of the payload channel LPW will frame the area specified by FPT_TDIO.DATA_. The default is NIL.

DCB = DCBNAME must be specified by the time of the call for this service. The default is NIL.

DCWLIST = VARIABLE locates an area containing the Data Control Word List. The DCW List must be below the upper bound of the data buffer area but is not necessarily within the data buffer area. All DCW addresses must be in relation to the beginning of the data buffer area. The DCWLIST parameter must be specified by the time of the call for this service. The default is NIL.

EVENT = VALUE-DEC(0-n) specifies the code to be returned on completion of no-wait I/O. If the value is 0 then no event will be reported. The default is 0.

IOM = VALUE-DEC(0-7) specifies the IOM number, if a path other than the one specified in the DCB is to be used. The default is 0./

PCW = VALUE-BIT(36) specifies the first word of the Peripheral Control Word as an octal string. The default is '000000700000'O.

TIMEOUT = VALUE-DEC(0-63). Specifies the number of five second intervals after which this I/O is to be timed out. A value of N means the timeout will occur in (N-1)*5 to N*5 seconds. A value of 0 is interpreted as 64 intervals. The default is 0.

WAIT = {YES|NO} specifies, if YES, that the user is to be blocked until the I/O is complete. The default is YES.

M$WRSYSLOG - Write System Log

The M$WRSYSLOG service allows users with Test and Diagnostic or SYSLOG privilege to place an entry in the system log. If the entry is larger than 1000 words, an error code is returned and the alternate return is taken.

The form of the call for this service is:

CALL M$WRSYSLOG (FPT_WRSYSLOG) [ALTRET (label)];

The parameters for this service are as follows:

ENTRY = VARIABLE specifies the location of an entry to be placed in the system log. The default is NIL.

FILENUM = {ERRLOG|ACCOUNTING|SECLOG|VALUE-DEC(4-99)} specifies the file to which the entry is to be written. File number 1 is ERRLOG, file number 2 is ACCOUNTING, and file number 3 is SECLOG. All other files have the name ELFnn, where nn is the file number. The default is 1 (ERRLOG).

MMINHIB = {YES|NO} specifies, if YES, that the system logging routine is prevented from stealing a page for buffer space on this call. If no room exists on the current buffer page, the error is counted as a lost error. The default is NO.

SIZE = VALUE-DEC(0-500) specifies the size of the entry in words. This size does not include the 4-word header. The default is 0.

TYPE = VALUE-DEC(0-511) is the user-defined type code to be placed in the entry header. The default is 0.

M$RDSYSLOG - Read System Log

The M$RDSYSLOG service returns the first filled or partially filled system log buffer to the area specified by BUF. This service is used only by the ELF ghost to maintain the system log files. Each record in the buffer contains a 5-word header and a log entry; for the format of the header, include the file EL$TABLES and invoke the macro %EL$HDR.

The form of the call for this service is:

CALL M$RDSYSLOG (FPT_RDSYSLOG) [ALTRET (label)];

The parameter for this service is as follows:

BUF = VARIABLE specifies the location of a 512-word buffer that is to receive the system log buffer. The default is NIL.

M$SYSCON - Peripheral Control or Status

This service is used by the SYSCON processor to partition and return peripheral components and to get the current status of a component.

The form of the call for this service is:

CALL M$SYSCON (FPT_SYSCON) [ALTRET(label)];

The parameters for this service are as follows:

CHAN = VALUE-DEC(0-63). Specifies the channel number and is significant only if COMPONENT = CHANNEL or FEP. If COMPONENT=FEP, then CHAN is the FEP number.

COMPONENT = {CHANNEL|DVC|MPC|FEP} Specifies the type of component.

DEVNM = VALUE-CHAR(8). Specifies the name of the component and is significant only if COMPONENT = DVC or MPC.

FLAG = {TDOK|CHECKWRITE} Specifies the flag which is to be set to FLAGVAL. TDOK means the TDOK bit in the corresponding Device Control Table (DCT) or Driver Queue Header (DQH). CHECKWRITE means the CHECKWRITE bit in the corresponding DCT. Note that CHECKWRITE is meaningful for disks only.

FLAGVAL = VALUE-BIT(1). Specifies the value to be placed in the bit specified by FLAG.

FUNCTION = {PARTITION|RETURN|DISPLAY|SETFLAG} Specifies the action to be taken. PARTITION means the component specified is to be partitioned from the system. RETURN means the specified component that has previously been partitioned is to be returned. DISPLAY means return the current status of the component in the RESULT area. SETFLAG means to set the flag specified in FLAG to the value specified in FLAGVAL.

IOM = VALUE-DEC(0-7). Specifies the IOM number and is significant only if COMPONENT = CHANNEL.

RESULT = VARIABLE Locates an area into which the status of the component is to be returned. See VLR_SYSCON macro for description.

VLR_SYSCON

The VLR_SYSCON macro generates an area for the results of the M$SYSCON service. The contents of this area are as follows:

CHECKWRITE = VALUE-BIT(1). Set if CHECKWRITE is turned on for this device. Applies to disk only.

PART = VALUE-BIT(1). Set if component is partitioned.

TDOK = VALUE-BIT(1). Set if T&D is allowed on this component.

M$TDREQCPU - Request CPU for T&D

The Request CPU service allows a user with diagnostic feature authorization to request that a specified processor (CPU) be placed in diagnostic mode, and to specify whether or not execution of the current job step is to be performed by the processor only. Only one user on one processor may be in diagnostic mode at any time, and that processor may not be removed from service by an operator STOP or MAKE key-in.

The form of the call for this service is:

CALL M$TDREQCPU (FPT_TDREQCPU) [ALTRET(label)];

The parameters for this service are as follows:

CPU = VALUE-DEC(0-n). specifies a processor port number. The default is 0.

MODE = {DIAG|NORMAL} DIAG specifies the processor is to be placed in diagnostic mode and NORMAL specifies it is to be released from diagnostic mode. If MODE is not specified, no change is made in the processor mode. At completion of the current job step, the processor mode is reset to normal. If NORMAL is specified, XEQ is ignored. The default is NIL.

TYPE = {L66|DPSE|ELS} TYPE specifies the CPU type on which the user wants to run. If MODE is DIAG and XEQ is NONTDJASSIGN and CPU is zero, then execution is to be performed on a CPU of the specified type for the remainder of the job or session without checking or setting diagnostic mode. The default is L66.

XEQ = {NONTDJASSIGN|JASSIGN|ASSIGN|ANY} ASSIGN specifies that subsequent s ve mode execution during the current job step is to be performed by the specified diagnostic mode processor. JASSIGN specifies that execution is to be performed by the specified diagnostic mode processor for the remainder of the job or session. If MODE is DIAG, NONTDJASSIGN specifies that execution is to be performed on a specific processor or a specific processor type for the remainder of the job or session. Multiple users may be running in NONTDJASSIGN mode at the same time on the same processor or on different processors. If NONTDJASSIGN is specified and CPU is non-zero, then execution is performed on the CPU specified by the CPU field; if CPU is zero, then execution is performed on a processor of the type specified by the TYPE field. ANY specifies that execution may be performed by any processor in diagnostic mode. The default for ASSIGN is ANY.

M$BADPP - Declare Bad Physical Page

The M$BADPP service allows a user with diagnostic privilege to declare a particular physical page as suspected of being bad (e.g., because of a high incidence of reported parity errors). The physical page is removed from normal use as soon as it becomes unused (freed) by whatever CP-6 process may currently own it. It is then placed on the "bad" page list and made available to test and diagnostic programs (see M$GBPL).

Note that it may not be possible for a "suspect" page to be removed from active use during normal CP-6 operation (e.g., it may be part of the monitor or a resident ghost or command program). However, during a system recovery all "suspect" pages are removed from the available page list and placed on the "bad" page list before the monitor is brought back into memory. Thus, recovery from failing memory pages should be possible in all cases except when the failing pages are located in the first few pages of real memory occupied by fault and interrupt locations, the IOM control data, and the AARDVARK boot program.

The form of the call for this service is:

CALL M$BADPP (FPT_BADPP) [ALTRET (label)];

The parameter for M$BADPP is:

PPNO = VALUE-DEC specifies the physical page number of the suspect page. The default is 0.

M$GOODPP - Return Page to Normal Use

The M$GOODPP service allows a user with diagnostic privilege to return to normal use a page which was on the "bad" page list and has presumably been determined to be good by memory test and diagnostics. If the page is marked "still in test" (see M$MPL), an alternate return results.

The form of the call for this service is:

CALL M$GOODPP (FPT_GOODPP) [ALTRET (label)];

Parameters for M$GOODPP are:

PPNO = VALUE-DEC specifies the physical page number of the page which is to be returned to normal use. The default is 0.

M$GBPL - Get Bad Page List

The M$GBPL service allows a user with diagnostic privilege to obtain a list of physical page numbers which have been removed from use by the CP-6 system (i.e., are on an internal "bad" page list). The user specifies a range of physical page numbers within which he is interested in locating any bad pages, as well as the maximum number of bad pages within that range whose page numbers he wants returned. The page numbers, as well as a count of the number actually returned, are passed back to the user in a structure defined by the VLP_PGLIST macro (see the description of VLP_PGLIST in this section).

The form of the call for this service is:

CALL M$GBPL (FPT_GBPL) [ALTRET (label)];

Parameters for M$GBPL are:

HIGHPP = VALUE-DEC specifies the high physical page number of the range to be searched. The default is 0.

LOWPP = VALUE-DEC specifies the low physical page number of the range to be searched. The default is 0.

MAXPGS = VALUE-DEC specifies the maximum number of physical page numbers to be returned. The default is 0.

PGLIST = VARIABLE specifies an area into which M$GBPL returns the list of bad page numbers which it finds, as well as the number of pages that it found. The VLP_PGLIST macro described later in this section is provided to generate the structure for the area. The default is NIL.

M$MPL - Mark Pages as in Test Mode

The M$MPL service allows a user with diagnostic privilege to mark a list of previously obtained "bad" physical pages (see M$GBPL) as in test mode. This prevents other diagnostic users from accessing or releasing to normal use any of the pages so marked.

The form of the call for this service is:

CALL M$MPL (FPT_MPL) [ALTRET (label)];

The parameter for M$MPL is:

PGLIST = VARIABLE specifies the list of physical pages to be marked as in test mode, as well as the number of pages in this list. (The VLP_PGLIST macro described later in this section is provided to generate the structure for this area).

M$UMPL - Remove Pages from Test Mode

The M$UMPL service allows a user with diagnostic privilege to "un-mark" a list of physical pages so that they are no longer in test mode. This allows access to these pages by other diagnostic programs as well as allowing their return to normal service via the M$GOODPP service.

The form of the call for this service is:

CALL M$UMPL (FPT_UMPL) [ALTRET (label)];

The parameter for M$UMPL is:

PGLIST = VARIABLE specifies the list of physical pages to be marked as no longer in test mode, as well as the number of pages in this list. The VLP_PGLIST macro described later in this section is provided to generate the structure for this area.

VLP_PGLIST

The VLP_PGLIST macro creates the list of physical page numbers used by the M$GBPL, M$MPL, and M$UMPL services. It has one parameter, NPAGES, which specifies the maximum number of physical page numbers which may be stored in the structure.

NPAGES = VALUE-DEC(0-n) Specifies the actual number of physical page numbers in VLP_PGLIST.PGLIST (see below). It is set by the M$GBPL service, and may be modified by the user to reflect changes made to the page number list.

PGLIST (0:NPAGES) UBIN(18) Is an array containing the list of page numbers used by the M$GBPL, M$MPL, and M$UMPL services. PGLIST (0) is never used: the first physical page number in the list in VLP_PGLIST.PGLIST(1).

Index

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE	CP-6 HOST MONITOR SERVICES REFERENCE MANUAL VOLUME 1 (DESCRIPTIONS)	ORDER NO.	CE74-00
		DATED	MARCH 1985

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

FROM: NAME ____________ DATE ____________

TITLE ____________

COMPANY ____________

ADDRESS ____________

____

.EASE FOLD AND TAPE–
)TE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell

Together, we can find the answers.

Honeywell

Honeywell Information Systems
U.S.A.: 200 Smith St., MS 486, Waltham, MA 02154
Canada: 155 Gordon Baker Rd., Willowdale, ON M2H 3N7
U.K.: Great West Rd., Brentford, Middlesex TW8 9DH **Italy**: 32 Via Pirelli, 20124 Milano
Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F. **Japan**: 2-2 Kanda Jimbo-cho, Chiyoda-ku, Tokyo
Australia: 124 Walker St., North Sydney, N.S.W. 2060 **S.E. Asia**: Mandarin Plaza, Tsimshatsui East, H.K.

42013, 5C185, Printed in U.S.A.

CE74-00