Programmer's Companion

DOC10045-1XA

Prime™

MIDASPLUS™ PROGRAMMER'S COMPANION

First Edition

by
Ethan Richardson

This document reflects the software as of Master Disk Revision 19.4.

Prime Computer, Inc.
Prime Park
Natick, Massachusetts 01760

COPYRIGHT INFORMATION

PRINTING HISTORY
MIDASPLUS PROGRAMMER'S COMPANION
DOC10045-1XA

Edition	*Date*	*Software Release*
First	April 1986	19.4

CUSTOMER SUPPORT CENTER

Prime provides the following toll-free numbers for customers in the United States needing service:

1-800-322-2838 (within Massachusetts)
1-800-541-8888 (within Alaska)
1-800-651-1313 (within Hawaii)
1-800-343-2320 (within other states)

HOW TO ORDER TECHNICAL DOCUMENTS

Follow the instructions below to obtain a catalog, a price list, and information on placing orders

United States Only

Call Prime Telemarketing,
toll free, at 800-343-2533,
Monday through Friday,
8 30 a m to 8 00 p m (EST)

International

Contact your local Prime
subsidiary or distributor

CREDITS

Editor	Thelma J Henner
Project Support	Anne Marie Fantasia Joan Karp Kathy Normington Michael Rawlings Richard M Walsh Susan Windheim

CONTENTS

Prime Documentation Conventions

The following conventions are used in command formats, statement formats, and in examples throughout this document Examples illustrate the uses of these commands and statements in typical applications Terminal input may be entered in either uppercase or lowercase

Uppercase: In command formats, words in uppercase indicate the actual names of commands, statements, and keywords They can be entered in either uppercase or lowercase

MPLUSCLUP

Lowercase italic: In command formats, words in lowercase italic indicate items for which the user must substitute a suitable value

key-name

Brackets []: Brackets enclose a list of one or more optional items. Choose none or one of these items (0 to 1).

```
[OWNER-IS literal-1]
```

Color: In examples, color indicates required user input.

```
CREATK      Yes
```

Braces {}: Braces enclose a vertical list of items. Choose one and only one of these items.

```
{ SEQUENTIAL }
{ RANDOM     }
{ DYNAMIC    }
```

[RETURN]: Indicates a return key. [RETURN] is used in examples to show that the user presses the RETURN key and nothing else in response to a MIDASPLUS utility prompt.

```
[RETURN]
```

The term *word* in this manual means a 16-bit entity.

MONOSPACE: Indicates system prompts and messages, and interactive dialog.

```
FUNCTION?
```

1

UTILITIES

This chapter contains information concerning the seven utilities that are available to the MIDASPLUS user For each utility there is an explantion of its use, and, in most cases, a dialog of the utility's prompts and user responses Also included is information that the MIDASPLUS user may wish to reference while using a utility For instance, in the KBUILD section you will find a table of the key type codes and the allowable length specifications For more detailed information on the MIDASPLUS utilities see the *MIDASPLUS User's Guide.*

CREATK

The CREATK utility sets up an empty MIDASPLUS file template according to user supplied specifications It also provides you with a way to modify a file, or obtain the template information needed to design an access program

Use CREATK to

- Set up a template and allocate space for a MIDASPLUS file
- Get estimates of how much room is needed for a projected number of files
- Alter an existing MIDASPLUS file
- Obtain information about an existing MIDASPLUS file template and its contents

Interface Requirements

When using CREATK to set up a MIDASPLUS file template you must make sure that the parameters you provide correspond to the interface requirements of the language you intend to use to access the file See the section of this manual that pertains to your language for specific interface requirements

The CREATK Dialog

The following is a sequential list of the CREATK dialog Not all of the prompts appear with any given invocation of CREATK For instance, a response of \O to the NEW FILE? prompt will cause the FUNCTION? prompt to appear, and eliminate the prompts that request specifications for a new file

MINIMUM OPTIONS?

YES, for minimum options

NO, if you wish to change default settings of segment length, index block size, etc

FILE NAME

The pathname of the file to be created, modified, or examined

NEW FILE

YES, to create a new template

NO, to invoke the extended functions option

FUNCTION?

The function you wish to invoke See Table 1-1 below This prompt appears only if NO was entered in response to the preceding prompt Type QUIT to exit

DIRECT ACCESS?

YES, to create a direct access file.

NO, to create a keyed-index file.

PRIMARY KEY TYPE:

With keyed-index file, enter appropriate key-type code (A,B,D,I,L,S). See Table 2.

With direct access file, enter B if the file is for COBOL interface, A if the file is for VRPG interface, or enter the appropriate code (A,B,I,S,L) if the file is for FORTRAN interface. See Table 2.

PRIMARY KEY SIZE = :

The key size. See Table 2.

DATA SIZE IN WORDS = :

If records are fixed-length, enter data length in words (one word=two bytes). If records are variable-length, enter 0 or press [RETURN]. If the MIDASPLUS file will be accessed by a COBOL program, include the total length of all keys in the data size.

NUMBER OF ENTRIES = :

The maximum number of entries (records) you expect to have in the file. This prompt appears only if you are creating a direct access file.

The following three prompts appear only if you entered NO in response to the MINIMUM options prompt For best results, all three block sizes should be the same

FIRST LEVEL INDEX BLOCK SIZE =
The first level block size The recommended block size is 1024

SECOND LEVEL INDEX BLOCK SIZE =
The second level index block size

LAST LEVEL INDEX BLOCK SIZE =
The last level block size

SECONDARY INDEX

INDEX NO ?
A number(1-17) to indicate the secondary index being defined Type 0 or press [RETURN] to end the secondary index definition sequence

DUPLICATE KEYS PERMITTED?
YES or NO

KEY TYPE
The appropriate key data type code (A,B,D,I,L,S) See Table 2

KEY SIZE =
The size in words, bytes, or bits

SECONDARY DATA SIZE IN WORDS =

The number of words of secondary data to be stored with this secondary key, or RETURN to continue The use of secondary data is not recommended

The secondary index prompts repeat, enabling you to enter information about each secondary index To complete the CREATK process, press RETURN at the INDEX NO? prompt

Table 1 lists the possible responses to CREATK'S FUNCTION? prompt and summarizes the resulting action

TABLE 1 CREATK Functions

Response	*Action*
ADD	Add an index
COUNT	Count actual index entries
DATA	Change data record size
EXTEND	Change segment & segment directory length
FILE	Open a new file
HELP	Print this summary
MODIFY	Modify an existing subfile
PRINT	Print descriptor information
QUIT	Exit CREATK
RETURN	Same as QUIT
SIZE	Determine the size of a file
USAGE	Display current index usage
VERSION	MIDASPLUS defaults for this file

Key Types

Table 2 lists appropriate responses to CREATK'S KEY TYPE prompt, and their meanings Your choice must be compatible with the restrictions of the access language you plan to use See the *MIDASPLUS User's Guide* for examples of CREATK dialogs and more complete information concerning the features of CREATK

TABLE 2. MIDASPLUS File Key Types

Key Code and Type		*Length Specifications*
A	ASCII	Words or Bytes: W *nn* or B *nn* Max. 32 words (64 bytes)
B	Bit String	Bits or Words: B *nnn* or W *nn* Max. 16 words (256 bits)
D	Double Precision Floating Point	Hardware-defined: 4 words
I	Short Integer (INT*2)	Hardware-defined: 1 word
L	Long Integer (INT*4)	Hardware-defined: 2 words
S	Single Precision Floating Point	Hardware-defined: 2 words

KBUILD

The primary function of KBUILD is adding large numbers of records to an empty MIDASPLUS file, but KBUILD performs other functions, including

- Adding data to a new (that is, "empty") MIDASPLUS file template
- Adding new data and index entries to an existing MIDASPLUS file that already contains data entries
- Building keyed-index MIDASPLUS files containing either fixed-length or variable-length records
- Building direct access MIDASPLUS files
- Adding entries from an external data file to a new secondary index subfile
- Reformatting a file (adding more keys or transferring data)
- Converting a field from existing MIDASPLUS data subfile records to a secondary key field

Before using KBUILD be sure you know the record structure of your input files and the configuration of your MIDASPLUS file template Use the PRINT function of CREATK to obtain this information

Input Files

You may process up to 99 input files at one time by giving them names that start with the same letters, and end with a 2 digit number that denotes the sequence of the input process For example, the following files would be processed by KBUILD in the order in which they are listed

BRANCH01
BRANCH02
BRANCH03
etc

Multiple input files must all be in the same directory

The KBUILD Dialog

Type KBUILD to invoke the KBUILD utility The following is a summary of the KBUILD dialog and user responses Prompt numbers are added for clarity The numbers do not appear on screen

1 SECONDARIES ONLY?

YES, to build one or more of the secondary index subfiles

NO, to build data and index subfiles The dialog continues at prompt 4

2 USE MIDASPLUS DATA?

YES, to build secondary indexes from records in an existing MIDASPLUS file The dialog continues at prompt 3

NO, to build secondary indexes from a non-MIDASPLUS input file The dialog continues at prompt 4

3 ENTER MIDASPLUS FILENAME

The pathname of the MIDASPLUS files from which the secondary keys will come The dialog continues at prompt 14

4 ENTER INPUT FILENAME

The name of the file to be processed If you are processing multiple files, enter the name of the file with the lowest sequence number

5 ENTER INPUT RECORD LENGTH (WORDS)

The size, in 16-bit words, of the input file record

6 INPUT FILE TYPE

The appropriate KBUILD code See Table 3

7 ENTER NUMBER OF FILES

The number of input files

8. ENTER OUTPUT FILENAME:
   The pathname of the MIDASPLUS file being built.

   If the MIDASPLUS file has fixed-length records, the dialog continues at prompt 13. Otherwise, the dialog continues with the next prompt.

9. THE OUTPUT FILE SELECTED CONTAINS VARIABLE LENGTH DATA RECORDS. IS THE OUTPUT RECORD LENGTH SPECIFIED IN EACH INPUT RECORD AN ASCII STRING OR A BINARY (INT*) STRING? (ENTER A OR B):
   A, if the input file was created by an Editor, COBOL, or VRPG, and the output record length is in ASCII form. The dialog continues with prompt 10.

   B, if the file is BINARY or FTNBIN. The dialog continues with prompt 12.

10. ENTER STARTING CHARACTER POSITION IN INPUT RECORD:
    The character position where the output record length specification begins.

11. ENTER ENDING CHARACTER POSITION IN INPUT RECORD:
    The character position where the output

record length specification ends Dialog continues with prompt 13

12 ENTER STARTING WORD NUMBER IN INPUT RECORD
The *word* number that specifies the output record length for Binary (INT*2) representations

13 ENTER STARTING CHARACTER POSITION, PRIMARY KEY
The starting position of the input record field that contains the primary key

14 SECONDARY KEY NUMBER
The number of the secondary index you are building

15 ENTER STARTING CHARACTER POSITION
The character position in the input record where the secondary key field begins

Prompts 14 and 15 are repeated until you press [RETURN]

16 IS THE FILE SORTED?
YES, if the file is sorted The dialog continues with prompt 17

NO, if the file is unsorted The dialog continues at prompt 19

17 IS THE PRIMARY KEY SORTED?
YES, if the input file is sorted by the primary key field

NO, if the input file is not sorted by the primary key

18 ENTER INDEX NUMBER OF SECONDARY SORT KEY
The secondary key number on which the file is sorted, or press [RETURN] The prompt is repeated until you press [RETURN]

19 ENTER LOG/ERROR FILE NAME
A file name in which to store errors and KBUILD statistics

If you press [RETURN] the information is sent to your terminal, but is not recorded

20 ENTER MILESTONE COUNT
The frequency at which you want the statistics displayed and recorded in a log/error file

If you enter 0, milestones are printed for the first and last records of the input file only

KBUILD Supported Input File Types

Table 3 lists KBUILD's file type codes and their descriptions

TABLE 3 KBUILD-Supported Input File Types

File Type	*Description*
TEXT	A file containing ASCII data that has been created by an editor
BINARY	A binary file created by PRWF$$ which is usually called from a FORTRAN program
COBOL	An uncompressed ASCII file created by a COBOL WRITE statement
FTNBIN	A binary file created by a FORTRAN WRITE statement via the routine 0$BD07
RPG	An uncompressed output file created by a VRPG program

Kiddel

The KIDDEL utility offers the fastest way to delete or zero out a MIDASPLUS file DELETE gets rid of the entire index subfile, ZERO deletes only the entries and unused space, but leaves the empty subfile By responding to the appropriate KIDDEL prompt you may

- Delete an entire MIDASPLUS file, or delete one or more of the secondary index subfiles
- Delete work files (called *junk files*) left over from an aborted MPACK run
- Zero out all entries from one or more secondary index subfiles
- Zero out an entire MIDASPLUS file, leaving an empty template

The KIDDEL Dialog

Type **KIDDEL** to invoke the KIDDEL utility The following dialog shows the KIDDEL prompts and appropriate user responses

`FILE NAME`

The name of the MIDASPLUS file to be used

DELETE INDEXES

One of the following

Secondary index subfile numbers, separated by commas, indicating the indexes you want to delete

ALL, to kill the entire file, delete the file from its UFD, and return you to PRIMOS

JUNK, to delete work area information left over after an aborted MPACK operation

NONE, to get you to the next prompt without deleting any index subfiles

ZERO INDEXES

This prompt appears only if you entered NONE to the above prompt Enter one of the following

Secondary index subfile numbers, separated by commas, indicating the indexes you want to zero out

ALL, to zero all index subfiles and the data subfile, or to reinitialize direct access files

NONE, to return you to PRIMOS

Note

If you want to delete or zero the primary index, use the ALL response Do not enter 0 (primary index) to either the DELETE INDEXES or ZERO INDEXES prompts

See the *MIDASPLUS User's Guide* for more complete information

MPACK

The MPACK utility is used to recover data record space marked for deletion, to increase file efficiency, to unlock records, and to restructure index subfiles Further, MPACK must be run on a file whose template has been modified by the ADD, DATA, EXTEND, or MODIFY options of CREATK before the changes are actualized

The MPACK functions are

- Reclaiming the space that deleted records occupy
- Packing indexes to minimize disk space used

- Reordering the data subfile to match the order of primary index subfile entries (complete file restructure)
- Logging errors and milestones to keep tabs on errors and to monitor the ongoing operation
- Unlocking any of the data records left locked on disk after a program abort or failure

The MPACK Dialog

The MPACK dialog and the appropriate responses are shown below Prompt numbers are added for clarity

1 ENTER MIDASPLUS FILENAME
The pathname of the existing MIDASPLUS file

2 'MPACK' or 'UNLOCK'
MPACK or UNLOCK If you enter UNLOCK, the dialog continues at prompt 6

3 ENTER LIST OF INDEXES ALL OR DATA
Index numbers separated by commas or spaces, in order to pack index subfiles The dialog continues at prompt 6

ALL, to restructure all indexes in the file and unlock all data records The dialog continues at prompt 6

DATA, to restructure the data file and all indexes

4 OK TO OVERWRITE FILE?
YES or NO This prompt is given only if you gave the DATA response to the previous prompt

5 NEW FILENAME
The name that you want MPACK to give to the restructured file This prompt appears only if you answered NO to prompt 4

6 ERR/LOG FILE?
An optional error/log filename for errors and milestone counts Press [RETURN] if you do not want an error/log file

7 MILESTONE COUNT?
The appropriate number of records after which a milestone report will be generated

MDUMP

The MDUMP utility dumps a MIDASPLUS file into a sequential disk file Once you run MDUMP on a MIDASPLUS file, you can

- Edit the resulting sequential file, if the data is in ASCII format
- Edit the MDUMP output file and use it as an input file to KBUILD in order to build a new MIDASPLUS file
- Examine the resulting sequential file to check a MIDASPLUS file's data records and key values

MDUMP also reports any damage that it finds in the MIDASPLUS file being dumped, and therefore can be a means of validating the integrity of an index

MDUMP Options

MDUMP prompts you for the order, contents, and format of the dump The prompts also ask you how and where MDUMP should record the status and error information generated by the dump The following options are available

- Order of the dump
- Contents of the dump
- Format of the dump

- Log/error file
- Milestone recording

The MDUMP Dialog

The following list shows the MDUMP dialog, appropriate responses, and the effect of the responses:

ENTER TREENAME OF MIDAS FILE TO DUMP:
The pathname of the MIDASPLUS file to be dumped.

ENTER DUMP METHOD ('DATA' OR AN INDEX #):
DATA, to dump records in the order of the data subfile records, or an index number (0 to 17) to dump records in order found in the specified index.

DO YOU WANT THE DATA RECORD DUMPED?
YES, to dump the data record.

NO, to dump index keys only.

DO YOU WANT THE PRIMARY INDEX KEY DUMPED?
YES, to dump the primary index keys. The key value will be appended to the data record if data record is also being dumped.

NO, if you do not want to dump the primary key

DO YOU WANT THE INDEX <#> KEY DUMPED?

YES, to dump the specified index.

NO, if you do not want to dump the specified index

The above prompt appears only if you specify a secondary index in response to the second prompt

ENTER OUTPUT FILE TREENAME

A filename for the output file

ENTER OUTPUT FILE FORMAT

BINARY, COBOL, RPG, or TEXT If you are unsure, press [RETURN] or type HELP to get a list of these options

ENTER LOG/ERROR FILE NAME

The name of the file to be opened for recording errors and statistics If this file already exists, it will be overwritten

Press [RETURN] if you do not want to open a log/error file

ENTER MILESTONE COUNT

A number to indicate how often you want the milestone report to appear Enter 0 for the briefest version of the status report

See the *MIDASPLUS User's Guide* for a more complete explanation of MDUMP'S features

MPLUSCLUP

MPLUSCLUP cleans up MIDASPLUS segment directories and subfiles, releases locks held in memory, and cleans up and re-initializes per-user information The -ALL option cleans up system information MPLUSCLUP also releases record locks that are recorded in main memory and reports if any record locks are recorded on disk (Use the MPACK utility to release locks recorded on disk)

Use MPLUSCLUP when you or another user receive a fatal error or are force-logged out and automatic cleanup has not occurred

MPLUSCLUP Options

If you do not specify any options, MPLUSCLUP will affect your own files only. The system administrator can clean up other user's files from the system console in the debug mode, by using the following options:

```
MPLUSCLUP { -USER usernumber }
          { -ALL             }
```

See the *MIDASPLUS User's Guide* for more complete information.

SPY

SPY is a menu-driven utility that provides the user with information used and updated by MIDASPLUS during runtime. This information includes:

- A table of data locks taken
- System-wide statistics on performance and use of the system
- System-wide configurable parameters
- User-specific configurable parameters

To invoke the SPY utility, enter the name SPY and then choose the appropriate option from the menu The three categories of information available are

- DATA RECORD LOCKS
- SYSTEM STATISTICS
- SYSTEM CONFIGURATION

Depending upon your request, a second menu may appear requiring an additional choice See the *MIDASPLUS User's Guide* for more complete information

2

THE FORTRAN INTERFACE

This section contains information concerning FORTRAN call sequences to MIDASPLUS, flags, and the communication array. Since MIDASPLUS was originally written for FORTRAN, no special restrictions apply other than the restrictions inherent to MIDASPLUS itself, such as the limit of 17 secondary keys. The FORTRAN subroutines can also be called from programs written in PL/I-G or in C.

Insert the following files at the start of every FORTRAN program that accesses a MIDASPLUS file.

$INSERT SYSCOM>PARM K INS FTN

$INSERT SYSCOM>KEYS INS FTN

Subroutine Call Sequences

The following list shows the FORTRAN subroutines and their call sequences

Subroutine	*Call Sequence*
ADD1$, DELET$, FIND$, LOCK$, NEXT$, UPDAT$	(funit, buffer, key, array, flags, altrtn, index, file-no, bufsiz, keysiz)
CLOSM$	(funit, status)
GDAT$	(unit, flags, buffer, bufsiz, status)
NTFYM$	(key, unit, status)
OPENM$	(key, pathname, namlen, funit, status)

Flag Use

Table 4 shows in which subroutines the various flags may be used.

TABLE 4 Flag Use

Flag	ADD1$	FIND$	NEXT$	LOCK$	UPDAT$	DELET$	GDATA$
FL$USE	x	x	x	x	x(R)	x	
FL$RET	x	x	x(R)	x(R)			
FL$KEY	x	x	x	x	x		
FL$BIT		x	x				
FL$PLW		x	x				
FL$UKY		x	x				
FL$SEC		x	x				
FL$ULK					x		
FL$FST		x	x				x(R1)
FL$NXT		x	x				x(Rs)
FL$PRE			x				

(R)-Required (R1)-Required for first record
(Rs)-Required for all records after the first

MIDASPLUS Flag Meanings

The flag settings affect FORTRAN subroutine calls. For example, if you included the following program line in your program

```
FLAGS=FL$RET+FL$USE
```

and then included *FLAGS* in the proper place in the call sequence, the two flags *FL$RET* and *FL$USE* would be ON, effecting the execution of your program in the manner listed below. All other flags would be considered OFF.

The flag settings and how they affect a FORTRAN subroutine call are described below.

FL$USE

ON: Use current copy of array.

OFF: Do not use current copy of array.

FL$RET

ON: Return entire array for use on subsequent calls.

OFF: Return completion code only, in array(1).

FL$KEY

ON Return primary key in data record buffer on calls to FIND$, NEXT$ and LOCK$

Do not make a copy of the primary key to store in data record on calls to ADD1$ Use only if primary key is first field in data record

OFF Store a copy of primary key in each data subfile record on calls to ADD1$

Do not add primary key to the beginning of the data buffer on calls to FIND$, NEXT$ and LOCK$

FL$BIT

ON Interpret key size as bits if key is a bit string, or in bytes if key is ASCII

OFF Interpret key size as words (default)

FL$PLW

ON Position to next index entry greater than or equal to current or user-supplied entry

OFF Not in use

FL$UKY

ON Update user-supplied key field with version stored in the file Useful in partial key searches

OFF Do not update user-supplied key field

FL$SEC

ON Return secondary data instead of data record

OFF Return data record read from data subfile

FL$ULK

ON Unlock data entry only - do not update it

OFF Update data entry and unlock it

FL$FST

ON Position to first index entry in subfile

OFF Position to first entry that matches current entry or user-supplied key value

FL$NXT

ON Position to next index entry greater than current entry or user-supplied key value

OFF Position to next index entry that matches current entry or user-supplied key value

FL$PRE

ON Position to previous index entry

OFF Do not position to previous entry

Precedence of Conflicting Flags

Certain flags have effects that conflict with each other In the case where conflicting flags are set, the priority level is as follows

- FL$FST
- FL$NXT
- FL$PLW

Communications Array

During keyed-index accessing, only the first word of the communication array may be modified If the word is set to -1, MIDASPLUS ignores the contents of the array If the word is set to 0 or 1, MIDASPLUS uses the contents of the array

During direct accessing, the user must

modify the first four words of the array, as outlined by Table 5.

TABLE 5. Direct Access Array Format

Words	*Setting*	*Meaning*
1	0 or 1	Use array contents, which are supplied by the user
2	entry size (in words)	Primary key length in words, plus data record length in words, plus 2 words. Supplied by user.
3-4	record number	A single-precision (REAL*4) floating-point record number. Supplied by user.
5-14		Hash value, based on current key value. Supplied by system.

3

THE COBOL INTERFACE

The COBOL interface to MIDASPLUS is based on the standard COBOL I/O statements for INDEXED and RELATIVE files. The following is a list of MIDASPLUS terms and their COBOL equivalents:

MIDASPLUS	*COBOL*
Keyed-Index file	INDEXED file
Direct Access file	RELATIVE file

MIDASPLUS	*COBOL*
Primary Key (indexed files)	RECORD KEY
Primary Key (relative files)	RELATIVE KEY
Secondary Key	ALTERNATE RECORD KEY

See the *COBOL 74 Reference Guide* or the *COBOL Reference Guide* for detailed information on COBOL syntax and concepts. See the *MIDASPLUS User's Guide* for differences between the CBL compiler and the COBOL compiler.

Language-Dependent Restrictions

COBOL places the following limitations on MIDASPLUS files used in COBOL applications

- Up to 17 secondary keys are supported per file (INDEXED only).
- Relative files do not support primary

keys A relative key must be defined in the working-storage section

- The primary key and all secondary keys (if any) must be included in the data record
- Secondary keys should not be embedded in the primary key
- The only key types that COBOL supports are ASCII (A) and bit string (B)
- The maximum ASCII key size is 64 characters
- The maximum bit string key size is 32 characters
- For RELATIVE files, the primary key is always defined as being from 8 bits to 48 bits in length This allows for a maximum of 999,999 entries in the file
- Secondary data is not supported
- The COBOL compiler does not support variable length records A COBOL program is able to READ a variable length record, but not WRITE it
- The COBOL record size must match the data size defined for the file during CREATK

Opening and Closing a MIDASPLUS File

To open a MIDASPLUS file from a COBOL program follow the standard COBOL file I/O procedures. The basic format for the SELECT statement for an INDEXED file is as follows:

SELECT *filename*

ASSIGN TO PFMS

ORGANIZATION IS INDEXED

[ACCESS MODE IS { SEQUENTIAL / RANDOM / DYNAMIC }]

RECORD KEY IS *key-name-1*

[ALTERNATE RECORD KEY
IS *key-name-2*
[WITH DUPLICATES]...]

[FILE STATUS IS *status-code*].

The basic format for the SELECT statement for a RELATIVE file is as follows

```
SELECT filename

    ASSIGN TO PFMS

    ORGANIZATION IS RELATIVE

[   ACCESS MODE IS { SEQUENTIAL }  ]
                   { RANDOM     }
                   { DYNAMIC    }

    [RELATIVE KEY IS key-name-1] *

    [FILE STATUS IS status-code]
```

* optional only if sequential access

See the *COBOL 74 User's Guide* for DATA DIVISION requirements

To open a MIDASPLUS file use the OPEN statement You can open more than one file, but each file name specified in an OPEN statement must appear in a SELECT and ASSIGN statement and must be described with an FD entry in the DATA DIVISION The format is

```
OPEN { INPUT  }
     { I-O    }
     { OUTPUT }
       filename-1[, filename-2]
```

To close a MIDASPLUS file from a COBOL program use

CLOSE *filename-1* [, *filename-2*]...

Table 6 shows what statements can be used in each access mode.

TABLE 6. Statements Permitted in Each Access Path

File Access Mode	*Statements*	*Open Mode*			
		INPUT	OUTPUT	I-O	
SEQUENTIAL	READ	x		x	*
	WRITE		x	x	**
	REWRITE			x	
	START	x		x	
	DELETE			x	
RANDOM	READ	x		x	*
	WRITE		x	x	
	REWRITE			x	
	START				
	DELETE			x	
DYNAMIC	READ	x		x	*
	WRITE		x	x	
	REWRITE			x	
	START	x		x	
	DELETE			x	

* Records, are locked. ** Indexed files only.

Error Handling

One of the following three clauses for handling runtime errors must be specified for each I/O verb

- AT END directs control to an *end-of-file* procedure if a logical end of the file has been reached during a sequential read The format is as follows

  AT END *imperative-statement*

- INVALID KEY specifies the procedure to be executed when an error occurs during keyed operations It tranfers program control to a designated procedure or performs some useful action or series of actions The format is as follows

  INVALID KEY *imperative-statement*

- USE AFTER specifies a program procedure that is executed if an error occurs and the INVALID KEY or AT END clause are not present

Put the USE AFTER statement in the DECLARATIVES section of the program The format is as follows

USE AFTER STANDARD

$$\left\{ \begin{array}{l} \text{EXCEPTION} \\ \text{ERROR} \end{array} \right\}$$

PROCEDURE ON

$$\left\{ \begin{array}{l} \textbf{filename} \\ \text{INPUT} \\ \text{OUTPUT} \\ \text{I-O} \end{array} \right\}$$

Refer to the *COBOL 74 Reference Guide* for complete details on error handlers

File Positioning

The START Statement

The START statement positions the file pointer to a specific record in a file opened for SEQUENTIAL or DYNAMIC access Use a MOVE statement to assign a value to *key-name* before implementing START

The format is as follows:

START filename

$$[\text{KEY IS } \left\{ \begin{array}{l} \text{GREATER THAN} \\ \text{NOT LESS THAN} \\ \text{EQUAL TO} \end{array} \right\} \text{key-name}]$$

[INVALID KEY *imperative-statement*].

For indexed files *key-name* can be the primary key, a secondary key, or a part of either type of key. For relative files *key-name* must be the relative key.

The READ Statement

Sequential Reads: A sequential READ can be executed on a file opened in the SEQUENTIAL or DYNAMIC mode. A sequential READ to a DYNAMIC file requires the NEXT option. Position the file pointer to the desired starting location by a START, OPEN, or keyed READ. The format for a sequential READ is as follows:

READ *filename* [NEXT] RECORD
[INTO *read-var*]
[AT END *imperative-statement*].

Keyed Reads: A keyed read can be executed on a file opened in the RANDOM or DYNAMIC mode To perform keyed reads on an Indexed file, first specify the key by using a MOVE statement to put the value into *key-name* and then use the following READ statement

READ *filename* RECORD
[INTO *read-var*] [KEY IS *key-name*]
[INVALID KEY *imperative-statement*]

The KEY IS clause is not used for Relative files Instead, the key value being searched for must be moved into the relative key field

Adding Records

If a file's access mode is RANDOM or DYNAMIC, you may write to it when it is opened for OUTPUT or I-O Be sure to put the appropriate value in the RECORD KEY or ALTERNATE KEY before executing a WRITE You may write to a RELATIVE SEQUENTIAL file only when it is open for OUTPUT For a RELATIVE SEQUENTIAL file do not supply the relative key value It is returned to you after each write

The WRITE statement format is:

WRITE *record-name* [FROM *from-area*]
[INVALID KEY *imperative-statement*].

Rewriting Records

The COBOL programming language updates MIDASPLUS records by rewriting the entire record. You can change any field with the exception of the RECORD KEY. Only the current record may be updated, so be sure to position the file pointer and lock the record first with a READ. The file must be open for I-O. The REWRITE statement format is:

REWRITE *record-name* [FROM *from-area*]
[INVALID KEY *imperative-statement*].

Deleting Records

To delete a record in a RANDOM or DYNAMIC file, move the primary key to the RECORD KEY (or RELATIVE KEY if the file is a relative file) before executing the DELETE statement With a SEQUENTIAL file, first position the file pointer with a READ before executing the DELETE statement The file must be opened for I-O in order to delete entries from it The DELETE statement format is

DELETE *filename* RECORD
[INVALID KEY *imperative-statement*]

The INVALID KEY clause is not used for SEQUENTIAL files

4

THE BASIC/VM INTERFACE TO MIDASPLUS

The BASIC/VM interface to MIDASPLUS consists of a special set of file handling statements that are similar in format to the standard file handling statements of BASIC/VM. They are

- ADD
- DEFINE FILE
- POSITION
- READ

- REMOVE
- REWIND
- UPDATE

The READ, POSITION, and DELETE statements all automatically lock a record before performing an operation

Language-Dependent Restrictions

When creating a MIDASPLUS file to be accessed by a BASIC/VM program be sure the template conforms to the following restrictions

- Only keyed-index files are allowed
- Do not use more than 17 secondary indexes per file Duplicate index entries are allowed
- Do not add secondary data

Although key fields are not required to be part of the data record, it is strongly recommended that you include them

Opening/Closing MIDASPLUS Files

BASIC/VM uses the same methods of opening and closing a MIDASPLUS file as it uses for any other type of file - the DEFINE FILE and the CLOSE statements Use the following format to open a file

DEFINE [READ] FILE *#unit* = *filename*, MIDAS [, *record-size*]

The parameters and their uses are

READ
: Opens file for reading only

#unit
: User supplied unit number from 1 to 12

filename
: Name of the MIDASPLUS file, expressed as either a BASIC string variable that contains a file name or a quoted string constant

record-size
: Length of the MIDASPLUS data subfile in words, where one word equals two characters If the file has variable length records, omit *record-size*

MIDASPLUS files are closed in the same way as any other BASIC/VM file

CLOSE *#unit*

Error Handling

The standard BASIC/VM ON ERROR statement traps any MIDASPLUS errors occurring during runtime Use the following format

ON ERROR [*#unit*] GOTO *line-number*

Use the following PRINT statement to print out the MIDASPLUS error code

PRINT MIDASERR

File Positioning

There are two BASIC/VM statements that specifically move the file pointer, the POSITION statement, and the REWIND statement The following is the format for the POSITON statement

POSITION #*unit*,

$$\left\{\begin{array}{l}\text{SEQ}\\ \text{KEY [}\textit{key-number}\text{]} = \textit{key-value}\\ \text{SAMEKEY}\end{array}\right\}$$

The parameters and their uses are

#unit
: File unit number, previously assigned by DEFINE.

key-number
: Number from 0 to 17 that specifies the key. If zero or unspecified, the primary key is used.

key-value
: Key value enclosed in quotes, or a legal string expression.

SEQ
: Advances the pointer to the next record in the file, according to the order of the current index.

SAMEKEY
: Advances the file pointer through a series of records with duplicate key values.

Note

If there is no record at the specified file position, an error is flagged

Use the following to reset the file pointer to the lowest value of a specified key

REWIND *#unit* [, KEY *num-expr*]

The parameters and their uses are

#unit
: File unit number, previously assigned by DEFINE

num-expr
: Key (index subfile number) whose subfile pointer is to be rewound

Adding Records

The ADD statement adds a record to a MIDASPLUS file without changing the current file position or the current record Although only the primary key value is required in an ADD, one or more secondary key values may be added to the appropriate index subfiles with a single ADD The format of the ADD statement is

ADD *#unit, new-record,*

{ PRIMKEY
{ KEY[0-*expr*] = key0-*val* [,*keylist*] }

Where *keylist* has the form:

Key key-number=key-val ...

The parameters and their uses are

#unit
File unit number, previously assigned by DEFINE.

new-record
Record to be added. Length must equal record length declared for the file.

PRIMKEY
Primary key.

KEY[0-*expr*]
A literal or numeric expression that evaluates to zero.

key0-val
Primary key value. May be a literal or numeric expression.

Reading Records

The BASIC/VM READ statement allows you to read records from a MIDASPLUS file sequentially, by duplicate keys, and by primary or secondary key values. READ locks the record until another operation is performed to change the file pointer location.

READ [KEY] #*unit*
{ SEQ
,[KEY [*key-num*] = *key-val*] ,*readvar*
SAMEKEY }

The parameters and their uses are

KEY
: The KEY option instructs MIDASPLUS to return a full key value in *readvar* during a partial key search.

#unit
: File unit number, previously assigned by DEFINE.

key-num
: Literal or numeric expresion indicating key (index subfile) to be used in the read.

key-val
: Full or partial value on which to conduct the search.

SEQ

Advances the pointer to the next record in the file, according to the order of the current index.

SAMEKEY

Advances the file pointer through a series of records with duplicate key values.

read-var

String variable that the retrieved record is read into.

Updating Records

The UPDATE statement replaces the current record with a new record value. UPDATE does not change any of the index subfile entries. To change key values you must first delete the record, and then add it to the file along with the new key values. The UPDATE format is:

```
UPDATE #unit, new-record
```

The parameters and their uses are

#unit

File unit number, previously assigned by DEFINE.

new-record
New data record value

Deleting Records

The REMOVE statement selectively deletes index entries for a particular data record. If you specify only the primary key, the associated data record and the primary index entries are deleted. In this case, the secondary key entries are not deleted until they are used to reference the now deleted data record, or until MPACK is run on the file. The REMOVE format is:

REMOVE #*unit*
[,KEY [*key-num*] = *key-val*]
[,KEY [*key-num*]=*key-val*]...

The parameters and their uses are

#*unit*
File unit number, previously assigned by DEFINE.

key-num
Numeric variable containing optional key number Primary key is the default value.

key-val

String expression indicating the key entry to be deleted

Note

REMOVE without any options deletes the current key

5

THE PL/I SUBSET G INTERFACE

This section documents the PL/I Subset G interface to MIDASPLUS files PL/I-G views a MIDASPLUS file as a RECORD KEYED SEQUENTIAL file that the standard PL/I-G READ, WRITE, REWRITE, and DELETE statements can access

For more complete information concerning the PL/I-G interface to a MIDASPLUS file, see the *MIDASPLUS User' Guide* For information concerning programming in PL/I-G, see the *PL/I Subset G Reference Guide*

Language-Dependent Restrictions

The PL/I-G interface to MIDASPLUS does not support the following MIDASPLUS features:

- Non-ASCII primary keys
- Secondary keys
- Direct access MIDASPLUS files
- Secondary data

If you want to set up a MIDASPLUS file with secondary keys, fixed-length records, or a primary key of less then 32 characters, use CREATK. You will still be able to access the file with PL/I-G, but you will not be able to use PL/I-G to access the secondary index subfiles.

Opening/Creating A MIDASPLUS File From PL/I-G

To create a MIDASPLUS file from a PL/I-G program, use

DECLARE *filename* FILE
KEYED SEQUENTIAL,
OPEN FILE (*filename*) OUTPUT,

To open an existing MIDASPLUS file use

OPEN FILE (*filename*) { OUTPUT | INPUT | UPDATE }

The parameters and their uses are

Filename
: The name of file being opened, not longer than 8 characters The file should be declared as KEYED SEQUENTIAL

INPUT
: File access mode is "read only"

OUTPUT
: File access mode is "write only" Primarily used to open and create new files, but may also be used to make additions to existing KEYED SEQUENTIAL files

UPDATE
: File access mode permits READ, WRITE, UPDATE, and DELETE operations.

Table 7 shows which PL/I-G statements are allowed by the various access modes.

TABLE 7. Access Mode Statements

Access Mode	*Valid PL/I-G Statements*
INPUT	READ
OUTPUT	WRITE
UPDATE	READ WRITE REWRITE DELETE

You can open a file while declaring it by including one of the I/O modes in the DECLARE statement, for example:

```
DCL filename FILE KEYED
 SEQUENTIAL INPUT;
```

Adding Records

To add records to a new or existing file, open the file for OUTPUT or UPDATE. Use the following to WRITE records to a MIDASPLUS file.

WRITE FILE(*filename*) FROM(*var*)
KEYFROM(*keyvar*);

The parameters and their uses are

var

A variable, declared as type character, containing the new record information.

keyvar

A variable containing a unique value which can be either a character string or numeric field, and which will be the primary key. If *character string*, it cannot be VARYING, and cannot be more than 32 characters.

Reading a MIDASPLUS File

There are three types of file reads that PL/I-G can perform on a MIDASPLUS file: a *keyed read*, which returns a record based on a user-supplied key; a *sequential read* in which records are read in primary key order; a *key-value read*, in which the full value of the primary key is returned. The key-value read can only be done during a non-keyed read. Open the file for INPUT or UPDATE in order to read it. The READ statement format is:

READ FILE (*filename*) INTO(*var*)
[KEY(*keyvar*)]
[KEYTO (*curkey*)];

The parameters and their uses are

var
: A variable into which the file is read.

keyvar
: The primary key of the record to be read. If omitted, next sequential record is read.

curkey
: A variable in which the current key is returned. Declare *curkey* as CHAR(32) VARYING.

Updating File Records

Records in an existing MIDASPLUS file can be updated with the REWRITE statement The record is locked during a REWRITE and is kept locked until another I/O operation is performed The current record position is not updated Use the following format

REWRITE FILE(*filename*)
FROM(*datavar*) [KEY(*keyvar*)],

The parameters and their uses are

datavar

A variable containing the data that will replace the record to be updated

keyvar

The primary key of the record to be read If omitted, the current record is updated

Deleting Records

To delete a record from a MIDASPLUS file, you must open the file for UPDATE. Use the following format:

DELETE FILE(*filename*) KEY(*keyvar*) ;

The parameters and their uses are

keyvar
: The primary key of the record to be deleted. If omitted, the current record is deleted.

6

THE VRPG INTERFACE

Prime's VRPG supports both **keyed-index** and direct access MIDASPLUS files. In VRPG, keyed-index MIDASPLUS files are called Indexed files. Column 32 of the FILE DESCRIPTION Statement indicates the file organization:

- I = indexed file
- D = direct access file

Language-Dependent Restrictions

The following rules apply for using VRPG with MIDASPLUS files

- All keys in an indexed file must be in the data record
- The maximum length of the primary key of an indexed file is 32 characters
- The primary key of a direct access file must be type ASCII
- A VRPG program can update only the data field
- A Delete operation (DEL on output) may be performed on indexed files only

Describing a MIDASPLUS File in VRPG

Table 8 lists the VRPG Description Specifications for MIDASPLUS files Table 9 lists the MIDASPLUS specific fields in the other VRPG statements See the *RPG II V-Mode Compiler Reference Guide* for more complete details of these statements

TABLE 8 VRPG File Description Specifications for MIDASPLUS Files

Attributes	*Column(s)*	*What to Specify*
File Type	15	I = Input O = Output U = Update
File Designation	16	P = Primary S = Secondary C = Chained D = Demand Blank = Output
File Format	19	F = Fixed-length
Record Length	24-27	The record length, including the Primary key length
Mode of Processing	28	R = random by key, relative record number, or ADDROUT file

Attributes	*Column(s)*	*What to Specify*
		L = sequential within limits, or by record address file. Blank = sequential by key or consecutive (direct access).
Key-Field Length	29-30	Number of characters in primary index - for Indexed files only. (Length from 1-32)
File Organization	32	I = Indexed D = Direct
Key Col. Position	35-38	Column where the primary key starts in the data record - for indexed files only.
Device	40-46	DISK (Required)

Attributes	*Column(s)*	*What to Specify*
File Addition	66	A = Add records to a non-empty Indexed file. U = Add unordered records to an empty Indexed file. Specify O in column 15. Blank = Add ordered records to an empty Indexed file. Specify O in column 15.

TABLE 9. VRPG Fields For Other Statements

Statement	*Column(s)*	*What to Specify*
Calculation	28-32	Operation: SETLL READ CHAIN
Calculation	54-55	Indicator on for record not found
Extension	11-18	Name of the separate sequential limits file for the RAF method
Extension	19-26	Name of the MIDASPLUS file to be processed
Input	61-62	Matching fields or chaining fields

Statement	*Column(s)*	*What to Specify*
Output Specifications	16-18	ADD = Add records to an indexed file DEL = Delete records (Blank = Add records to a direct file)

7

THE OFFLINE CREATE ROUTINES

This section discusses KX$CRE and KX$RFC, which are user-callable, offline routines that serve as an alternative to CREATK. You will find KX$CRE and KX$RFC are helpful only if you need to create file templates or read a file configuration from within your application. It is recommended that you use command files that invoke CREATK rather than using KX$CRE or KX$RFC.

KX$CRE

You can use KX$CRE to create a MIDASPLUS file from a program It is the same routine used by CREATK The calling sequence is

CALL KX$CRE (*filnam,namlen,flags,alloc, pridef,secdef,errcod*)

The parameters, data types, and their uses are

filnam (INT*2)
: The pathname of the file to be opened, two characters per word

namlen (INT*2)
: Length of *filnam* in characters

flags (INT*2)
: Global flags M$DACC enables direct access, M$NRNW sets the file lock to *n* readers and *n* writers (required)

alloc (REAL*4)
: The number of data records to preallocate if file is direct access Use only if M$DACC flag is set Set to 0 for keyed index file

pridef(6) (INT*2)

Definition array for the primary index See Table 10, below

secdef(6,17) (INT*2)

Definition array for the 17 secondary indexes See Table 10, below

errcod(2) (INT*2)

Error code returned by MIDASPLUS in *errcod(1)* *errcod(2)* contains an index number if applicable

The Pridef and Secdef Arrays

Assign values to *pridef* and *secdef* to indicate the characteristics of the primary index and any secondary indexes that will be included in the file template The six-element, one-dimensional array *pridef* (1 6) contains the necessary information to define the primary index The two-dimensional *secdef* array defines the secondary indexes *Secdef*(1 6,1) defines secondary index "1" All of the elements in these arrays are INTEGER*2

Table 10, below, lists the six elements of each array.

TABLE 10. Pridef and Secdef Array Elements

Pridef	*Secdef*	*Description*
(1)	(1,i)	Contains one or more flag values specifying the key type and size. For *secdef*, it also determines the duplicate status of the key. See Table 11, below, for key-type flags.
(2)	(2,i)	States the primary key size in bits, bytes, or words (*pridef*) or the secondary key size in bits, bytes, or words (*secdef*). A *0* in *secdef* indicates that the index does not exist.
(3)		Data record size. Supply a *0* for variable- length records.
	(3,i)	Secondary data size. Supply a *0* if you do not want this feature.

Pridef	*Secdef*	*Description*
(4)	(4,i)	Level 1 block size. Supply a 0 to use the default block size (1024 words).
(5)	(5,1)	Level 2 block size. Supply a 0 to use the default block size (1024 words).
(6)	(6,i)	Last level block size. Supply a 0 to use the default block size (1024 words).

Table 11, below, lists the flags used for the first element of each of *pridef* and *secdef* arrays:

TABLE 11. Flags for pridef(1) and secdef(1)

Flag Type	*Flag Value*	*Meaning*
Index-specific	M$DUPP	Duplicates permitted for this key (secondaries only)

Flag Type	*Flag Value*	*Meaning*
Key	M$BSTR	Bit string
Key	M$SPFP	Single-precision floating point (REAL*4)
Key	M$DPFP	Double-precision floating point (REAL*8)
Key	M$SINT	Short (16 bits) integer (INT*2)
Key	M$LINT	Long (32 bits) integer (INT*4)
Key	M$ASTR	ASCII string
Key size	M$BIT	Key length is specified in bits
Key size	M$BYTE	Key length is specified in bytes
Key size	M$WORD	Key length is specified in words

KX$RFC

KX$RFC is a user-callable routine that returns the file configuration of an already existing MIDASPLUS file. The KX$RFC calling sequence is:

CALL KX$RFC (*filnam,namlen,flags, alloc,pridef,secdef,errcod*)

The parameters, data types, and their uses are

filnam (INT*2)
: Pathname of the MIDASPLUS file whose configuration is to be returned. User supplied.

namlen (INT*2)
: Length of *filnam* in characters. User supplied.

flags (INT*2)
: Global flags: M$DACC is returned for direct access files; 0 for keyed-index files.

alloc (REAL*4)
: Number of data records pre-allocated if direct access is enabled. Returned by KX$RFC.

pridef (INT*2)
Definition array for the primary index Returned by KX$RFC

secdef (INT*2)
Definition array for the 17 secondary indexes *Secdef*(1 6,1) contains the definition for secondary index 1 Returned by KX$RFC

errcod (INT*2)
Error code or 0 if no error Returned by KX$RFC

KX$RFC Arguments

You do not supply the *flags* argument in a call to KX$RFC, a successsful call to the KX$RFC subroutine returns it If the file is a direct access file, the flag M$DACC is returned, otherwise, *flags* is returned as 0 If the file is not enabled for direct access, KX$RFC zeroes the argument

Pridef and *Secdef* Flags The flags returned on this call are

M$BSTR, M$MSPFP, M$SINT, M$LINT, and M$ASTR
Possible settings for bits 1-4

M$DUPP
: The setting of bit 6

MBIT, MBYTE, and M$WORD
: Possible values of bits 6-7 (KX$RFC does not usually return M$WORD.)

Element *secdef*(2,i) is returned as 0 if there is no index "i" in this file. See Table 11 for flag meanings.

8

THE OFFLINE BUILD ROUTINES

This section lists the three routines that can be used to populate a MIDASPLUS file. Never use OPENM$, NTFYM$, or CLOSM$ with these routines. For more complete information on the restrictions and use of the offline routines see the *MIDASPLUS User's Guide*.

When using these routines be sure to $INSERT the following files in your FORTRAN program

```
SYSCOM>PARM K INS FTN
SYSCOM>ERRD INS FTN
SYSCOM>KEYS> INS FTN
```

PRIBLD

PRIBLD builds a primary index subfile and a corresponding data subfile from an input file sorted by the primary key The MIDASPLUS (output) file must be empty PRIBLD's calling sequence is

CALL PRIBLD (*seqflg,primkey,data, dlength,unit,altrtn,danum*)

The parameters, data types and their uses are

seqflg (INT*2)
: The event sequence flag

primkey (INT*2)
: The primary key value to use on this call.

data (INT*2)
: Data to be added

dlength (INT*2)
Length of the data

unit (INT*2)
File unit number.

altrtn (INT*2)
Alternate return statement number.

danum (REAL*4)
Entry number for direct access files

SECBLD

SECBLD builds secondary index subfiles from input data that is sorted by secondary key The secondary index subfile must be empty, and a copy of the primary key must be included as one of the arguments of the call The SECBLD calling sequence is

CALL SECBLD (*seqflg,seckey,pkey,index, secdat,sdsiz,unit,altrtn*)

The parameters, data types and their uses are

seqflg (INT*2)
The event sequence flag

seckey (INT*2)
The secondary key value to be added to the index subfile.

pkey (INT*2)
The corresponding primary key.

index (INT*2)
The secondary index subfile number.

secd at (INT*2)
The secondary data to be stored in this index entry. Specify zero if not applicable.

sdsiz (INT*2)
The size of secondary index data, in words. Specify zero if not applicable.

unit (INT*2)
File unit number.

altrtn (INT*2)
Alternate return statement number.

BILD$R

BILD$R builds a data subfile and primary index subfile,or a secondary index subfile

CALL BILD$R (*seqflg,key,pbuf,bufsiz, danum,index,unit,altrtn*)

The parameters, data types, and their uses **are**

seqfl (INT*2)
: The event sequence flag

key (INT*2)
: The primary or secondary key value to be added to the index subfile

pbuf (INT*2)
: Data subfile entry, primary key, or primary key and secondary data, depending on the action to be taken on this call to BILD$R

bufsiz (INT*2)
: Size of *pbuf* in words Set to zero if adding fixed length date entry Set to length of data entry if adding variable length data entry Set to length of primary key length plus secondary data length if adding secondary data

danum (REAL*4)
Record entry number for direct access files Specify zero if file is keyed-index, or if you are adding a secondary index entry

index (INT*2)
The number of the index subfile being built Specify zero for the primary index

unit (INT*2)
File unit number

altrtn (INT*2)
Alternate return statement number

Note

When adding a primary index entry, the *bufsiz* argument is ignored unless the file contains variable length records In this case, *bufsiz* represents the length of the data record only When adding a secondary index entry, supply a 0 if you have already put the desired secondary data into *pbuf* If non-zero, *bufsiz* indicates the total size of the primary key and the size of the secondary data supplied in *pbuf* Extra secondary data is ignored and insufficient data is padded with zeros

9

ERROR MESSAGES

This chapter lists the error messages for the MIDASPLUS utilities and offline routines, MIDASPLUS runtime error codes, and the COBOL status codes. If you find yourself with an error that does not seem to be explained by the list of MIDASPLUS error codes, it may be a COBOL status code (if your access program is written in COBOL), or it may be a PRIMOS error code. See *MIDASPLUS User's Guide* for a list of PRIMOS error codes.

KBUILD Error Messages

The following are KBUILD runtime error messages If an error is fatal, KBUILD aborts after reporting it Although files are sometimes damaged in fatal errors, the files are usually still usable A non-fatal error is a warning only and does not harm the KBUILD process The record that causes the warning message, however, is not added to the file

UNABLE TO REACH BOTTOM INDEX LEVEL
: The last level index block could not be located, file is damaged (Fatal)

FILE IN USE
: The file is not available for KBUILD use KBUILD must have exclusive access to the file You are returned to PRIMOS (Fatal)

INDEX 0 FULL -- INPUT TERMINATED
: If the maximum number of entries in primary index is exceeded, KBUILD aborts (Fatal, but file is still okay)

INDEX index-no FULL -- NO MORE ENTRIES WILL BE ADDED TO IT
: If the maximum number of entries in the secondary index is exceeded, KBUILD aborts Building of other indexes continues (Fatal, but file is still okay)

INDEX 0 FULL -- REMAINING RECORDS WILL BE DELETED

Data records are added to the subfile first, in the order read in from the input file Then the primary index entries are added, in sorted order, to point to them KBUILD ran out of room in the primary index when trying to add entries to point to those already in the data subfile KBUILD is forced to set the delete bit on in data subfile entries whose primary keys do not fit in the primary index (Fatal, but file is still okay)

INVALID DIRECT ACCESS ENTRY NUMBER -- RECORD NOT ADDED

The user-supplied direct access record number is an ASCII string, but it is not legitimate if it contains non-numeric characters Also, the entry number may be less than or equal to 0, may not be a whole number or may exceed the number of records allocated (Non-fatal)

INVALID OUTPUT DATA RECORD LENGTH -- RECORD NOT ADDED

The output record length is an invalid ASCII string that is, it contains non-numeric characters Also, the size specified might exceed the input record size (Non-fatal)

THIS INDEX IS NOT EMPTY. EITHER ZERO THE INDEX OR DO NOT SPECIFY THIS KEY AS SORTED.

.sk KBUILD cannot add sorted data entries to any index subfile that already contains entries. Do not specify the sorted option during the KBUILD dialog. (Non-fatal)

CAN'T FIND PRIMARY KEY IN INDEX -- RECORD NOT ADDED

This error occurs when adding secondary index entries to an already populated file. The primary key value that you supplied in the input file was not found in the primary index. (Non-fatal)

INDEX 0: INVALID KEY -- RECORD NOT ADDED

This error could occur if the input file is sorted and an entry was out of order, or if a duplicate key value appeared for an index that does not allow duplicates. (Non-fatal)

INDEX index-no: KEY SEQUENCE ERROR -- RECORD NOT ADDED

A duplicate value was discovered for the primary key or for a secondary key that does not allow duplicates. (Non-fatal)

MDUMP Error Messages

When MDUMP dumps a file, errors that it finds are reported along with the milestone statistics The following are MDUMP's error messages and their meanings

BAD DATA RECORD POINTER - IGNORED
: MDUMP found a bad data record pointer in the MIDASPLUS file The dump continues

BAD INDEX BLOCK OR INDEX BLOCK POINTER
: MDUMP found an incorrect index block or index block pointer in the MIDASPLUS file The dump halts

UNABLE TO REACH BOTTOM INDEX LEVEL
: MDUMP found an incorrect index block or index block pointer before dumping any records The dump does not occur

INDEX BLOCK SIZE GREATER THAN MAXIMUM DEFAULT
: MDUMP found an index block larger than the maximum default size The dump halts

Kiddel Error Messages

FILE IN USE
: The file is not available for KIDDEL use KIDDEL must have exclusive access to the file You are returned to PRIMOS

SPY Errors

Internal system errors and user input errors are the only kind of errors that can occur during the execution of SPY Internal system errors are fatal User input errors can usually be trapped since only specific input choices are allowed

If you make a detectable error when entering a menu option, you are given two more chances to enter a valid choice and then SPY stops If an *out-of-range* or otherwise detectable invalid entry is made for user number or filename, you are given two more chances before SPY gives up

If you request that SPY report the number of locks on a file and the SPY_FNAME configuration is OFF (the default), the following error message appears

```
The SPY_FNAME configuration is OFF
for MIDASPLUS  SPY cannot display
locks by FILENAME  See your system
administrator if you wish to have
the SPY_FNAME configuration changed
Hit RETURN to Continue
```

See the *MIDASPLUS User's Guide* for additional information about SPY_FNAME and the configurations

MPACK Error Messages

The following are MPACK error messages If an error is fatal, MPACK aborts after reporting it A non-fatal error is a warning only and does not harm the MPACK process

Fatal Messages

`UNABLE TO REACH BOTTOM INDEX LEVEL`
: MPACK was unable to find a last level index block for an index The file is damaged Use MDUMP to dump the data file into a sequential disk file and use KBUILD to rebuild the file

DATA SUBFILE FULL

This message may appear if MPACK is used to implement segment subfiles or segment directories that are smaller than the default Use the EXTEND option of CREATK to enlarge the subfile size or segment directory length

INDEX FULL

This message may appear if MPACK is used to implement index subfiles or segment directories that are smaller than the default There is no more room in the index subfile. Use the EXTEND option of CREATK to enlarge the subfile size or segment directory length.

ABORTING MPACK

This message appears when a fatal error occurs Use the JUNK option of KIDDEL to delete the scratch files created by MPACK, or if you are not overwriting the old file, delete the new file.

FILE IN USE

This file is not available for MPACK use MPACK must have exclusive access to the file. You are returned to PRIMOS.

Warning Messages

INDEX SUBFILE DOES NOT EXIST

You supplied an index number that was not defined for this file

FILE ALREADY EXISTS -- TRY AGAIN

You specified the name of an existing file in response to ENTER NEW MIDASPLUS FILE NAME? prompt of the DATA option path MPACK does not overwrite an existing file in this case You must enter the name of a non-existent file

INVALID KEY SEEN (IGNORED)

A key is out of order in the index or the key is a duplicate and duplicates are not allowed in the specified index

INVALID DIRECT ACCESS ENTRY NUMBER SEEN (IGNORED)

A record number is not greater than zero, or is not a whole number, or is greater than the pre-allocated record number limit

KX$CRE Error Messages

Errors occurring during the building of a template could originate in the file system or in MIDASPLUS Errors can result from invalid user arguments or an internal MIDASPLUS problem This section lists the most common KX$CRE error codes

ME$BAS
: Allocation size is invalid The number specified in *alloc* was either less than 1 0, not a whole number, or too big to allocate the number passed in the user supplied *alloc* argument due to the default segment directory and segment subfile lengths

ME$BDS
: Data size is invalid because the data size is negative, or the data size specified in *prdef*(3) indicates variable-length data records, but the file is configured for direct access, and thus requires fixed-length data records

ME$BKS
: Key size is invalid For example, the key size is too big, the key size is negative, or the primary key size is 0 (The limit is

16 words except for ASCII strings, which may be up to 32 words)

ME$BKT
: Key type is invalid

ME$BL1
: Level 1 block size is invalid The block size must be positive, not larger than 1024 words, and must hold at least two index entries

ME$BL2
: Level 2 block size is invalid

ME$BLL
: Last level block size is invalid When building a secondary index, this error may also occur when the secondary data size, *secdef* (3,1), is too large (in comparison to the block size) to fit the mandatory two entries per block

ME$NDA
: No duplicates are allowed You specified the flag M$DUPP in *pridef*(1) Duplicates are never allowed for the primary key

PRIBLD, SECBLD, and BILD$R Error Messages

This section lists the PRIBLD, SECBLD, and BILD$R error messages If you get one of the following error messages, you can call the routine to finish building the index (Set *seqflg* to 2) Your file will be built except for the problem that the error code noted SECBLD and BILD$R replace the symbols ## (used in the error messages below) with a secondary index number

If a file system error, which is not listed below, occurs, your file may be damaged If this happens, use KIDDEL to zero your file Try to figure out what happened, and try again

CAN'T USE PRIBLD AND BILD$R SIMULTANEOUSLY

You added one or more entries to the primary index with BILD$R and then called PRIBLD Simultaneous access to the primary index subfile is not allowed Either continue adding entries or finish building index 0 with the appropriate calls to BILD$R, but not PRIBLD

ILLEGAL SEQFLG

INDEX ## ILLEGAL SEQFLG

If either one of the two above messages appears, the value of *seqflg* is incorrect The first call to the routine to add an entry must have a *seqflg* of 0, which the routine returns as a 1 Later calls to the routine to add additional entries must continue to have a *seqflg* of 1 The final call to the routine to finish building index 0 for that MIDASPLUS file must have a *seqflg* of 2, which the routine returns as a 3

NOT A VALID MIDASPLUS FILE

INDEX ## NOT A VALID MIDASPLUS FILE

The first time that the routine is called to add an entry (*seqflg* = 0) to the primary or secondary index of a MIDASPLUS file, the routine calls KX$RFC to verify that the file is a valid MIDASPLUS file and to gather certain configuration data needed to build the file.

INDEX 0 INDEX BLOCK SIZE GREATER THAN MAXIMUM DEFAULT

INDEX ## INDEX BLOCK SIZE GREATER THAN MAXIMUM DEFAULT

The above two messages indicate that a fatal error may have occurred on the first call to the routine that adds an entry to a file created with the extended options path The index block size was defined as larger than the default block size of 1024 words Recreate the file

KEY SEQUENCE ERROR

The key provided in the current call is less than or equal to the key provided in the previous call to PRIBLD

INDEX 0 ±0 nnnnnnn E±nn INVALID DIRECT ACCESS ENTRY NUMBER

This error occurs during direct access file processing only It can happen for one of three reasons

1 The record number supplied was less than zero

2 The record number supplied was not a whole number

3 The supplied number exceeds the number of entries preallocated by CREATK You may have changed this number with CREATK without performing MPACK on the file to effect the change Use

MPACK against the file before restarting

DATA SUBFILE FULL

INDEX 0 DATA SUBFILE FULL

If either one of the above two messages appears, no more entries may be added to the data subfile and, therefore, to the primary index Call the routine to finish the primary index (with *seqflg* = 2) with the entries already added

INDEX ## DOES NOT EXIST

The indicated index is either an invalid index number or does not exist in the MIDASPLUS file Either go back, ADD the index with CREATK, and try again, or remove all references to this index from the program

INDEX ## CAN'T USE SECBLD AND BILD$R SIMULTANEOUSLY

You may have added one or more entries to this index with BILD$R and have now called SECBLD to add an entry to it You may continue adding entries or finish building this index with the appropriate calls to BILD$R, but not to SECBLD

INDEX ## NOT ZEROED

The index cannot contain any entries if you are trying to use PRIBLD or SECBLD to build it Use KIDDEL to zero this index or to zero the entire file

INDEX ## KEY SEQUENCE ERROR

The supplied key is less than the key supplied in the last call to the routine for this index, or the secondary key is a duplicate of the secondary key supplied in the last call to the routine for this index, and the index does not allow duplicates

INDEX ## CAN'T FIND PRIMARY KEY IN FILE

The routine was unable to find the key value supplied for the *pbuf* or *pkey* argument The primary index subfile does not contain this value

INDEX ## INDEX FULL

You may not add any more entries to this particular secondary index, but you may still call the routine (set *seqflg* to 2) to finish the index

INDEX ## CAN'T USE BILD$R AND PRIBLD/SECBLD SIMULTANEOUSLY

This error message appears when you have added one or more entries to this

index with PRIBLD or SECBLD and then called BILD$R to add an entry to the same index You can either continue adding entries or make the appropriate calls to either PRIBLD or SECBLD (but not to BILD$R) to finish building this index

INDEX 0 DIRECT ACCESS FILE - INDEX OF -1 AND ENTRY # REQUIRED FOR PRIMARY KEY
You attempted to build the primary index of a direct access file Use an index number of -1 (not 0), supply a REAL*4 entry number in *danum*

Runtime Error Codes

The following is a list and explanation of the MIDASPLUS runtime error codes The error codes are returned directly to you unless error traps are included in your program If an error is not listed, see the section, PRIMOS Error Messages, below If you are using COBOL, check to see if the error is a MIDASPLUS error or a file status error (COBOL status codes are listed later in this chapter)

Miscellaneous Error Codes

Code	*Explanation*
1	Duplicates exist for the current key
7	The sought-after entry does not exist in the file
10	Locking was requested on a record that is already locked COBOL status code is 90 Check your ACLs and make sure that the READ/WRITE locks are set correctly The record might not be locked, you could lack the necessary ACLs
11	The data record does not have the *locked* bit set when it should This happens when a user attempts an update without first locking the record
12	Duplicate keys are not allowed
13	An unrecoverable concurrency error has occurred For example, another user deleted your current entry
19	The disk is full

READ/WRITE Error Codes

Error codes in the 20 range are usually READ/WRITE errors. Try to do a LOGPRT which creates a LOGLST. This will tell you if there are any unrecoverable errors on disk or memory parity errors. (See the *System Operator's Guide* for additional information about LOGPRT.) The problem could be a hardware problem.

Make sure that the CREATK template is the same size as the FD in COBOL or the buffer size in FORTRAN

Code	*Explanation*
20	An error was encountered while writing a record or index block
21	An error was encountered while reading a data record or index block When using FORTRAN, it is necessary to use K$GETU rather than hard-coded file units so that MIDASPLUS can monitor the file units that are open
22	A file system error was encountered while attempting to get a file unit or the internal file unit table is full When using FORTRAN, it is necessary to use K$GETU rather than hard-coded file units so that MIDASPLUS can monitor the file units that are open
23	The unit is not open as a segment directory
28	You attempted to write to a read-only file

Programming Error Codes

Error codes in the 30 range are usually programming errors for example, reading past the end of a file Check your program if you receive an error code in this range

Code	*Explanation*
30	You did not ask for the array to be returned when it must be returned. Set FL$RET in flags on the call.
31	The array must be supplied but was not Set the flag FL$USE
32	You supplied a bad length (for example, an invalid record length) or the index supplied is invalid
33	You supplied an invalid array
34	The use of NEXT$ is not allowed in direct access files
35	You cannot do an indexed add to a direct access file
36	You set FL$USE in flags but the current array involved a different index from the one that you supplied in this call

Internal Error Codes

Error codes in the 40 range are usually internal errors If you receive a 40 or 41 error message, run MPLUSCLUP -ALL from the system console See the *MIDASPLUS User's Guide* If the problem persists, reshare MIDASPLUS

If you receive a 42-45 error, there are corrupted pointers in the file Run MPACK with the DATA option See the *MIDASPLUS User's Guide* If the problem persists run MDUMP and KBUILD to restructure the file

Code	*Explanation*
40	Fatal internal error
41	Timeout occurred while waiting for buffers
42	There is no offspring pointer or no next block found The index tree is corrupted
44	You attempted to access an indexed file as direct access or the file is direct access and the entry was not found
45	Got a data record but expected an index block

Additional Error Codes

Code	*Explanation*
51	Invalid index pointer in index entry For example, segment number is 0
71	An error occurred while attempting to delete an entry from a direct access file
85	The index or data subfiles are full Use the EXTEND option of CREATK to extend the subfile Increase the subfile in a multiple of 512,000 Use MPACK with the DATA option on the file after the EXTEND to restructure the indexes and the data subfiles
92	Network error
10001	You supplied an invalid parameter on an OPEN or CLOSE
10002	The MIDASPLUS internal tables are full

Code	*Explanation*
10003	Not a segment directory.
10004	Fatal internal error. Contact Prime Customer Service.
10005	Error opening remote file.

COBOL STATUS CODES

The following section lists the COBOL status codes and the equivalent MIDASPLUS error codes and states whether the status codes appear with INDEXED (I) and/or RELATIVE (R) files.

Status Code	*MPLUS Code*	*File Type*	*Interpretation*
00			Successful completion of the operation.

Status Code	*MPLUS Code*	*File Type*	*Interpretation*
10	**7**	I, R	The end of file was reached on a READ operation. The file pointer is positioned past the logical end of the file.
22	**12**	I, R	An attempt was made to perform a WRITE or RE-WRITE that would create a duplicate primary key entry. Duplicate primary key values are illegal.
23	**7**	I, R	The record was not found on an unsuccessful key search. There is no record in the file with this key value.
30	19/20	I, R	An error parity such as quota exceeded or disk full.

Status Code	*MPLUS Code*	*File Type*	*Interpretation*
90	10	I, R	The record is already locked Another user or process has already locked this record for update
91	11	I, R	The record is not locked A REWRITE operation was attempted without first locking the record with a READ operation
92	12	I	An attempt was made to add a duplicate secondary key value to a secondary index subfile that does not permit duplicates

Status Code	*MPLUS Code*	*File Type*	*Interpretation*
93	30-33	I	The indexes referred to in the program do not match those defined during template creation
94	**13**	I, R	A MIDASPLUS concurrency error The command attempted to operate on a record that another user just deleted
95	**32**	I, R	A record length was supplied for the file that does not match the record size assigned to the file during template creation

Status Code	*MPLUS Code*	*File Type*	*Interpretation*
96	20/21	R	A record number was supplied larger than the number that CREATK allocated.
98	35	R	An attempt was made to do an indexed add to a direct access file. Entries cannot be added to a RELATIVE file even if it is opened for INDEX access.
99		I, R	Any MIDASPLUS system error (greater than 40) that cannot be handled at the program level.