

Advantage™ VISION:Transact® for CICS®

Toolkit Reference Guide

Release 7.8



Computer Associates®

B02573-1E

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227- 19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2004 Computer Associates International, Inc.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: The VISION:Transact Toolkit

Overview	1-1
Purpose of this Document	1-2
Record Layouts and Sample Code	1-2
Related Documents	1-3

Chapter 2: Tailoring the Toolkit Programs

Introduction	2-1
Changing the Date Format (Optional)	2-1
Assemble and Link Edit Toolkit Programs (Optional)	2-2

Chapter 3: Using the VISION:Transact Toolkit

Communication Between VISION:Transact and the Toolkit Subroutines	3-1
The Common Control Block	3-2
The Routine Control Block	3-3
Abend M5TK and Incorrect RCB Displacement	3-3

Chapter 4: The Toolkit Subroutines

M5TKBIT and M5TKBYTE	4-1
Description	4-1
M5TKBIT Example	4-2
M5TKBYTE Example	4-3
M5TKCOMM	4-3
Description	4-3
Example	4-4

M5TKCOPY	4-5
Description	4-5
Example.....	4-8
M5TKDATG and M5TKDATJ	4-8
Description	4-8
M5TKDATG Example.....	4-10
M5TKDATJ Example	4-10
M5TKHEXA	4-11
Description	4-11
Example.....	4-12
M5TKLJST and M5TKRJST	4-13
Description	4-13
M5TKLJST Example	4-15
M5TKRJST Example	4-15
M5TKPREV.....	4-16
Description	4-16
Example.....	4-19
M5TKPRNT	4-20
Description	4-20
Example.....	4-21
M5TKTAB.....	4-22
Description	4-22
M5TKTAB Example	4-25
How To Define Tables for M5TKTAB	4-25
Table Definition Example.....	4-30
How Tables Are Searched	4-31
Table Structure.....	4-31
Assemble and Link Edit M5TKTAB Tables.....	4-34
M5TKTCT	4-35
Description	4-35
Example.....	4-37
M5TKTS	4-37
Description	4-37
Example.....	4-40
M5TKUSRA	4-40
Description	4-40
Example.....	4-41
M5TKVLID.....	4-42
Description	4-42
Check Digit Calculation	4-47
Edit Pattern Examples.....	4-48
Example.....	4-48

Appendix A: Defining Validation Symbols for M5TKVLKID

Introduction	A-1
The VALIDGEN Macro	A-2
The VALIDTAB Macro	A-2
Example	A-4
Coding Validation Routines	A-4
M5TKVLID Source Listing	A-5

Appendix B: Sample File Definitions

TWA Filler	B-1
Fields Used by the Toolkit	B-1

Index

The VISION:Transact Toolkit

Overview

The Advantage™ VISION:Transact® Toolkit is a set of subroutines which enables you to perform some useful functions that are not directly supported by VISION:Transact. The Toolkit contains the following subroutines:

M5TKBIT	Translates the zeroes and ones in an eight-byte field to zero and one bits in a one-byte field.
M5TKBYTE	Translates the zero and one bits in a one-byte field to zeroes and ones in an eight-byte field.
M5TKCOMM	Allows a CICS COMMAREA to be passed to a called program.
M5TKCOPY	Duplicates (copies) variable length VSAM KSDS records.
M5TKDATG	Translates a date from Julian format (YY/DDD) to Gregorian format (MM/DD/YY or DD/MM/YY).
M5TKDATJ	Translates a date from Gregorian format (MM/DD/YY) or DD/MM/YY to Julian format (YY/DDD).
M5TKHEXA	Translates a field to hexadecimal. Is useful for testing and debugging.
M5TKLJST	Left-justifies a field and fills on the right with a pad character.
M5TKPREV	Reads a VSAM KSDS file backwards. Used in applications that must browse backwards to determine the starting key value for a subsequent forward browse.
M5TKPRNT	Sends the current screen contents to another CICS terminal which may be a 3270-type display device or a printer.
M5TKRJST	Right-justifies a field and fills on the left with a pad character.
M5TKTAB	Performs table lookup functions. Uses tables created by batch jobs
M5TKTCT	Obtains the cursor's position on the application's input screen.
M5TKTS	Creates, retrieves, updates, and deletes records in temporary storage queues.

M5TKUSRA	Allows you to save data in the Terminal Control Table User Area (TCTUA).
M5TKVLID	Performs field validation. There are predefined validation symbols for various date, alphanumeric, and numeric formats. You can define additional symbols.

Purpose of this Document

This document describes the VISION:Transact Toolkit subroutines and explains how to use them. It assumes that the reader knows how to write applications using VISION:Transact and, therefore, does not attempt to teach VISION:Transact.

[Chapter 1, “The VISION:Transact Toolkit”](#) is a brief introduction to the Toolkit.

[Chapter 2, “Tailoring the Toolkit Programs”](#) shows how to tailor the toolkit programs to conform to your particular requirements.

[Chapter 3, “Using the VISION:Transact Toolkit”](#) relates the general rules for using the Toolkit.

[Chapter 4, “The Toolkit Subroutines”](#) describes the individual Toolkit subroutines and how to use them.

[Appendix A, “Defining Validation Symbols for M5TKVLKID”](#) shows how to define additional validation symbols for M5TKVLID.

[Appendix B, “Sample File Definitions”](#) contains sample file definitions with the same field names used in the examples.

Record Layouts and Sample Code

Record layouts are presented in a tabular format that shows the field name, length, type, and description. Field offsets are omitted because the actual offsets in working storage are application dependent.

Examples are written using ASL, the Computer Associates free-form language used to build procedures.

Related Documents

- *Advantage VISION:Transact for CICS Specifications Guide*
- *Advantage VISION:Transact for CICS Reference Guide*
- *Advantage VISION:Transact for CICS ASL Reference Guide*

Tailoring the Toolkit Programs

Introduction

The VISION:Transact toolkit programs are delivered as both Assembler source code programs and as ready-to-use CICS program modules. If no tailoring is required, you can use the CICS program modules received on the installation tape as is.

The remainder of this chapter discusses how you can tailor the toolkit programs to conform to your particular requirements.

Changing the Date Format (Optional)

The Toolkit, as delivered, assumes that Gregorian dates use the MMDDYY format. If necessary, you can modify the Toolkit source to use the DDMMYY format.

The TKDATES macro is used to select the format for Gregorian dates.

TKDATES DATE=MMDDYY Gregorian dates are in MMDDYY format

TKDATES DATE=DDMMYY Gregorian dates are in DDMMYY format

The TKDATES macro appears in the source code of three programs:

- M5TKDATG
- M5TKDATJ
- M5TKVLID

The date format must be the same in all three programs and must agree with the date format specified in M5PARM. If you modify the date format on the TKDATES macro in the above source programs, you must assemble and link edit these programs. The JCL to do this is described in [Assemble and Link Edit Toolkit Programs \(Optional\) on page 2-2](#).

Assemble and Link Edit Toolkit Programs (Optional)

You must assemble and link edit any Toolkit programs that you modify. The JCL to do this is in source library member ASMTK and is shown in [Figure 2-1](#). The JCL consists of a PROC and statements to execute the PROC for each Toolkit program. The EXEC statements have an asterisk in column 3 so that they are treated as comments. You can remove the asterisk for the programs that you want to assemble and link edit.

You will need to modify the PROC to use the data set names for your source and load libraries.

The PROC has five steps:

1. Allocates a temporary PDS for Step 2.
2. Uses the CICS translator to translate TKENT\$\$ and puts the output in the temporary PDS (created in Step 1) as member TKENT.

TKENT\$\$ is the macro that generates the common entry/exit code used by all Toolkit programs. It contains CICS commands so it must be translated before it can be used. (The CICS translator does not expand assembler macros. If a macro contains commands, it must be translated before it can be used in a program.) The Toolkit programs use the translated version of the macro, TKENT, not TKENT\$\$.
3. Uses the CICS translator to translate the Toolkit source program.
4. Assembles the output from Step 3. The PDS created in Step 1 is in the SYSLIB concatenation.
5. Link edits the output from Step 4.

```
//jobname JOB
//*
//*****
//*
//*   THIS PROCEDURE ASSEMBLES AN ADVANTAGE VISION:TRANSC
//*   TOOL KIT SOURCE PROGRAM.
//*
//*   THE CICS COMMAND LEVEL TRANSLATER IS INVOKED FOLLOWED BY AN
//*   ASSEMBLY AND LINKEDIT.
//*
//*****
//*
//ASMTK  PROC INDEX='CICSSYS.CTS220.CICS',      LEVEL OF CICS TO USE
//          SRCMBR=,                          SOURCE MEMBER NAME
//          SRCLIB='your.toolkit.srclib',      SOURCE LIBRARY NAME
//          LOADMOD=,                          LOAD MODULE NAME
//          LOADLIB='your.toolkit.loadlib',    LOAD LIBRARY NAME
//          PRINTQ='*',                        SYSOUT Q FOR PRINT
//          SYSDA=SYSDA                        SYSTEM DASD UNIT TYPE
//*
//*****
//*
//*   CREATE TEMPORARY PDS
//*
//*****
//PDS    EXEC PGM=IEFBR14
```

Figure 2-1 JCL to Assemble and Link Edit Toolkit (Page 1 of 3)

```

//SYSPUNCH DD DSN=&&SYSLIB,DISP=(NEW,PASS),
//          UNIT=&SYSDA,SPACE=(CYL,(1,1,1)),
//          DCB=BLKSIZE=27920
//*
//*****
//*
//* EXECUTE THE CICS COMMAND LEVEL TRANSLATOR
//* TO TRANSLATE THE COMMON ENTRY/EXIT MACRO
//*
//*****
//*
//TRX EXEC PGM=DFHEAP1$,REGION=4M
//STEPLIB DD DISP=SHR,DSN=&INDEX..SDFHLOAD
//SYSPRINT DD SYSOUT=&PRINTQ
//SYSPUNCH DD DSN=&&SYSLIB(TKENT),DISP=(OLD,PASS)
//SYSIN DD DISP=SHR,DSN=&SRCLIB(TKENT$$)
//*
//*****
//*
//* EXECUTE THE CICS COMMAND LEVEL TRANSLATOR
//* TO TRANSLATE THE SOURCE PROGRAM
//*
//*****
//*
//TRN EXEC PGM=DFHEAP1$,REGION=4M
//STEPLIB DD DISP=SHR,DSN=&INDEX..SDFHLOAD
//SYSPRINT DD SYSOUT=&PRINTQ
//SYSPUNCH DD DSN=&&SYSIN,DISP=(NEW,PASS),
//          UNIT=&SYSDA,SPACE=(CYL,(1,1)),
//          DCB=BLKSIZE=27920
//SYSIN DD DISP=SHR,DSN=&SRCLIB(&SRCMBR)
//*
//*****
//*
//* ASSEMBLE THE PROGRAM
//*
//*****
//*
//ASM EXEC PGM=ASMA90,REGION=4M,TIME=(2,00),
//      PARM=(NOTERM,RENT,NODECK,OBJECT,'SYSPARM(OS)',
//      LIST,RLD,'LINECOUNT(58)','XREF(FULL)')
//SYSPRINT DD SYSOUT=&PRINTQ
//SYSLIN DD DSN=&&SYSLIN,DISP=(NEW,PASS),
//          UNIT=&SYSDA,SPACE=(CYL,(1,1)),
//          DCB=BLKSIZE=27920
//SYSUT1 DD UNIT=&SYSDA,SPACE=(CYL,(1,1))
//SYSLIB DD DISP=SHR,DSN=&SRCLIB
//          DD DSN=&&SYSLIB,DISP=(OLD,DELETE)
//          DD DISP=SHR,DSN=&INDEX..SDFHMAC
//          DD DISP=SHR,DSN=SYS1.MACLIB
//SYSIN DD DSN=&&SYSIN,DISP=(OLD,DELETE)
//*
//*****
//*
//* LINKEDIT THE PROGRAM
//*
//*****
//*
//LKED EXEC PGM=IEWL,PARM='LET,LIST,MAP,RENT',REGION=4M
//SYSPRINT DD SYSOUT=&PRINTQ
//SYSLMOD DD DISP=SHR,DSN=&LOADLIB(&LOADMOD)
//SYSUT1 DD UNIT=&SYSDA,SPACE=(CYL,(1,1))
//SYSLIB DD DISP=SHR,DSN=&INDEX..SDFHLOAD
//SYSLIN DD DSN=&&SYSLIN,DISP=(OLD,DELETE)
// PEND
//*
//*****
//*
//* EXEC ASMTK,SRCMBR=M5TKBIT,LOADMOD=M5TKBIT
//* EXEC ASMTK,SRCMBR=M5TKBYTE,LOADMOD=M5TKBYTE
//* EXEC ASMTK,SRCMBR=M5TKCOMM,LOADMOD=M5TKCOMM
//* EXEC ASMTK,SRCMBR=M5TKCOPY,LOADMOD=M5TKCOPY
//* EXEC ASMTK,SRCMBR=M5TKDATG,LOADMOD=M5TKDATG
//* EXEC ASMTK,SRCMBR=M5TKDATJ,LOADMOD=M5TKDATJ
//* EXEC ASMTK,SRCMBR=M5TKHEXA,LOADMOD=M5TKHEXA
//* EXEC ASMTK,SRCMBR=M5TKPRNT,LOADMOD=M5TKPRNT

```

Figure 2-1 JCL to Assemble and Link Edit Toolkit (Page 2 of 3)

```
/* EXEC ASMTK, SRCMBR=M5TKLCAL, LOADMOD=M5TKLCAL
/* EXEC ASMTK, SRCMBR=M5TKLJST, LOADMOD=M5TKLJST
/* EXEC ASMTK, SRCMBR=M5TKRJST, LOADMOD=M5TKRJST
/* EXEC ASMTK, SRCMBR=M5TKPREV, LOADMOD=M5TKPREV
/* EXEC ASMTK, SRCMBR=M5TKTAB, LOADMOD=M5TKTAB
/* EXEC ASMTK, SRCMBR=M5TKTCT, LOADMOD=M5TKTCT
/* EXEC ASMTK, SRCMBR=M5TKTS, LOADMOD=M5TKTS
/* EXEC ASMTK, SRCMBR=M5TKUSRA, LOADMOD=M5TKUSRA
/* EXEC ASMTK, SRCMBR=M5TKVLID, LOADMOD=M5TKVLID
/*
```

Figure 2-1 JCL to Assemble and Link Edit Toolkit (Page 3 of 3)

.

Using the VISION:Transact Toolkit

Communication Between VISION:Transact and the Toolkit Subroutines

VISION:Transact application programs and the Toolkit subroutines communicate through VISION:Transact working storage which is physically located in the CICS Transaction Work Area (TWA). Toolkit routines contain CICS commands and must not be link edited with the application.

All Toolkit subroutines pass data through the Toolkit common control block which is located four bytes from the beginning of VISION:Transact working storage and, thus, four bytes from the beginning of the CICS TWA. The format of the common control block is constant for all subroutines.

Each Toolkit subroutine also passes data through another area in VISION:Transact working storage called the Toolkit routine control block. The format of the routine control block varies depending upon the subroutine. The various formats are described in [Chapter 4, “The Toolkit Subroutines”](#).

The Common Control Block

The common control block (CCB) is used by all Toolkit subroutines. It tells a called subroutine how to find its routine control block, contains the return code from the subroutine and, if necessary, also contains an error message. [Figure 3-1](#) shows the format of the common control block. The field names are arbitrary. You can, of course, use whatever field names you desire, but the *format* of the common control block must be as shown.

Field Name	Field Length	Field Type	Field Description
CCBRCBDS	2	2	The displacement (offset), relative to 1, of the routine control block from the beginning of the common control block. For example, the displacements of CCBRCBDS, CCBRCOD, and CCBMSG are 1, 3, and 5 respectively.
CCBRCOD	2	F	Each Toolkit subroutine places a return code in this field. The return code indicates the result of the operation. Zero always indicates success. The meaning of any other value depends upon the subroutine.
CCBMSG	80	C	If a subroutine encounters some kind of error (CCBRCOD is non zero), then it places a message in this field.

Figure 3-1 The Toolkit Common Control Block

The Toolkit routines expect CCBRCBDS to start in the fifth byte of the TWA. Thus, if you are using Release 5.0 (or later) of VISION:Transact, you can ensure the correct starting position by creating a file definition to map the first four bytes of working storage and assign it to WORK01 on an RF statement for your VISION:Transact application. A sample of this file definition is contained in source library member TKFD01 and is shown in Appendix B. The file definition for the common control block is then assigned to WORK02 on an RF statement. A sample file definition for the common control block and routine control blocks for all Toolkit subroutines is contained in source library member TKFD02 and is shown in [Appendix B, “Sample File Definitions”](#).

The Routine Control Block

A Toolkit subroutine communicates with the VISION:Transact application through a routine control block (RCB) that is tailored to the function that the subroutine performs. The routine control block is in VISION:Transact working storage (CICS TWA) but the subroutine needs to be told exactly where in working storage. This is the purpose of the routine control block displacement field, called CCBRCBDS in the table above.

Before you call a Toolkit subroutine, move any data required by the subroutine to the routine control block and then put the displacement of the block in CCBRCBDS. When the subroutine is called, it uses the value you placed in CCBRCBDS to determine the exact location of its routine control block in working storage.

If you use more than one Toolkit subroutine in a VISION:Transact application, then you will have to include a file definition for each of the routine control blocks in working storage or one file definition for all routine control blocks. The routine control blocks can overlay each other in working storage or be defined as separate areas. We recommend the latter except in the case of M5TKDATG and M5TKDATJ, M5TKLJST and M5TKRJST, and M5TKBIT and M5TKBYTE. These subroutines perform similar functions and it may be useful to have the routine control blocks for each pair share the same area.

A sample file definition for the common control block and routine control blocks for all Toolkit subroutines is contained in source library member TKFD02 and is shown in [Appendix B, “Sample File Definitions”](#). All of the examples in [Chapter 4, “The Toolkit Subroutines”](#) use this file definition.

Abend M5TK and Incorrect RCB Displacement

If a Toolkit subroutine is called by a transaction that has no CICS TWA (the TWA length is zero), then the subroutine ABENDs with the code M5TK. No other action is possible because the subroutine has no place to store an error message or any other indication of the problem.

Each Toolkit subroutine validates the displacement in CCBRCBDS. If the displacement is such that the RCB overlaps the CCB or extends beyond the limits of the TWA, then the following error message is placed in CCBMSG:

```
XXXXXXXX - INCORRECT RCB DISPLACEMENT IN CCB
```

where XXXXXXXX is the name of the Toolkit subroutine that issued the error. The return code placed in CCBRCOD is usually 1. Exceptions are noted in the documentation for the individual subroutines.

The Toolkit Subroutines

M5TKBIT and M5TKBYTE

Description

M5TKBIT translates the zeroes and ones in an eight-byte field to zero and one bits in a one-byte (eight-bit) field.

M5TKBYTE translates the zero and one bits in a one-byte field to zeroes and ones in an eight-byte field.

A data base segment can contain many fields that have only two possible values, such as Y or N for Yes and No, or T or F for True and False. M5TKBIT can be used to squeeze eight of these fields into a single byte to reduce segment size and save disk space. Just convert the values that you display, such as Y, N, T, and F, to ones and zeroes in an eight-byte field and call M5TKBIT. To reverse the process, call M5TKBYTE and convert the ones and zeroes to display values.

[Figure 4-1](#) describes the routine control block used by both of these subroutines. We recommend that these subroutines share the same routine control block because this makes it easier to translate from bits to bytes and then back again.

[Figure 4-2](#) lists the return codes and messages issued by these subroutines. A zero return code indicates the operation was successful.

Field Name	Field Length	Field Type	Field Description
BTBITS	1	C	M5TKBYTE translates zero and one bits in this field to character zero (X'F0') and character one (X'F1') bytes in the BTBYTES field. The correspondence is from left to right, that is, the high order bit in BTBITS corresponds to the leftmost byte in BTBYTES.
BTBYTES	8	C	M5TKBIT translates character zeroes and ones in this field to zero and one bits in the BTBITS field. A zero (X'F0') is translated to a zero bit. Any other value is translated to a one bit.

Figure 4-1 M5TKBIT and M5TKBYTE Routine Control Block

Return Code	Message
01	M5TKBIT-INCORRECT RCB DISPLACEMENT IN CCB.
01	M5TKBYTE - INCORRECT RCB DISPLACEMENT IN CCB

Figure 4-2 M5TKBIT and M5TKBYTE Return Codes and Messages

M5TKBIT Example

```
PUT 8-BYTE FIELD CONTAINING 0'S AND 1'S IN RCB
;
LET W.BTBYTES = '00000011'
;
;PUT RCB OFFSET IN CCB AND CALL M5TKBIT
;
LET W.CCBRCBDS = 85
CALL MODULE M5TKBIT
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE W.BTBITS WILL CONTAIN THE
;HEXADECIMAL VALUE 03
```

M5TKBYTE Example

```
;PUT 1-BYTE IN W.BTBITS
;IN THIS EXAMPLE WE'LL ASSUME THAT T.BITS
;CONTAINS THE HEXADECIMAL VALUE 07

;
LET W.BTBITS = T.BITS
;
;PUT RCB OFFSET IN CCB AND CALL M5TKBYTE
;
LET W.CCBRCBDS = 85
CALL MODULE M5TKBYTE
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE W.BTBYTES WILL CONTAIN THE
;8-CHARACTER VALUE '00000111'
;
```

M5TKCOMM

Description

M5TKCOMM allows you to pass a CICS COMMAREA to a called subroutine. This subroutine is provided for compatibility only, since the VISION:Transact CALL statement can be used to pass a COMMAREA in VISION:Transact Release 5.0 and later.

You supply the name of the program to be called, the length of the COMMAREA, and the displacement of the COMMAREA (in working storage).

[Figure 4-3](#) describes the routine control block used by this subroutine.

[Figure 4-4](#) lists the return codes and messages issued by this subroutine. A zero return code indicates the operation was successful.

Field Name	Field Length	Field Type	Field Description
CMPRGNAM	8	C	The name of the program you want to call as specified in the CICS PPT.
CMLen	2	F	The length of the COMMAREA.
CMDISPL	2	F	The displacement of the COMMAREA from the beginning of the CCB.

Figure 4-3 M5TKCOMM Routine Control Block

Return Code	Message
01	M5TKCOMM - INCORRECT RCB DISPLACEMENT IN CCB.
02	CM02 - INVALID COMMAREA DISPLACEMENT/LENGTH. This message is issued when one of the following is true: ■ The module value in CMLen is less than 1 ■ The CMDISPL field implies that the COMMAREA overlaps the CCP. ■ CMDISPL and CMLen imply that the COMMAREA extends past the end of the TWA.
03	CM03 - MODULE XXXXXXXXX IS DISABLED OR NOT DEFINED IN THE CICS PPT. The program named in CMPRGNAM is not in the PPT or the PPT entry indicates that the program is disabled.

Figure 4-4 M5TKCOMM Return Codes and Messages

Example

```

;
; FILL IN RCB
;
LET W.CMPRGNAM = 'YOURPROG'          ;PROGRAM TO CALL
LET W.CMLen = 121                    ;COMMAREA LENGTH
LET W.CMDISPL = 2629                 ;COMMAREA OFFSET FROM CCB
;
;PUT RCB OFFSET IN CCB AND CALL M5TKCOMM
;
LET W.CCBRCBDS = 2885
CALL MODULE M5TKCOMM
;
; AFTER ANY TOOLKIT SUBROUTINE IS CALLED, YOU SHOULD CHECK

```

```
; THE RETURN CODE IN THE CCB. A NON-ZERO VALUE INDICATES  
; THAT THERE WAS A PROBLEM.  
;  
IF W.CCBRCOD <> 0  
  < CODE TO HANDLE THE CONDITION >  
END  
;  
;WITH VISION:Transact RELEASE 5.0 AND LATER, A COMMAREA CAN BE PASSED  
;WITHOUT CALLING M5TKCOMM USING THE FOLLOWING STATEMENT:  
;  
CALL MODULE YOURPROG COMMAREA-NAME COMMAREA-LENGTH
```

M5TKCOPY

Description

M5TKCOPY duplicates (copies) variable length VSAM KSDS records.

You supply the name of the file, the key for an existing record, and the new key. M5TKCOPY reads the existing record and then inserts it with the new key value.

One possible use for this subroutine is to change the key of an existing record. VISION:Transact does not allow you to change a record's key. If you need to change a key, you can call M5TKCOPY to copy the record whose key needs to be changed. If the copy is successful, you can use VISION:Transact to delete the record with the old key.

[Figure 4-5](#) describes the routine control block used by this subroutine.

[Figure 4-6](#) lists the return codes and messages issued by this subroutine. A zero return code indicates the copy was successful.

Field Name	Field Length	Field Type	Field Description
COFILNAM	8	C	The CICS File Control Table name for the file containing the record you want to copy.
COKEYLEN	2	F	The length of the record key.
COKEYOFF	2	F	The record key's displacement (offset) from the beginning of the data portion of the record. This is relative to 1. If the key starts in position 18, then the value used here is 18. If the key is at the beginning of the record, then use 1 here.
COOLDKEY	(1)	(2)	Key for the existing record.
CONEWKEY	(1)	(2)	Key to use for the copied record. The record whose key is COOLDKEY is retrieved and then inserted with key CONEWKEY. The record with CONEWKEY is not modified.

Figure 4-5 M5TKCOPY Routine Control Break

Note:

1. The field length must match the record's key length, that is, the value that is in COKEYLEN.
2. The field type specified must match the field type of the record's key field.

Return Code	Message
01	M5TKCOPY - INCORRECT RCB DISPLACEMENT IN CCB.
02	CO02 - NOT ENOUGH SPACE AVAILABLE FOR GETMAIN. COPY ABORTED. Not enough storage is available to create a copy of the record with its new key value. The copy cannot be performed.
03	CO03 - RECORD ALREADY EXISTS. COPY ABORTED. A record with the key that you supplied in CONEWKEY already exists. The copy is not performed. The existing record is not modified.

Figure 4-6 M5TKCOPY Return Codes and Messages (Page 1 of 2)

Return Code	Message
04	CO04 - FILE IS FULL. COPY ABORTED. There is no more room in the file named in COFILNAM. The copy cannot be performed.
05	CO05 - RECORD TO BE COPIED COULD NOT BE FOUND. COPY ABORTED. The record whose key is in COOLDKEY does not exist. The copy cannot be performed.
06	CO06 - RECORD KEY LENGTH NOT SUPPLIED. COPY ABORTED. The length supplied in COKEYLEN is less than 1. The copy cannot be performed.
07	CO07 - FILE XXXXXXXX NOT DEFINED IN FCT. COPY ABORTED. The file named in COFILNAM is not in the CICS FCT. The copy cannot be performed.
08	CO08 - FILE XXXXXXXX NOT OPENED. COPY ABORTED. The file named in COFILNAM is not open. The copy cannot be performed.
09	CO09 - UNEXPECTED ERROR HAS OCCURRED WHILE READING FILE XXXXXXXX. An error occurred while trying to read from the file named in COFILNAM. The copy cannot be performed. Possible causes: wrong key length or offset, not authorized to read the file, the file is disabled, the file is not a KSDS file, or an I/O error occurred.
10	CO10 - UNEXPECTED ERROR HAS OCCURRED WHILE WRITING FILE XXXXXXXX. An error occurred while trying to insert the copied record into the file named in COFILNAM. The copy cannot be performed. Possible causes: wrong key length or offset, not authorized to insert into the file, file is disabled, file is not a KSDS file, unrecoverable I/O error occurred.

Figure 4-6 M5TKCOPY Return Codes and Messages (Page 2 of 2)

Example

For this example we assume that the file THEFILE has a 5-character key that starts at the beginning of the record. We also assume that COOLDKEY and CONEWKEY have been defined as character (C) fields with length 5. Note that in this example, the starting location of field CONEWKEY must be 5 bytes past the starting location of field COOLDKEY.

```
;
;FILL IN RCB
;
LET W.COFILNAM = 'THEFILE' ;FILE (FCT) NAME
LET W.COKEYLEN = 5 ;LENGTH OF KEY
LET W.COKEYOFF = 1 ;OFFSET OF KEY IN RECORD
LET W.COOLDKEY = '10110' ;KEY OF RECORD TO COPY
LET W.CONEWKEY = '10111' ;KEY OF NEW RECORD
;
;PUT RCB OFFSET IN CCB AND CALL M5TKCOPY
;
LET W.CCBRCBDS = 94
CALL MODULE M5TKCOPY
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, THE RECORD WITH KEY '10110' WOULD BE
;DUPLICATED WITH THE KEY '10111'
;
```

M5TKDATG and M5TKDATJ

Description

M5TKDATG converts a Julian date in the format YY/DDD to a Gregorian date in the format MM/DD/YY (or DD/MM/YY).

M5TKDATJ converts a Gregorian date in the format MM/DD/YY (or DD/MM/YY) to a Julian Date in the format YY/DDD.

[Figure 4-7](#) describes the routine control block used by both of these subroutines. We recommend that these subroutines share the same routine control block because this makes it easier to convert from one format to the other and then back again.

[Figure 4-8](#) lists the return codes and messages for both subroutines. A zero return code indicates the date conversion was successful.

See [Changing the Date Format \(Optional\) on page 2-1](#) for instructions on changing the Gregorian date format used by the Toolkit from MM/DD/YY to DD/MM/YY.

Field Name	Field Length	Field Type	Field Description
GJDATEJ	6	C	Julian date in format YY/DDD. M5TKDATG takes this field, converts it to a Gregorian date, and puts the result in field GJDATEG.
GJDATEG	8	C	Gregorian date in format MM/DD/YY (or DD/MM/YY). M5TKDATJ takes this field, converts it to a Julian date, and puts the result in field GJDATEJ.

Figure 4-7 M5TKDATG and M5TKDATJ Routine Control Block

Note: Neither M5TKDATG nor M5TKDATJ reads the slash (/) so any character can be used.

Return Code	Message
01	M5TKDATG - INCORRECT RCB DISPLACEMENT IN CCB.
02	GG02 - JULIAN DATE IS NOT NUMERIC. The Julian date (YY/DDD) contains non-numeric characters in the YY or DDD.
04	GG04 - NUMBER OF DAYS IN JULIAN DATE IS INVALID. DDD in the Julian date (YY/DDD) cannot exceed 365 (366 for leap years).
01	M5TKDATJ- INCORRECT RCB DISPLACEMENT IN CCB.
02	GJ02 - INVALID MONTH SPECIFIED IN GREGORIAN DATE. Month cannot exceed 12.
03	GJ03 - INVALID DAY SPECIFIED IN GREGORIAN DATE. Day cannot exceed number of days in the month.
04	GJ04 - GREGORIAN DATE IS NOT NUMERIC. The Gregorian date contains non-numeric characters.

Figure 4-8 M5TKDATG and M5TKDATJ Return Codes and Messages

M5TKDATG Example

```
;PUT JULIAN DATE (YY/DD) IN W.GJDATEJ
;
LET W.GJDATEJ = '93/005'
;
;PUT RCB OFFSET IN CCB AND CALL M5TKDATG
;
LET W.CCBRCBDS = 616
CALL MODULE M5TKDATG
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, W.GJDATEG WILL CONTAIN 01/05/93
```

M5TKDATJ Example

```
;PUT GREGORIAN DATE (MM/DD/YY) IN W.GJDATEJ
;
LET W.GJDATEG = '01/07/93'
;
;PUT RCB OFFSET IN CCB AND CALL M5TKDATJ
;
LET W.CCBRCBDS = 616
CALL MODULE M5TKDATJ
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, W.GJDATEJ WILL CONTAIN 93/007
```

M5TKHEXA

Description

M5TKHEXA translates a field to hexadecimal so that its exact contents can be displayed. This is useful when debugging an application, especially if you need to see what is in a packed field.

[Figure 4-9](#) describes the routine control block used by this subroutine.

[Figure 4-10](#) lists the return codes and error messages issued by this subroutine. A zero return code indicates the operation was successful.

Field Name	Field Length	Field Type	Field Description
XALEN	2	F	Length of data to be converted to hexadecimal. The maximum value that you can use is 127. The number of bytes of hexadecimal data will be twice this value since the hexadecimal representation takes two bytes for each input byte. For example, the hexadecimal representation of the character 2 is F2.
XAFROM	127	C	The field that you want to translate to hexadecimal. Note that the field type is C. Normally, you would overlay (redefine) this field with packed and binary fields to make it easy to obtain the hexadecimal representation of those types of fields.
XATO	254	C	M5TKHEXA puts the hexadecimal representation of the contents of XAFROM in this field. The hexadecimal representation uses XALEN * 2 bytes. The remainder of the field is not modified.

Figure 4-9 M5TKHEXA Routine Control Block

Return Code	Message
01	M5TKHEXA - INCORRECT RCB DISPLACEMENT IN CCB.
02	XA02 - INVALID 'FROM' FIELD LENGTH. LENGTH MUST BE BETWEEN 1 AND 127.

Figure 4-10 M5TKHEXA Return Codes and Messages

Example

For this example we assume that XAFROMP2 is a 2-byte packed field that overlays XAFROM.

```
;FILL IN RCB
;
LET W.XALEN = 2 ;LENGTH OF FIELD
LET W.XAFROMP2 = 27 ;VALUE TO CONVERT TO HEX
;
;PUT RCB OFFSET IN CCB AND CALL M5TKHEXA
;
LET W.CCBRCBDS = 630
CALL MODULE M5TKHEXA
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, XATO WILL CONTAIN THE
;HEXADECIMAL VALUE '027C'
```

M5TKLJST and M5TKRJST

Description

M5TKLJST left-justifies a field and then pads it on the right with an optional pad character. It uses the following algorithm:

- Scan JSDATA from left to right skipping blanks and the character specified in JSLEADTR.
- Left-justify the remaining contents of JSDATA.
- Move spaces to JSDATA bytes JSLEN + 1 through the end of the field.
- Replace trailing spaces in the first JSLEN bytes of JSDATA with the character specified in JSPAD.

M5TKRJST right-justifies a field and then pads it on the left with an optional pad character. It uses the following algorithm:

- Scan JSDATA from right to left skipping blanks and the character specified in JSLEADTR.
- Right-justify the remaining contents of JSDATA in the first JSLEN bytes of JSDATA.
- Move spaces to JSDATA bytes JSLEN + 1 through the end of the field.
- Replace leading spaces in the first JSLEN bytes of JSDATA with the character specified in JSPAD.

[Figure 4-11](#) describes the routine control block used by these subroutines.

[Figure 4-12](#) lists the return codes and error messages issued by these subroutines. Azero return code indicates the operation was successful.

Field Name	Field Length	Field Type	Field Description
JSLEN	2	F	Length of the field to be left- or right-justified. This length must be between 1 and 255. The result will occupy the first JSLEN bytes of JSDATA.
JSLEADTR	1	C	For M5TKLJST leading occurrences of this character are treated as blanks. For M5TKRJST trailing occurrences of this character are treated as blanks. Normally, this field is blank.
JSPAD	1	C	The padding character. Replaces trailing spaces after left-justification and leading spaces after right-justification.
JSDATA	255	C	Before the CALL, this field contains the data to be justified. After the CALL, the first JSLEN bytes of this field contain the justified data with its padding. The remainder of the field is filled with spaces.

Figure 4-11 M5TKLJST and M5TKRJST Routine Control Block

Return Code	Message
01	M5TKLJST - INCORRECT RCB DISPLACEMENT IN CCB.
04	JL04 - THE FIELD LENGTH SUPPLIED IS INVALID. LENGTH MUST BE BETWEEN 1 AND 255.
01	M5TKRJST - INCORRECT RCB DISPLACEMENT IN CCB.
04	JR04 - THE FIELD LENGTH SUPPLIED IS INVALID. LENGTH MUST BE BETWEEN 1 AND 255.

Figure 4-12 M5TKLJST and M5TKRJST Return Codes and Messages

M5TKLJST Example

```
;FILL IN RCB
;
LET W.JSLEN = 10 ;LENGTH OF RESULT
LET W.JSLEADTR = '.' ;IGNORE LEADING '.' (AND BLANKS)
LET W.JSPAD = '.' ;PAD WITH '.'
;
;PUT DATA TO BE LEFT JUSTIFIED IN W.JSDATA
;
LET W.JSDATA = ' .... 12345'
;
;PUT RCB OFFSET IN CCB AND CALL M5TKLJST
;
LET W.CCBRCBDS = 1013
CALL MODULE M5TKLJST
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, W.JSDATA WILL CONTAIN '12345....'
```

M5TKRJST Example

```
;FILL IN RCB
;
LET W.JSLEN = 10 ;LENGTH OF RESULT
LET W.JSLEADTR = '.' ;IGNORE TRAILING '.' (AND BLANKS)
LET W.JSPAD = '.' ;PAD WITH '.'
;
;PUT DATA TO BE RIGHT JUSTIFIED IN W.JSDATA
;
LET W.JSDATA = '12345.... '
;
;PUT RCB OFFSET IN CCB AND CALL M5TKRJST
;
LET W.CCBRCBDS = 1013
CALL MODULE M5TKRJST
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, W.JSDATA WILL CONTAIN '.....12345'
```

M5TKPREV

Description

M5TKPREV reads a VSAM KSDS file backwards. It can be used in applications that must browse backwards to determine the starting key value for a subsequent forward browse. It is normally used in the following manner:

- You have a VISION:Transact screen that holds data from some number of records, for example, 10.
- Your application outputs a screenful of data by browsing a file.
- Your application provides some method that allows the operator to request that you skip forward and backward through the file's records.
- Forward skipping is no problem. You start a forward browse in a Type O (output processing) request, skip some number of records, and output a screenful.
- Backward skipping presents a problem because VISION:Transact does not support backward browsing. You need to find the key to use to start a forward browse.
- To solve your problem, you tell M5TKPREV to browse backwards through the number of records that you specify. It will return the key of the record at which it stops, and you can use that key to start a forward browse.

[Figure 4-13](#) describes the routine control block used by this subroutine.

[Figure 4-14](#) lists the return codes and error messages issued by this subroutine. A zero return code indicates the operation was successful.

Field Name	Field Length	Field Type	Field Description
PVCOUNT	2	F	The number of VSAM records to read backwards. This value is required and must be 1 or greater.
PVKEYLEN	2	F	The length of the VSAM record key as defined in the CICS File Control Table (FCT). This value is required and must be 1 or greater.
PVGENLEN	2	F	Generic key length. This field can be zero, in which case generic key processing is not used. If a non-zero value is specified, it can be any value from 1 up to 1 less than the value specified in PVKEYLEN. Generic key processing allows you to browse backwards until one of the following conditions is met: PVCOUNT records have been read or the first PVGENLEN bytes in the key of the last record that was read do not match the first PVGENLEN bytes in PVKEY.
PVFCTNAM	8	C	The CICS FCT name of the VSAM KSDS.
PVKEY	<i>n</i>	<i>t</i>	The full record key of the record that you want to use as the starting point of the backward browse. The record with this key must exist. If the record does not exist, then a NOT FOUND condition occurs. The value specified for <i>n</i> must be the length of the key field as defined in the CICS FCT. The field type, in <i>t</i> , must match the field type of the record key.

Figure 4-13 M5TKPREV Routine Control Block

Return Code	Message
01	No message returned. The beginning of the VSAM file has been reached before exhausting the specified number of records to be read backwards.
02	PV02 - FILE XXXXXXXX IS NOT OPENED. REQUEST CANCELLED.
03	PV03 - FILE XXXXXXXX IS NOT DEFINED IN THE FCT. REQUEST CANCELLED.
04	PV04 - A NOT FOUND CONDITION HAS BEEN RAISED WHILE XXXXXXXX FILE YYYYYYYY. XXXXXXX will be either READING or BROWSING. In either case, it means that the starting key you supplied in PVKEY has no matching record in the file.
05	PV05 - THE KEY LENGTH MAY NOT BE SET TO ZERO. REQUEST CANCELLED.
06	PV06 - AN UNEXPECTED ERROR WAS ENCOUNTERED WHILE XXXXXXXX FILE YYYYYYYY. XXXXXXX will be either READING or BROWSING. Possible causes: wrong key length or offset, not authorized to read the file, the file is disabled, the file is not a KSDS file, or an I/O error occurred.
07	M5TKPREV - INCORRECT RCB DISPLACEMENT IN CCB.

Figure 4-14 M5TKPREV Return Codes and Messages

Example

For this example we assume that the file whose CICS FCT name is THEFILE has a 5-character key. We also assume that PVKEYC5 is a character (C) field with a length of 5 that overlays PVKEY.

```
;FILL IN RCB
;
LET W.PVCOUNT = 10           ;READ 10 BACKWARDS
LET W.PVKEYLEN = 5            ;KEY LENGTH
LET W.PVGENLEN = 0            ;NOT A GENERIC BROWSE
LET W.PVFCTNAM = 'THEFILE'    ;FILE (FCT) NAME
LET W.PVKEYC5 = '10100'      ;STARTING KEY
;
;PUT RCB OFFSET IN CCB AND CALL M5TKPREV
;
LET W.CCBRCBDS = 1272
CALL MODULE M5TKPREV
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, IF THE RETURN CODE IS 1, THEN WE REACHED
;THE BEGINNING OF THE FILE BEFORE WE READ 10 RECORDS. PVKEYC5
;CONTAINS '10100'
;
;IF THE RETURN CODE IS 0, THEN PVKEYC5 CONTAINS THE
;KEY OF THE 10TH RECORD THAT WAS READ.
```

M5TKPRNT

Description

M5TKPRNT sends the contents of the current screen to another CICS terminal which can be a 3270-type display device or a printer. Support is limited to 80 column displays and printers.

M5TKPRNT reads the contents of the screen and starts transaction M5TK for the specified terminal with the screen contents as input.

M5TKLCAL is the program associated with transaction M5TK. It retrieves its data (the screen contents) and writes the data to the terminal or printer for which it was started. Nulls and dark fields are replaced with blanks.

[Figure 4-15](#) describes the routine control block used by this subroutine.

[Figure 4-16](#) lists the return codes and messages issued by this subroutine. A zero return code indicates the operation was successful.]

Field Name	Field Length	Field Type	Field Description
PRTERMID	4	C	Terminal ID, as specified in the CICS Terminal Control Table (TCT), of the 3270-type terminal or printer where you want to send the current screen.

Figure 4-15 M5TKPRNT Routine Control Block

Return Code	Message
01	M5TKPRNT - INCORRECT RCB DISPLACEMENT IN CCB.
02	PR02 - LTERM XXXX NOT DEFINED IN TCT. COPY ABORTED.
03	PR03 - TRANID M5TK IS NOT DEFINED. COPY ABORTED. Transaction M5TK, associated with program M5TKLCAL, is used to print or display the screen at the destination terminal. M5TK is not in the PCT or M5TKLCAL is not in the PPT.
04	PR04 - UNEXPECTED ERROR. COPY ABORTED. Possible causes: unrecoverable I/O error, not enough temporary storage space.

Figure 4-16 M5TKPRNT Return Codes and Messages

Example

```
;FILL IN RCB
;
LET W.PRTERMID = 'PTR5'          ;SEND TO TERMINAL <@147>PTR5"
;
;PUT RCB OFFSET IN CCB AND CALL M5TKPRNT
;
LET W.CCBRCBDS = 1541
CALL MODULE M5TKPRNT
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
```

M5TKTAB

Description

M5TKTAB performs table lookup functions. The tables used by this subroutine are created by batch jobs and may contain up to four columns. You can use any of the columns as the *argument* (the value you know) or as the *function* (the result of the search).

[How To Define Tables for M5TKTAB on page 4-25](#) shows how to define tables for M5TKTAB.

[Figure 4-17](#) describes the routine control block used by this subroutine.

[Figure 4-18](#) lists the return codes and messages for this subroutine. A zero return code indicates the table lookup was successful.

Field Name	Field Length	Field Type	Field Description
TLTABNAM	8	C	Table name. This is the name that you assigned to the table's module in the CICS Processing Program Table (PPT).
TLARGCOL	2	F	Column number, from 0 to 4, that you want to use as the argument column. The value cannot exceed the number of columns actually in the table. For a direct lookup table, you can use zero to indicate that the argument value is the row number that you want. If the table is not a direct table, then a zero in this field is changed to a 1.

Figure 4-17 M5TKTAB Routine Control Block (Page 1 of 2)

Field Name	Field Length	Field Type	Field Description
TLFUNCOL	2	F	Column number, from 0 to 4, that you want to use as the function column. The value cannot exceed the number of columns actually in the table. For a direct lookup table, you can use zero to indicate that the function value is the row number that contains the argument. If the table is not a direct table, then a zero in this field is changed to the value in TLARGCOL.
TLARGVAL	argn	argt	The argument value. The length, argn, and type, argt, must match the length and type of the table column specified in TLARGCOL. You supply this value before the CALL. The subroutine tries to locate this value in column TLARGCOL of the table named in TLTABNAM. For a direct table, if TLARGCOL is zero, then this field should be defined as a fixed point (F) field with length 2. The value that you put in this field is the number of the row in the table.
TLFUNVAL	fn	ft	The function value. The length, fn, and type, ft, must match the length and type of the table column specified in TLFUNCOL. If the argument is found in the table, then the contents of column TLFUNCOL in the same table row are placed here. If the argument is not found, the default value for the column (TLFUNCOL) is placed here. For a direct table, if TLFUNCOL is zero, then this field should be defined as a fixed point (F) field with length 2. The value that is returned in this field is the number of the row in the table.

Figure 4-17 M5TKTAB Routine Control Block (Page 2 of 2)

Return Code	Message
01	M5TKTAB - INCORRECT RCB DISPLACEMENT IN CCB.
02	TL02 - TABLE XXXXXXXXX IS DISABLED OR NOT DEFINED IN CICS PPT.
03	This return code means that the argument value was not found in the table. No message is issued.
04	TL04 - INVALID ARGUMENT COLUMN NUMBER. Column number must be between 0 and the number of columns in the table.
05	TL05 - INVALID FUNCTION COLUMN NUMBER. Column number must be between 0 and the number of columns in the table.
06	TL06 - ERROR IN TABLE XXXXXXXXX. There is an error in the table definition. There are two likely reasons for getting this message: ■ The module named in the message is not a table that was defined using the M5TKTBLU macro. If a table and some other module have the same module name, then it is possible that the wrong module was loaded by M5TKTAB. ■ The table is defined as an ordered table, but its entries are not in ascending sequence.
07	TL07 - ARGUMENT NOT NUMERIC. A zoned argument is not numeric.

Figure 4-18 M5TKTAB Return Codes and Messages

M5TKTAB Example

This example uses the table definition in [Table Definition Example on page 4-30](#). We assume that TLARGVAL is defined as a fixed point (F) field with length 2 and that TLFUNVAL is defined as a character (C) field with length 3. Note that in this example, the starting location of field TLFUNVAL must be 2 bytes past the starting location of field TLARGVAL.

```
;FILL IN RCB
;
LET W.TLTABNAM = 'MONTHTAB'           ;NAME OF TABLE
LET W.TLARGCOL = 0                     ;USE DIRECT SEARCH
LET W.TLFUNCOL = 2                     ;RETURN COLUMN 2
LET W.TLARGVAL = 6                     ;LOOK FOR MONTH 6
;
;PUT RCB OFFSET IN CCB AND CALL M5TKTAB
;
LET W.CCBRCBDS = 1545
CALL MODULE M5TKTAB
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, TLFUNVAL WOULD CONTAIN JUN
```

How To Define Tables for M5TKTAB

You define tables that you want to use with M5TKTAB by using the M5TKTBLU assembler language macro instruction. Each table is assembled and linked as a separate load module with a corresponding entry in the CICS Processing Program Table (PPT). The JCL used to assemble and link edit a table is described in [Assemble and Link Edit M5TKTAB Tables on page 4-34](#).

You must supply a minimum of three M5TKTBLU macros to define a table.

Use the first M5TKTBLU macro (TYPE=INITIAL) to specify the table name, the number of columns in the table, the length and type of data in each column, and the value to return for each column if an entry is not found. Also indicate whether the table is ordered, unordered, or direct.

Create an *ordered* table so that it is in ascending sequence on its first column. M5TKTAB uses a binary search when the argument is the first column of an ordered table. If the argument is not the first column, then a sequential search is used.

M5TKTAB uses a sequential search on any column of an unordered table.

You can access a *direct* table by row number. If the argument column is zero, then the argument value is the row you want. It must be between 1 and the number of rows in the table. If the row number is out of range, then the return code (in the CCB) will indicate that the search failed.

A direct table can be used like an unordered table. If the argument column is not zero, then M5TKTAB uses a sequential search and returns the specified function column. If the function column is zero, then the function value is the row number containing the value.

Use the last macro (TYPE=FINAL) to indicate the end of the table.

Use the macros between the first and the last (TYPE=ENTRY) to supply the contents of the table.

[Figure 4-19](#) describes the parameters used with M5TKTBLU TYPE=INITIAL.

[Figure 4-20](#) describes the parameters used with TYPE=ENTRY.

[Figure 4-21](#) describes the parameters used with TYPE=FINAL.

Label	Operation	Operands
Name	M5TKTBLU	TYPE=INITIAL ,COLUMNS=(Size ₁ ,Type ₁ ,S ₂ ,T ₂ ,S ₃ ,T ₃ ,S ₄ ,T ₄) [,NOTFND=(V ₁ ,V ₂ ,V ₃ ,V ₄)] [,OPTION={ <u>ORDERED</u> UNORDERED DIRECT}]
Parameter	Description	
Name	The name of the table. This must be the same as the load module name and the name entered in the CICS PPT. This parameter is required.	
TYPE	This parameter indicates the start of the table and is required.	
COLUMNS	This parameter is required. You use it to specify the sizes and types for up to four data columns. Each column requires two entries: size and type. Entries are separated by commas.	
NOTFND	Use this parameter to supply the result values that should be returned for each column when a search fails. Values for each column are separated by commas, and values for character (C or A) fields must be enclosed in single quotation marks. This parameter is optional. However, if you elect to supply values, then you must supply a value for each defined column or skip the column by coding a comma. If no value is specified for a column, then question marks (?) are used. If the result field is numeric, VISION:Transact invalidates the field if it is used.	
OPTION	Use this parameter to indicate whether the table is ordered, unordered, or direct. Since the default is ordered, this parameter is only required when the table is unordered or direct. Important! If you omit this parameter or specify ordered, make sure that the column 1 values are in ascending sequence.	

Figure 4-19 M5TKTBLU Parameters Used With TYPE=INITIAL

Label	Operation	Operands
Name	M5TKTBLU	TYPE=ENTRY ,DATA=(V ₁ ,V ₂ ,V ₃ ,V ₄)
Parameter	Description	
TYPE	TYPE=ENTRY need not be specified since this is the default for the M5TKTBLU macro. This form of the macro is used to define a row in the table. Note that the assembler statement's label field must be blank.	
DATA	This parameter is required. Use this parameter to supply the values for each column in the table. Values for each column are separated by commas and values for C fields must be enclosed in single quotation marks. If you skip a column by coding a comma, then the row will contain question marks (?) in the column that you skipped.	

Figure 4-20 M4TKTBLU Parameters Used With TYPE=ENTRY

Label	Operation	Operands
	M5TKTBLU	TYPE=FINAL
Parameter	Description	
TYPE	This parameter is required. This form of the macro is used to end the table. No other parameters may be coded. Note that the assembler statement's label field must be blank.	

Figure 4-21 M5TKTBLU Parameters Used With TYPE=FINAL

Type Code	Internal Code	Assembler Data Type	Description
P	0	P	Packed field. The actual length is set to $(\text{size} / 2) + 1$, where $(\text{size} / 2)$ is truncated to an integer value. For example, if you specify a size of 3 for a P field, the length of the field is 2 because $(3 / 2) + 1 = 1 + 1 = 2$
B or F	1	F	Binary field. The length is determined by applying the following rules: ■ If $\text{size} < 3$, then $\text{length} = 1$. ■ If $\text{size} < 5$, then $\text{length} = 2$. ■ If $\text{size} < 8$, then $\text{length} = 3$. ■ Otherwise, $\text{length} = 4$
Z	2	C	Unsigned zoned field. The data entries must be numeric digits with no signs or decimal points. The data is represented in character format which means that the sign configuration is always X'F'. The length of the field is the same as the size that you specify. All data entries should be the same length or unexpected results may be obtained. Pad with leading zeroes where necessary.
C	3	C	Character field. The length of the data is the same as the size you specify.
H	4	X	Hexadecimal field. The data entries must be supplied as hexadecimal digits. Two hexadecimal digits are required to specify a single byte. The length of the data is set to $(\text{size} + 1) / 2$ which is truncated to an integer value. For example, if you specify size as 4, then the length of the data is $(4 + 1) / 2 = 5 / 2 = 2$ after truncation
T	6	B	Bit field. The data entries must be supplied in bit format which is a sequence of zeroes and ones. The length of the data is set to $(\text{size} - 1) / 8 + 1$, where $(\text{size} - 1) / 8$ is truncated to an integer. For example, if you specify size as 8, then the length of the data is $(8 - 1) / 8 + 1 = 7 / 8 + 1 = 1$.

Figure 4-22 M4TKTBLU Data Types (Page 1 of 2)

Type Code	Internal Code	Assembler Data Type	Description
S	12	P	Packed field. This is the same as data type P.
A	32	C	Character field. This is the same as data type C.
Y	33	Z	Signed zone field. Data is stored in zoned format and data entries may have a leading plus (+) or minus (-) sign.

Figure 4-22 M4TKTBLU Data Types (Page 2 of 2)

Table Definition Example

[Figure 4-24](#) is an example of a table definition. The table is a direct table with three columns. It contains months, 3-character abbreviations for months, and the number of days in each month. The comments at the start of the example are optional. They illustrate a method of documenting a table. Normally, you would also want to include comments describing any changes made to the table.

```

*   TABLE DEFINITION FOR VISION:TRANSACT TOOLKIT (M5TKTAB)
*
*   SOURCE MEMBER           : MONTHTAB
*   LOAD MODULE NAME       : MONTHTAB
*   CICS PPT NAME          : MONTHTAB
*   TABLE TYPE            : DIRECT - INDEXED BY MONTH
*
*   COL   TYPE   LEN   CONTENTS
*   ---   ---   ---   -----
*
*   1     C      9     MONTH
*   2     C      3     MONTH ABBREVIATION
*   3     Z      2     DAYS IN MONTH (NOT LEAP YEAR)
*
*
*   PRINT GEN
MONTHTAB M5TKTBLU TYPE=INITIAL, COL 72
          COLUMNS=(9,C,3,C,2,Z), X
          NOTFND=('*****','***','**'), X
          OPTION=DIRECT
          M5TKTBLU DATA=('JANUARY ','JAN',31)
          M5TKTBLU DATA=('FEBRUARY ','FEB',28)
          M5TKTBLU DATA=('MARCH ','MAR',31)
          M5TKTBLU DATA=('APRIL ','APR',30)
          M5TKTBLU DATA=('MAY ','MAY',31)
          M5TKTBLU DATA=('JUNE ','JUN',30)
          M5TKTBLU DATA=('JULY ','JUL',31)
          M5TKTBLU DATA=('AUGUST ','AUG',31)
          M5TKTBLU DATA=('SEPTEMBER','SEP',30)
          M5TKTBLU DATA=('OCTOBER ','OCT',31)
          M5TKTBLU DATA=('NOVEMBER ','NOV',30)
          M5TKTBLU DATA=('DECEMBER ','DEC',31)
          M5TKTBLU TYPE=FINAL
          END

```

Figure 4-23 Table Definition Example

How Tables Are Searched

M5TKTAB uses a binary search when searching on column 1 of an ordered table. A binary search will not work correctly if the values in column 1 are not in ascending sequence *as interpreted by M5TKTAB*.

The field comparison rules used by M5TKTAB must be considered when defining ordered tables.

The rules used to compare fields vary depending on the field type:

- Character (C or A) fields are compared logically using an assembler CLC (Compare Logical) instruction.
- When zoned (Z or Y) numbers are compared, they are first packed and then compared using an assembler CP (Compare Packed) instruction. This is an arithmetic comparison that takes the signs of the numbers into consideration.
- Packed (P or S) fields are compared using the CP instruction.
- Binary (F or B) fields are treated the same as character fields. They are compared logically using the CLC instruction. As long as only non-negative numbers are involved, the meaning of ascending conforms to your intuitive notions. However, if negative numbers are involved, then intuitive notions about the meaning of ascending no longer apply. For example, if you want an ordered table that contains the binary numbers from -5 to 5, the entries must be specified in the following sequence: 0, 1, 2, 3, 4, 5, -5, -4, -3, -2, -1.
- Hexadecimal (H) and Bit (T) fields are treated the same as character fields. They are compared logically using the CLC instruction.

M5TKTAB uses a sequential search when an unordered table is searched and when columns 2 through 4 of an ordered table are searched. A sequential search is also used when a direct table is searched on columns 1 to 4.

Table Structure

A table generated by the M5TKTBLU macro has a three distinct parts:

- The first part is a table header. The table header is described in [Figure 4-24](#).
- Following the table header, is a column information block for each defined column. The column information block is described in [Figure 4-25](#).
- Finally, there is the data for each column. Each column's data is stored separately with the default value (the value that is returned on a "not found" condition) as the first value for each column. A single X'FF' character is stored immediately after the last data entry in each column. The original intent was to use this as an end of data marker. However, a table may now contain X'FF' characters, so the X'FF' at the end of the column is no longer used to mark the end of the column and is present for compatibility only.

Assembler Declaration	Description
CL80	Copyright notice. Includes name of table.
C	Flag byte. Flag values are: ■ B'10000000' Table is unordered ■ B'01000000' Table is ordered ■ B'00100000' Direct lookup The following may be present but are ignored by M5TKTAB: ■ B'00001000' Lookup only ■ B'00000100' Verify only ■ B'00000010' Lookup and verify
AL1	Number of columns defined in this table (between 1 and 4).
AL2	Reserved for future use. Always zero.

Figure 4-24 M5TKTBLU Generated Table Header

Assembler Declaration	Description
A	Address of first data value for this column, the default data value.
H	Width of column. This is the width of a single data value for this column. Every data value in the column has the same width.
H	Number of data values in this column. This value is the same for all columns and does not include the default data entry.
C	Data type for this column. ■ X'03' Character (C) ■ X'02' Zoned (Z) ■ X'00' Packed (P) ■ X'01' Fixed Binary (B) ■ X'06' Bits (T) ■ X'0C' Packed (S) ■ X'04' Hexadecimal (H) ■ X'21' Zoned (Y) ■ X'20' Character (A)
AL1	Column size as specified on the M5TKTBLU TYPE=INITIAL macro. In general, this is not the same as column width.
AL2	Reserved for future use. Always binary zero.

Figure 4-25 M5TKTBLU Generated Column Information Block

Assemble and Link Edit M5TKTAB Tables

Tables used with M5TKTAB must be assembled and link edited before they can be used. A JCL PROC to do this is in source library member ASMTAB as shown in [Figure 4-26](#). You will need to modify the PROC to use your data set names.

```
//jobname JOB
//*
//*****
//*
//*      JCL TO ASSEMBLE AND LINKEDIT TABLES FOR M5TKTAB
//*
//*****
//*
//ASMTAB PROC LOADMOD=,          LOAD MODULE NAME
//      LOADLIB='your.table.loadlib',  LOAD LIBRARY NAME
//      SRCLIB='your.table.srclib',    TABLE SOURCE LIBRARY
//      SRCMBR=,                      TABLE MEMBER NAME
//      MACLIB='your.toolkit.srclib',  TOOL KIT SOURCE LIB
//      PRINTQ='*',                  SYSOUT Q FOR PRINT
//      SYSDA=SYSDA                  SYSTEM DASD UNIT TYPE
//*
//*****
//*      ASSEMBLE THE TABLE
//*
//*****
//ASM      EXEC PGM=ASMA90,REGION=4M,TIME=(2,00),
//          PARM=(NOTERM,RENT,NODECK,OBJECT,'SYSPARM(OS)',
//          LIST,RDL,'LINECOUNT(58)','XREF(FULL)')
//SYSPRINT DD SYSOUT=&PRINTQ
//SYSLIN   DD DSN=&&SYSLIN,DISP=(NEW,PASS),
//          UNIT=&SYSDA,SPACE=(TRK,(2,2)),
//          DCB=BLKSIZE=27920
//SYSUT1   DD UNIT=&SYSDA,SPACE=(CYL,(1,1))
//SYSLIB   DD DISP=SHR,DSN=&MACLIB
//SYSIN    DD DISP=SHR,DSN=&SRCLIB(&SRCMBR)
//*
//*****
//*      LINKEDIT THE TABLE
//*
//*****
//*
//LKED     EXEC PGM=IEWL,PARM='LET LIST,MAP,NCAL,RENT',REGION=4M
//SYSPRINT DD SYSOUT=&PRINTQ
//SYSLMOD  DD DISP=SHR,DSN=&LOADLIB(&LOADMOD)
//SYSUT1   DD UNIT=&SYSDA,SPACE=(CYL,(1,1))
//SYSLIN   DD DSN=&&SYSLIN,DISP=(OLD,DELETE)
//          PEND
//*
//*****
//*
//          EXEC ASMTAB,SRCMBR=tabsrc,LOADMOD=tabload
//
```

Figure 4-26 Assemble and Link Edit M5TKTAB Tables

M5TKTCT

Description

M5TKTCT is used to obtain the cursor position (row and column) on the application's input screen.

[Figure 4-27](#) describes the routine control block used by this subroutine. For compatibility with previous versions of the Toolkit and existing VISION:Transact applications, some obsolete data fields are still supported. We recommend that new applications use the alternate sources, specified in the table, for the data provided in these fields.

[Figure 4-28](#) lists the return codes and messages issued by this subroutine. A zero return code indicates a successful operation.

Field Name	Field Length	Field Type	Field Description
TCTERMID	4	C	CICS terminal ID of the terminal to which the transaction is attached. OBSOLETE - Use the VISION:Transact flag field TERMID.
TCTRANID	4	C	CICS transaction ID of the transaction that started the VISION:Transact application. OBSOLETE - Use the VISION:Transact flag field TRANCODE.
TCTIME	4	P	Current time in the format: 0HHMMSS+. OBSOLETE - Use the VISION:Transact flag field TIME.
TCOPERID	3	C	CICS operator ID of signed on operator. OBSOLETE - Use the VISION:Transact flag field OPERID.
TCOPERSC	3	C	CICS operator security key. OBSOLETE - CICS Release 3.2.1 does not support this function. Blanks will be returned.
TCDATEJ	6	C	Today's date in Julian format: YY/DDD. OBSOLETE - Use the VISION:Transact flag field JULIAN.
TCDATEG	8	C	Today's date in Gregorian format: MM/DD/YY (or DD/MM/YY). OBSOLETE - Use the VISION:Transact flag field TODAY.
TCCURROW	2	F	Row position of the cursor on the application's input screen.
TCCURCOL	2	F	Column position of the cursor on the application's input screen.

Figure 4-27 M5TKTCT Routine Control Block

Return Code	Message
01	M5TKTCT - INCORRECT RCB DISPLACEMENT IN CCB.
02	TC02 - MODULE M5TKDATG IS DISABLED OR NOT DEFINED IN THE CICS PPT
	M5TKTCT uses M5TKDATG to convert the Julian date to a Gregorian date.

Figure 4-28 M5TKTCT Return Codes and Messages

Example

```

;NOTHING IS MOVED TO THE RCB BEFORE THE CALL
;
;PUT RCB OFFSET IN CCB AND CALL M5TKTCT
;
LET W.CCBRCBDS = 2067
CALL MODULE M5TKTCT
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
; THE RCB FIELDS (INCLUDING THE OBSOLETE ONES)
; CONTAIN THE APPROPRIATE DATA.

```

M5TKTS

Description

M5TKTS creates, updates, and deletes records in CICS temporary storage queues. A temporary storage queue may contain one or more records.

Note that VISION:Transact automatically supports the use of CICS temporary storage queues when working storage and/or pseudo files are saved between interactions. VISION:Transact automatically retrieves temporary storage queue records at the beginning of the application execution cycle and writes (creates, updates, deletes) the records at the end of the execution cycle. The M5TKTS Toolkit may be useful if records in a temporary storage queue need to be created, updated, or deleted at specific points during the application execution cycle.

[Figure 4-29](#) describes the routine control block used by this subroutine.

[Figure 4-30](#) lists the return codes and messages issued by this subroutine. A zero return code indicates that the operation was successful.

Field Name	Field Length	Field Type	Field Description
TSSTGID	8	C	Temporary storage ID for the queue to be processed. Some kind of naming standard should be used to prevent duplicate storage IDs. The easiest method is to use the terminal ID of the application that owns the storage as the first four bytes of the ID. The remainder of the ID would then distinguish between different queues owned by the same terminal.
TSSTGLEN	2	F	Length of the temporary storage record to be read or written. If TSSTGFUN is *, this value must match the length of the existing record. Length is ignored if TSSTGFUN is D.
TSSTGOFF	2	F	Displacement (offset) of the working storage area holding the data to be read or written. This displacement is from the beginning of the common control block.
TSSTGENT	2	F	Entry number (record number) of the record to be processed. When you create a record, CICS assigns the actual entry number and it is returned in this field. If this field is zero, M5TKTS sets it to 1 if TSSTGFUN is U or *.
TSSTGFUN	1	C	Function to be performed. Valid values are: A - Add a temporary storage record. Entry number is returned in TSSTGENT. U - Update an existing temporary storage record. Place the entry number in TSSTGENT. D - Delete a temporary storage queue. The entire queue is deleted. Individual records cannot be deleted from a queue. * - Retrieves the record whose entry number is in TSSTGENT.

Figure 4-29 M5TKTS Routine Control Block

Return Code	Message
01	M5TKTS - INCORRECT RCB DISPLACEMENT IN CCB.
02	TS02 - INVALID OPTION. VALID ENTRIES ARE '*', 'A', 'U' AND 'D'.
03	TS03 - INSUFFICIENT SPACE ON TEMPORARY STORAGE DATA SET.
04	TS04 - TEMPORARY STORAGE QUEUE NOT FOUND. This error message is displayed if the function is U, D, or * and the queue does not exist.
05	TS05 - AN UNRECOVERABLE INPUT/OUTPUT ERROR HAS OCCURRED.
06	TS06 - INVALID REQUEST CONDITION HAS BEEN DETECTED. TS RECORD NOT PROCESSED. Possible causes: ■ The record may have been written with DFHTS TYPE=PUT macro as opposed to TYPE=PUTQ. (Only applies to versions of CICS that support MACRO LEVEL programming). ■ The queue is awaiting ISC session recovery. ■ You are not authorized to access the queue
07	TS07 - ENTRY NUMBER INVALID OR TEMPORARY STORAGE RECORD NOT FOUND. You get this error if the function is U or * and the record number you asked for does not exist.
08	TS08 - INVALID TEMPORARY STORAGE LENGTH. The length you specified is less than 1 or the function is * and the record length you specified is less than the existing record's length.
09	TS09 - INVALID TEMPORARY STORAGE AREA DISPLACEMENT. The data displacement is such that the data area overlaps the Toolkit CCB.

Figure 4-30 M5TKTS Return Codes and Messages

Example

This example assumes that the temporary storage queue whose name is THEQUEUE contains at least three records and that the third record is 128 bytes long.

```
;FILL IN RCB
;
LET W.TSSTGID = 'THEQUEUE'      ;NAME OF QUEUE
LET W.TSSTGLEN = 128            ;LENGTH OF RECORD
LET W.TSSTGOFF = 2374           ;RECORD OFFSET FROM CCB
LET W.TSSTGENT = 3              ;THE 3RD RECORD
LET W.TSSTGFUN = '*'            ;READ THE RECORD
;
;PUT RCB OFFSET IN CCB AND CALL M5TKTS
;
LET W.CCBRCBDS = 2103
CALL MODULE M5TKTS
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, THE 128 BYTES STARTING AT OFFSET
;2374 FROM THE START OF THE CCB WILL CONTAIN THE DATA
;FROM THE TEMPORARY STORAGE RECORD THAT WAS RETRIEVED.
```

M5TKUSRA

Description

M5TKUSRA provides access to the CICS Terminal Control Table User Area (TCTUA). The TCTUA may be used to save data between CICS transactions and to pass data to other transactions using the same terminal.

[Figure 4-31](#) describes the routine control block used by this subroutine.

[Figure 4-32](#) lists the return codes and messages for this subroutine. A zero return code indicates the operation was successful.

Field Name	Field Length	Field Type	Field Description
USFUNC	1	C	Function to be performed. Valid values are: U - Copies data from USDATA to the TCTUA. D - Deletes (clears) data in the TCTUA. The TCTUA is filled with binary zeroes. * - Copies data from the TCTUA to USDATA
USDATA	n	C	Data copied to or from the TCTUA. The length of the data, n, must be the same as the length of the TCTUA.

Figure 4-31 M5TKUSRA Routine Control Block

Return Code	Message
01	M5TKUSRA - INCORRECT RCB DISPLACEMENT IN CCB.
02	US02 - INVALID OPTION. VALID ENTRIES ARE '*', 'U' OR 'D'
03	US03 - NO USER AREA DEFINED IN TCT.

Figure 4-32 M5TKUSRA Return Codes and Messages

Example

For this example we assume that the working storage area that immediately follows USFUNC (USDATA) already contains nnn bytes of data that are to be stored in the CICS TCTUA, where nnn is the length of the TCTUA.

```

;FILL IN RCB
;
LET W.USFUNC = 'U'           ;UPDATE THE TCTUA
;
;PUT RCB OFFSET IN CCB AND CALL M5TKUSRA
;
LET W.CCBRCBDS = 2118
CALL MODULE M5TKUSRA
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, THE TCTUA CONTAINS THE DATA THAT
;WAS IN USDATA. USDATA REMAINS UNCHANGED.

```

M5TKVLID

Description

M5TKVLID performs field validation. There are predefined validation symbols for various date, alphanumeric, and numeric field formats. You can define additional symbols.

The following kinds of validation are predefined:

- Field content validation based on an edit pattern.
- Gregorian and Julian date validation for dates in American or European format.
- Modulo 11 check digit validation.

[Figure 4-33](#) describes the routine control block used by this subroutine.

[Figure 4-34](#) lists the return codes for this subroutine. A zero return code indicates the validation was successful; i.e., the field content matched the edit pattern. Only return code -4 displays a message; the other return codes do not.

[Figure 4-35](#) lists the validation symbols that you can use to create edit patterns.

[Appendix A, “Defining Validation Symbols for M5TKVLKID”](#) shows you how to change existing validation symbols, add your own, or change the delimiter character. Note that the delivered delimiter character is # but can be changed if desired.

Field Name	Field Length	Field Type	Field Description
VLDATA	255	C	The field to be validated.
VLPATT	255	C	Edit pattern used to validate VLDATA. Has the format $S_1S_2S_3 \dots S_n\#$ where S_n is a validation symbol and # is the delimiter character. The delimiter is used to indicate the end of the edit pattern. If the pattern is 255 characters long and there is no room for the delimiter in this field, then put the delimiter in VLDELIM or let M5TKVLID do it for you.
VLDELIM	1	C	Delimiter character. Indicates end of edit pattern. Used when VLPATT is 255 bytes long and has no room for the delimiter. M5TKVLID automatically places the delimiter in this field each time it is called.

Figure 4-33 M5TKVLID Routine Control Block

Return Code	Message
1 to n	Validation not OK. The return code value indicates the position of the field which did not match the edit pattern.
-1	Validation not OK. The edit pattern is invalid or the delimiter is missing.
-2	Validation not OK. The date is invalid. Only when validating a Gregorian or Julian date.
-3	Validation not OK. The digit check failed or the field is not numeric. Only when validating a check digit.
-4	M5TKVLID - INCORRECT RCB DISPLACEMENT IN CCB

Figure 4-34 M5TKVLID Return Codes and Messages

Symbol	Dash	Validation Performed
Z	Y	The letters A-Z.
B	Y	Blank.

Figure 4-35 M5TKVLID Validation Symbols (Page 1 of 5)

Symbol	Dash	Validation Performed
A	Y	The letters A-Z or blank.
9	Y	Digits 0-9.
1	Y	Digits 0-9 or blank.
X	Y	The letters A-Z, digits 0-9, or blank.
Y	Y	The letters A-Z or digits 0-9.
C	N	Any character (no validation).
Literal	Y	literal is one or more characters enclosed in single quotation marks. Corresponding characters in VLDATA must match the literal. For example, 99'-'999# matches input that contains two digits, a dash, and then three digits. The # is the delimiter. A single quotation mark can be included in a literal by using two consecutive single quotation marks within the literal. For example, 99'''' matches input that contains two digits followed by one single quotation mark.
D	N	Gregorian date validation. Date is in format MM/DD/YY (or DD/MM/YY). Implies a field length of 8.
F	N	Gregorian date validation. Date is in format MMDDYY (or DDMMYY). Implies a field length of 6.
G	N	Julian date validation. Date is in format YY/DDD. Implies a field length of 6.
H	N	Julian date validation. Date is in format YYDDD. Implies a field length of 5.
E..*	N	Check digit validation. The digits used to compute the check digit are represented by Es. The check digit itself is represented by *. For example, EEEE* means that four digits are used to compute the fifth digit which is the check digit. The algorithm used to calculate the check digit is described in Check Digit Calculation on page 4-47 .

Figure 4-35 M5TKVLID Validation Symbols (Page 2 of 5)

Symbol	Dash	Validation Performed
K	N	Numeric field validation. The number of Ks determines the length of the numeric field. The field can contain leading and trailing spaces, an optional leading plus or minus sign, numeric digits, and an optional decimal point character. The decimal point character can be a period (.) or a comma (,). If the field contains a sign, it must be the first non-blank character in the field, and it must be followed by a digit or a decimal point character. If the field contains a decimal point character, it must be followed by at least one digit. The field cannot contain embedded blanks.

Figure 4-35 M5TKVLID Validation Symbols (Page 3 of 5)

Symbol	Dash	Validation Performed
-	N/A	A dash (minus sign) can be used to reverse the effect of the next validation symbol or literal. A dash can be used only with those validation symbols that have a Y in the dash column as shown in this table. If it is used with any other symbol, it is ignored, except when it is used before the C (any character) symbol. -C is treated as an error because it indicates that no character can match the pattern. For example: ■ -9 means that any input character is acceptable except a digit. ■ -B means any non-blank character is acceptable. ■ -99 means any non-digit followed by a digit is acceptable. ■ -9-9 means any two non-digits are acceptable. ■ - ' A ' means any character except A is acceptable. ■ 99-D and 99D both validate for two digits followed by a Gregorian date because the dash is ignored. ■ - ' ABC ' means that any string of three characters that does not have A in the first position, B in the second position, or C in the third position is acceptable. Note that - ' ABC ' is equivalent to - ' A ' - ' B ' - ' C ' .

Figure 4-35 M5TKVLID Validation Symbols (Page 4 of 5)

Symbol	Dash	Validation Performed
User Defined	?	The user specifies the acceptable characters for the symbol and/or specifies a user coded validation routine (see Appendix A, "Defining Validation Symbols for M5TKVLKID"). The ? in the dash column means that the ability to reverse the effect of the validation symbol depends on how the symbol is defined. If the user does not supply a validation routine (the symbol simply represents some set of acceptable characters), then a dash before the symbol is automatically supported. However, if the symbol has an associated user coded validation routine, then the leading dash is not automatically supported. The user can code the routine to support the dash or not support it. If the dash is not supported by the user coded routine, it is ignored.

Figure 4-35 M5TKVLID Validation Symbols (Page 5 of 5)

Check Digit Calculation

Initialize multiplier to 2 and accumulator to zero.

Starting with digit to the left of the check digit, process all of the digits in the field from right to left:

- Multiply digit by multiplier.
- Add result to accumulator.
- Add 1 to multiplier.
- If multiplier is >11, reset it to 2. (The multiplier cycles from 2 to 11.)

After all digits have been processed, get the remainder from accumulator divided by 11. (Calculate accumulator modulo 11.)

Subtract the remainder from 11.

The result of the subtraction will be a value between 1 and 11. If the value is greater than 9, set it to 0.

The result of all the above calculations is the check digit.

For example, the check digit for 1735 is calculated as follows:

```
Accumulator = (5 * 2) + (3 * 3) + (7 * 4) + (1 * 5) = 52
Remainder of (52 / 11) = 8
11 - remainder = 11 - 8 = 3
The check digit is 3.
```

Edit Pattern Examples

In the following examples, the pattern delimiter is assumed to be the # character.

ZZ#	Accepts two letters.
XXXX#	Accepts four letters, digits, or blanks in any combination.
'ABC-'9#	Accepts the string ABC- followed by a digit.
999-'A'#	Accepts three digits followed by any character except A.
'DATE:'D#	Accepts the string DATE: followed by a date in MM/DD/YY (or DD/MM/YY) format.
H#	Accepts a date in the format YYDDD.
G#	Accepts a date in the format YY/DDD.
KKKKKK#	Accepts a numeric field up to 6 characters long, such as 123.45, 1.234, -5.789, 2, 12.

Example

```
;FILL IN RCB
;
LET W.VLDATA = 'AB123-Q76'      ;DATA TO CHECK
;
;THE PATTERN IN THE RCB CHECKS FOR:
; ZZ - 2 LETTERS
; 999 - FOLLOWED BY 3 DIGITS
; '-' - FOLLOWED BY A DASH
; Y - FOLLOWED BY A LETTER OR DIGIT
; 99 - FOLLOWED BY 2 DIGITS
; B - FOLLOWED BY BLANK
; I - FOLLOWED BY BLANK OR DIGIT
; # - (DELIMITER) END OF PATTERN
;
LET W.VLPATT = 'ZZ999'-'Y99BI#'
;
;PUT RCB OFFSET IN CCB AND CALL M5TKVLID
;
LET W.CBRCBDS = 2374
CALL MODULE M5TKVLID
;
; < CODE TO TEST THE RETURN CODE IN THE CCB GOES HERE >
;
;IN THIS EXAMPLE, THE DATA MATCHES THE PATTERN.
```

Defining Validation Symbols for M5TKVLKID

Introduction

The M5TKVLID subroutine has three main sections:

- The first section consists of the predefined validation routines for M5TKVLID along with the constants and variables used by the routines. This section is generated by the VALIDGEN macro.
- The second section is optional. It contains user coded validation routines, if any.
- The third section contains the definitions of the validation symbols used with M5TKVLID. The VALIDTAB macro is used to generate the definition for a validation symbol.

If you add validation symbols or user coded validation routines, you must assemble and link edit M5TKVLID. The JCL is described in [Assemble and Link Edit Toolkit Programs \(Optional\) on page 2-2](#).

The VALIDGEN Macro

Parameters for the VALIDGEN macro are described in this section. Optional parameters are enclosed in brackets[]

Label	Operation	Operands
	VALIDGEN	[DELIMTR=#] [,MAXTABL=16] [,DECCHAR=.]

Parameter	Description
DELIMTR	Use this parameter to specify the single, non-blank character that is to be used as the pattern delimiter. This character is used to indicate the end of the pattern and defaults to the # character. The delimiter cannot be an apostrophe (') or any character that is used as a validation symbol
MAXTABL	Use this parameter to specify the maximum number of VALIDTAB macros that can follow the VALIDGEN macro. The default is 16.
DECCHAR	Use this parameter to specify the character that is used as the decimal point character for numeric field validation (validation symbol K). The default is the period (.). If you specify any other value, the comma (,) is used as the decimal point.

The VALIDTAB Macro

This section shows the parameters for the VALIDTAB macro. Optional parameters are enclosed in brackets []. Alternate values are enclosed in braces { } and separated by vertical bars |. Default values are underlined.

This macro performs two functions:

- It associates a validation symbol with an M5TKVLID routine. The routine associated with a symbol may be one of the pre-coded routines or a user coded routine. The pre-coded VALCHAR routine is the one that checks to see if a data character is one of the characters corresponding to the validation symbol.
- When necessary, it creates a 256-byte translate table for the validation symbol. Characters that correspond to the validation symbol translate to X'00'. All other characters translate to X'01'. The VALCHAR routine uses such a table to determine if a data character ``matches" a validation symbol.

If the M5TKVLID routine associated with a validation symbol does not need a translate table, then no table should be generated for the symbol. For example, the check digit validation routine (associated with E...*) does not use a translate table to verify that the data characters it is working on are all numeric. That is the way the routine is coded; it could have used a translate table to check for numerics.

Label	Operation	Operands
Ts	VALIDTAB	(C ₁ ,...,C _n) [,ROUTINE={ <u>VALCHAR</u> name}] [,TABLE={ <u>YES</u> NO}]

Parameter	Description
Ts	The letter T followed by s, where s is the validation symbol that you are defining. For example, TX would be used to define X as a validation symbol. The characters that you can use as validation symbols are the letters A through Z, the digits 0 through 9, and the characters \$, #, and @. (Note that # is the default delimiter and the delimiter cannot be used as a validation symbol.)
C ₁ ,...,C _n	This is a list containing the set of characters that correspond to the validation symbol s. The characters in the list are separated by commas and the entire list is enclosed in parentheses. The list may not contain any embedded spaces. To include characters, such as space, that cannot be directly coded in the list, use the hexadecimal representation of the character. For example, 40 can be used to include "space" in the list. Do not enclose any entries in quotation marks. The list must contain at least one character even when TABLE=NO is specified
ROUTINE	Names the routine that will be used to validate the data field when the validation symbol s is encountered in the pattern. Omit this parameter or specify VALCHAR if you are defining a validation symbol that corresponds to the set of characters in the (C ₁ ,...,C _n) parameter.
TABLE	Omit this parameter or use TABLE=YES to generate a translate table based on the values specified in the (C ₁ ,...,C _n) parameter. Use TABLE=NO to suppress table generation if the routine (named by the ROUTINE parameter) does not need a translate table. The only pre-coded routine that uses a translate table is VALCHAR.

Example

```
*
*  DEFINE SYMBOL L WHICH MATCHES A, B, OR BLANK
*
TL      VALIDTAB (A,B,40)
*
*  DEFINE SYMBOL S WHICH IS PROCESSED BY THE USER-CODED
*  VALIDATION ROUTINE "YOURRTN"
*
TS      VALIDTAB (S),ROUTINE=YOURRTN,TABLE=NO
```

Coding Validation Routines

You can add your own validation routines to M5TKVLID. We suggest that you put any routines that you code into a separate source file and then use the assembler COPY command to include the file in M5TKVLID. This will make it easier to incorporate your routines in new releases of the Toolkit.

The COPY command should be placed after the VALIDGEN macro and before the first VALIDTAB macro. If you copy in a large routine, you may have to allocate a second base register for the program.

Register usage is as follows:

- Register 11 is the base register for the CICS EIB.
- Register 12 is the base register for M5TKVLID.
- Register 13 is the base register for dynamic storage (the DFHEISTG DSECT).
- Register 10 is the base register for the Toolkit CCB.
- Register 5 is the base register for the Toolkit RCB.
- FLDREG (equated to 3) points to the current character in the field being validated.
- PATREG (equated to 4) points to the current character in the edit pattern.
- Registers 0, 14, and 15 may be freely used as work registers.

If your routine processes *n* characters in the edit pattern, then you must add *n*-1 to PATREG.

Similarly, if your routine processes *m* characters in the field, then you must add *m*-1 to FLDREG.

If your routine adds to PATREG and/or FLDREG, you must ensure that you do not set these pointers beyond the end of the edit pattern or field.

Your routine may branch to label FIELDUPD to process the next character from the field and the edit pattern (PATREG and FLDREG will both be incremented by 1). To move to the next edit pattern character without incrementing the field pointer, branch to label MASKUPD (PATREG will be incremented by 1 and FLDREG will remain unchanged).

Each routine that you code should have the following general format:

```
YOURRTN EQU *      ENTRY POINT
      < CODE >
DFHEISTG DSECT
      < VARIABLES THAT BELONG IN DYNAMIC STORAGE >
M5TKVLID CSECT,     SWITCH BACK TO CSECT
```

You must not duplicate any labels. Examine the pre-coded validation routines to make sure that your labels are unique.

M5TKVLID Source Listing

```
*ASM XOPTS(NOPROLOG NOEPILOG)                                00010000
      TITLE 'M5TKVLID -- TRANSACT TOOLKIT FOR CICS'          00020001
*****                                                         00030000
*                                                         * 00040000
* C O N F I D E N T I A L   B U S I N E S S   I N F O R M A T I O N * 00050000
*                                                         * 00060000
*****                                                         00070000
*                                                         * 00080000
*      PROPERTY AND CONFIDENTIAL INFORMATION OF              * 00090000
*      COMPUTER ASSOCIATES INTERNATIONAL, INC.                * 00100000
*      USE RESTRICTED BY WRITTEN LICENSE AGREEMENT.           * 00110000
*      DO NOT REMOVE THIS NOTICE.                             * 00120000
*      (C) 2003 COMPUTER ASSOCIATES INTERNATIONAL, INC.       * 00130000
*      AS AN UNPUBLISHED WORK. ALL RIGHTS RESERVED.           * 00140000
*                                                         * 00150000
*****                                                         00160000
      EJECT                                                    00170000
*****                                                         00180000
*                                                         00190000
* DESCRIPTION                                                  00200000
*                                                         00210000
*      PERFORMS FIELD VALIDATION. USER SUPPLIES DATA TO BE   00220000
*      AND A VALIDATION PATTERN. THE DATA IS COMPARED TO THE  00230000
*                                                         00240000
*      THE VALIDATION PATTERN IS COMPOSED OF VALIDATION SYMBOLS THAT  00250000
*      ARE DEFINED BELOW USING THE VALIDTAB MACRO.              00260000
*                                                         00270000
*      A VALIDATION SYMBOL MAY REPRESENT A SET OF CHARACTERS OR IT  00280000
*      MAY BE ASSOCIATED WITH A SPECIAL VALIDATION ROUTINE.     00290000
*                                                         00300000
*      IF IT REPRESENTS A SET OF CHARACTERS, THEN THE CORRESPONDING  00310000
*      POSITION IN THE DATA FIELD MUST BE ONE OF THOSE CHARACTERS.  00320000
*                                                         00330000
*      IF IT IS ASSOCIATED WITH A SPECIAL VALIDATION ROUTINE, THEN  00340000
*      DATA IN THE INPUT FIELD IS EXAMINED TO SEE THAT IT MEETS  00350000
*      THE VALIDATION ROUTINE'S REQUIREMENTS.                  00360000
*                                                         00370000
* RETURN CODES AND MESSAGES                                     00380000
*      RC = 0 -> NO PROBLEMS ENCOUNTERED AND FUNCTION WAS PERFORMED.  00390000
*                                                         00400000
*      RC MESSAGE                                              00410000
*      -- -----                                              00420000
*      0 NO MESSAGE. RC MEANS VALIDATION WAS OK. THE FIELD CONTENT  00430000
*      MATCHED THE EDIT PATTERN.                                00440000
*      >0 NO MESSAGE. RC MEANS VALIDATION WAS NOT OK. THE RC VALUE  00450000
```

Figure A-1 M5TKVLID Source Listing (Page 1 of 2)

```

*          INDICATES THE FILED POSITION WHICH DID NOT MATCH THE      00460000
*          PATTERN.                                                  00470000
*      -1 NO MESSAGE. -1 MEANS THAT THE EDIT PATTERN IS INVALID OR    00480000
*          THE DELIMITER IS MISSING.                                  00490000
*      -2 NO MESSAGE. -2 MEANS THAT THE DATE IS INVALID. ONLY WHEN    00500000
*          VALIDATING A GREGORIAN OR JULIAN DATE.                     00510000
*      -3 NO MESSAGE. -3 MEANS THAT THE DIGIT CHECK FAILED OR THE     00520000
*          FIELD IS NOT NUMERIC. ONLY WHEN VALIDATING A CHECK DIGIT.  00530000
*      -4 M5TKVLID - INCORRECT RCB DISPLACEMENT IN CCB               00540000
*                                                                      00550000
* REVISIONS                                                           00560000
*                                                                      00570000
* MM/DD/YY REV# WHO WHY                                             00580000
* -----
* 07/05/92 000 KFM RECODED AS COMMAND LEVEL                         00590000
*                                                                      00600000
* *****                                                            00610000
* *****                                                            00620000
*          TKDATES DATE=MMDYY    SELECTS DATE FORMAT                00630000
*                                                                      00640000
*          VALIDGEN DELIMTR=#,MAXTABL=16,DECCHAR=.                  00650000
*                                                                      00660000
*          PRINT NOGEN                                               00670000
*          PRINT GEN                                                 00680000
* *****                                                            00690000
* *****                                                            00700000
*          ADDITIONAL CODE CAN BE APPENDED HERE                    * 00710000
*          PLEASE BEWARE OF DUPLICATE LABELS                       * 00720000
*          =====                                                * 00730000
*          =====                                                * 00740000
*          =====                                                * 00750000
*          =====                                                * 00760000
* *****                                                            00770000
*          EJECT                                                     00780000
TZ      VALIDTAB (A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z) 00790000
        SPACE 5                                                     00800000
TB      VALIDTAB (40)                                                00810000
        SPACE 5                                                     00820000
TA      VALIDTAB (A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z, 00830000
        40)                                                           00840000
        SPACE 5                                                     00850000
T9      VALIDTAB (0,1,2,3,4,5,6,7,8,9)                               00860000
        SPACE 5                                                     00870000
TI      VALIDTAB (0,1,2,3,4,5,6,7,8,9,40)                           00880000
        SPACE 5                                                     00890000
TX      VALIDTAB (A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z, *00900000
        0,1,2,3,4,5,6,7,8,9,40)                                     00910000
        SPACE 5                                                     00920000
TY      VALIDTAB (A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z, *00930000
        0,1,2,3,4,5,6,7,8,9)                                     00940000
        SPACE 5                                                     00950000
TD      VALIDTAB (D),TABLE=NO,ROUTINE=VALDATE1                      00960000
        SPACE 5                                                     00970000
TF      VALIDTAB (F),TABLE=NO,ROUTINE=VALDATE2                      00980000
        SPACE 5                                                     00990000
TG      VALIDTAB (G),TABLE=NO,ROUTINE=VALDATG1                      01000000
        SPACE 5                                                     01010000
TH      VALIDTAB (H),TABLE=NO,ROUTINE=VALDATG2                      01020000
        SPACE 5                                                     01030000
TE      VALIDTAB (E),TABLE=NO,ROUTINE=CHECKD11                     01040000
        SPACE 5                                                     01050000
TK      VALIDTAB (K),TABLE=NO,ROUTINE=DECIMALP                     01060000
*                                                                      01070000
*          END                                                       01080000

```

Figure A-1 M5TKVLID Source Listing (Page 2 of 2)

Sample File Definitions

TWA Filler

The following file definition defines the 4-byte filler required at the start of the TWA for VISION:Transact Release 5.0 and later.

```
TKFD01  FD          Y
TKFD01  FD          F  4
TKFD01  AA  *****
TKFD01  AA  *
TKFD01  AA  *      TWA FILLER (VISION:Transact RELEASE 5.0 AND LATER)  *
TKFD01  AA  *
TKFD01  AA  *****
TKFD01  LSTKWSFILL  1
TKFD01  LOTKWSFILL 11  1  4C1
```

Fields Used by the Toolkit

The following file definition defines the fields used by the Toolkit. The field names correspond to the names used in the rest of this document. In cases where a Toolkit field is of variable length, the maximum allowable length, 255, is used in the definition.

```
TKFD02  FD          Y
TKFD02  FD          F2897  2897
TKFD02  LSTKWS      1
TKFD02  AA  *****
TKFD02  AA  *
TKFD02  AA  *      COMMON CONTROL BLOCK
TKFD02  AA  *
TKFD02  AA  *****
TKFD02  LOCCBRCBDS  11  1  2F1
TKFD02  LOCCBRCOD   11  3  2F
TKFD02  LOCCBMSG    11  5  80C
TKFD02  AA  *****
TKFD02  AA  *
TKFD02  AA  *      M5TKBYTE/M5TKBIT ROUTINE CONTROL BLOCK
TKFD02  AA  *
TKFD02  AA  *****
TKFD02  LOBTBITS    11  85  1C
TKFD02  LOBTBYTES   11  86  8C
TKFD02  AA  *****
TKFD02  AA  *
TKFD02  AA  *      M5TKCOPY ROUTINE CONTROL BLOCK
TKFD02  AA  *
TKFD02  AA  *****
```

```
TKFD02 L0COFILNAM 11 94 8C
TKFD02 L0COKEYLEN 11 102 2F
TKFD02 L0COKEYOFF 11 104 2F
TKFD02 L0COOLDKEY 11 106255C
TKFD02 L0CONEWKEY 11 361255C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKDATJ/M5TKDATG ROUTINES CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 L0GJDATEJ 11 616 6C
TKFD02 L0GJDATEG 11 622 8C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKHEXA ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 L0XALEN 11 630 2F
TKFD02 L0XAFROM 11 632127C
TKFD02 L0XATO 11 759254C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKLJST/M5TKRJST ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 L0JSLEN 111013 2F
TKFD02 L0JSLEADTR 111015 1C
TKFD02 L0JSPAD 111016 1C
TKFD02 L0JSDATA 111017255C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKPREV ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 L0PVCOUNT 111272 2F
TKFD02 L0PVKEYLEN 111274 2F
TKFD02 L0PVGENLEN 111276 2F
TKFD02 L0PVFCTNAM 111278 8C
TKFD02 L0PVKEY 111286255C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKPRNT ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 L0PRTERMID 111541 4C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKTAB ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 L0TLTABNAM 111545 8C
TKFD02 L0TLARGCOL 111553 2F
TKFD02 L0TLFUNGCOL 111555 2F
TKFD02 L0TLARGVAL 111557255C
TKFD02 L0TLFUNVAL 111812255C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKTCT ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 L0TCTERMID 112067 4C
TKFD02 L0TCTRANID 112071 4C
TKFD02 L0TCTIME 112075 4P
TKFD02 L0TCOPERID 112079 3C
TKFD02 L0TCOPERS 112082 3C
TKFD02 L0TCDATEJ 112085 6C
TKFD02 L0TCDATEG 112091 8C
TKFD02 L0TCCURROW 112099 2F
TKFD02 L0TCCURCOL 112101 2F
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKTS ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
```

```

TKFD02 LOTSSTGID 112103 8C
TKFD02 LOTSSTGLEN 112111 2F
TKFD02 LOTSSTGOFF 112113 2F
TKFD02 LOTSSTGENT 112115 2F
TKFD02 LOTSSTGFUN 112117 1C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKUSRA ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 LOUSFUNC 112118 1C
TKFD02 LOUSDATA 112119255C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKVLID ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 LOVLDATA 112374255C
TKFD02 LOVLPATT 112629255C
TKFD02 LOVLDELIM 112884 1C
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * M5TKCOMM ROUTINE CONTROL BLOCK
TKFD02 AA *
TKFD02 AA *****
TKFD02 LOCMPRGNAM 112885 8C
TKFD02 LOCMLN 112893 2F
TKFD02 LOCMDISPL 112895 2F
TKFD02 AA *****
TKFD02 AA *
TKFD02 AA * THE END ...
TKFD02 AA *
TKFD02 AA *****
TKFD02 LOTHEEND 112896 1C

```


Index

A

assemble and link edit tables, 4-34
assemble and link edit Toolkit programs, 2-2

C

change date format, 2-1
Check Digit Calculation, 4-47
Check digit validation, 4-42
CICS COMMAREA, 4-3
CICS temporary storage, 4-37
CICS temporary storage queues, 4-37
CICS Terminal Control Table User Area, 4-40

D

date format, 2-1
Date validation, 4-42
defining tables, 4-25
definition example, 4-31

E

Edit Pattern Examples, 4-48
Edit pattern validation, 4-42

F

Field validation, 4-42
file definitions used by the Toolkit, B-1

M

M4TKTBLU
 data types, 4-29
 parameters, 4-28
M5TKBIT, 4-1
M5TKBYTE, 4-1
M5TKCOMM, 4-3
M5TKCOPY
 return codes, 4-6
M5TKDATEG, 2-1
M5TKDATEJ, 2-1
M5TKDATG, 4-8
 example, 4-10
M5TKDATG and M5TKDATJ
 return codes, 4-9
M5TKDATJ, 4-8
 example, 4-10
M5TKHEXA, 4-11
 return codes, 4-12
M5TKHEXA Routine
 control block, 4-12
M5TKLCAL, 4-20

M5TKLJST, 4-13

M5TKLJST and M5TKRJST
 return codes, 4-14

M5TKLJST and M5TKRJST Routine
 control block, 4-14

M5TKLJST Example, 4-15

M5TKPREV, 4-16

M5TKPRNT, 4-20
 return codes, 4-21

M5TKPRNT Routine
 control block, 4-20

M5TKRJST, 4-13

M5TKRJST Example, 4-15

M5TKTAB, 4-22
 how to define tables, 4-25
 return codes, 4-24

M5TKTAB Example, 4-25

M5TKTAB Routine
 control block, 4-22

M5TKTAB Tables
 assemble and link edit, 4-34

M5TKTBLU, 4-25, 4-31
 generated column information block, 4-33
 generated table header, 4-32
 parameters, 4-27, 4-28

M5TKTBLU macros, 4-25

M5TKTCT, 4-35
 return codes, 4-37

M5TKTCT Routine
 control block, 4-36

M5TKTS, 4-37
 return codes, 4-39

M5TKTS Routine
 control block, 4-38

M5TKUSRA, 4-40

M5TKVLID, 2-1
 defining validation symbols, A-1
 validation symbols, 4-43

P

PVCOUNT, 4-17

PVFCTNAM, 4-17

PVGENLEN, 4-17

PVKEY, 4-17

PVKEYLEN, 4-17

R

read a file backwards, 4-16

S

Search strategy, 4-31

T

Table Definition Example, 4-30

Table lookup, 4-31

table lookup, 4-25, 4-34

Table Structure, 4-31

TCTUA, 4-40

TKDATES macro, 2-1

Toolkit
 file definitions, B-1
 overview, 1-1

Toolkit subroutines, 4-16, 4-37
 M5TKDATEG, 2-1
 M5TKDATEJ, 2-1
 M5TKVLID, 2-1, A-1

V

VALIDTAB macro, A-2