4777 Magnetic Stripe Unit and
4778 PIN-Pad Magnetic Stripe Reader

IBM

OS/2 Programming Guide

SA34-2205-00

4777 Magnetic Stripe Unit and
4778 PIN-Pad Magnetic Stripe Reader

IBM

OS/2 Programming Guide

SA34-2205-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page ix.

First Edition (September 1994)

Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult your IBM representative to be sure you have the latest edition and any Technical Newsletters.

IBM does not stock publications at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader’s comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department 78C, 1001 W. T. Harris Boulevard West, Charlotte, NC 28262-8563, U.S.A. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Notices ix
License ix
Trademarks ix

About This Book xi
Who Should Read This Book xi
How This Book Is Organized xi
Related Publications xii

Chapter 1. Introducing the 4777 Magnetic Stripe Unit and the 4778 PIN-Pad Magnetic Stripe Reader 1-1
The 4777 Magnetic Stripe Unit 1-1
The 4778 PIN-Pad Magnetic Stripe Reader 1-2

Chapter 2. Loading and Initializing the Device Driver 2-1
Prerequisites 2-1
Automatically Loading the Device Driver 2-1
Manually Loading the Device Driver 2-2
Loading the Dynamic Link Libraries 2-3
Loading the 4777 Dynamic Link Library 2-3
Loading the 4778 Dynamic Link Library 2-3
Loading the Message Files 2-4

Chapter 3. Understanding the 4777 Application Program Interface 3-1
Using the 4777 Device-Support Code 3-2
Opening the 4777 (MagOpen) 3-2
Closing the 4777 (MagClose) 3-3
Loading the Device Parameters (MagLoadDevParms) 3-4
Setting the Multitrack-Read Operation Mode (MagSetOperationMode) . . . 3-7
Encoding Magnetic Data (MagEncodeData) 3-8
Resetting the 4777 (MagResetDevice) 3-9
Stopping an I/O Operation (MagAbort) 3-9
Reading the Device Parameters (MagReadDevParms) 3-10
Reading the Error Statistics (MagReadErrorStats) 3-11
Reading the Configuration Status (MagReadConfigStatus) 3-12
Reading Magnetic Data (MagReadData) 3-13

Chapter 4. Understanding the 4778 Application Program Interface 4-1
Opening the Device 4-3
PinOpen 4-3
PinMagOpen 4-4
Closing the Device 4-5
PinClose 4-5
PinMagClose 4-6
Stopping an I/O Operation 4-7
PinAbort 4-7
PinMagAbort 4-8
Reading the Configuration Status 4-9
PinReadConfigStatus 4-9
PinMagReadConfigStatus 4-11

Magnetic-Stripe Functions 4-13
Loading the Device Parameters (PinMagLoadDevParms) 4-13
Setting the Multitrack-Read Operation Mode (PinMagSetOperationMode) 4-16
Resetting the 4778 (PinMagResetDevice) 4-17
Reading the Device Parameters (PinMagReadDevParms) 4-18
Reading the Error Statistics (PinMagReadErrorStats) 4-19
Reading Magnetic Data (PinMagReadData) 4-20
4778 PIN Keypad Function Calls 4-22
Setting the Nonencrypted Mode (PinSetModeClr) 4-22
Setting the Encrypted Mode (PinSetModeEnc) 4-23
Entering the Master Key Manually (PinEnterMasterKey) 4-24
Loading the Master Encryption Key (PinLoadMasterKey) 4-26
Loading the Encryption Key (PinLoadKey) 4-28
Loading the Initial-Chaining Value (PinLoadICV) 4-29
Loading the PIN Verification Parameters (PinLoadVerifParms) 4-30
Creating the 4704 PIN Block (PinReadPin4704) 4-31
Creating the 3624 PIN Block (PinReadPin3624) 4-32
Creating the ANSI X9.8 PIN Block (PinReadPinAnsi98) 4-34
Verifying the PIN Block (PinVerifyPin) 4-36
Creating the Offset Data (PinCreateOffsetData) 4-38
Generating the Message Authentication Code (PinGenerateMac) 4-40
Verifying the Message Authentication Code (PinVerifyMac) 4-43
Running the Device Diagnostic Test (PinExecDevDiag) 4-45
Reading the Device Serial Number (PinReadSN) 4-47
Reading the Nonencrypted Data (PinReadClearData) 4-48
Writing to the Display (PinWriteDisplay) 4-49

Chapter 5. Data Formats 5-1
Magnetic-Stripe Data Format 5-1
Read Format 5-1
Reading Data 5-3
PIN Data Formats 5-4
Nonencrypted PIN Data Format 5-4
Encrypted PIN Data Format 5-5
4704 EPP Format 5-5
ANSI X9.8 Format 5-6
3624 PIN Format 5-7
Verifying the 3624 PINs 5-7
Managing the Cryptographic Keys 5-9
Loading Keys 5-9
Using Key Variants 5-10
Converting a Master Key to the Keypad-Entry Format 5-12
Using Message Authentication Codes 5-14

Chapter 6. 4777 Multiple-Virtual-DOS-Machine I/O System 6-1
4777 MVDM I/O System 6-1
Loading the 4777 MVDM Device Driver 6-2
Automatic Installation 6-2
Manual Installation 6-2
Using the 4777 MVDM Device Driver 6-3
DOS Application Program Interface 6-4
OS/2 Protected-Mode Operation 6-4
Avoiding Unwanted Initialization Error Messages 6-5

Chapter 7. 4778 Multiple-Virtual-DOS-Machine I/O System 7-1
4778 MVDM I/O System 7-1
Loading the 4778 MVDM Device Driver 7-2
Automatic Installation 7-2
Manual Installation 7-2
Using the 4778 MVDM Device Driver 7-3
DOS Application Program Interface 7-4
OS/2 Protected Mode 7-4
Avoiding Unwanted Initialization Error Messages 7-5

Chapter 8. Messages and Status Codes 8-1
Installation Messages 8-1
Application Program Status Codes 8-5
PIN Keypad Status Codes 8-5
Magnetic-Stripe Operation Status Codes 8-7

Index X-1

Figures

1-1. 4777 Magnetic Stripe Unit 1-1
1-2. 4778 PIN-Pad Magnetic Stripe Reader 1-3
3-1. Default Values for the MagLoadDevParms Track Parameters 3-6
4-1. Default Values for the PinMagLoadDevParms Track Parameters . . 4-15
5-1. Single-Track Read-Data Format 5-1
5-2. Double-Track Read-Data Format 5-1
5-3. Translations for Nonencrypted PIN Keypad Data 5-4
5-4. 4704 EPP PIN Format 5-5
5-5. ANSI X9.8 PIN Format 5-6
5-6. 3624 PIN Format 5-7
5-7. Verifying the 3624 PINs 5-8
5-8. Key Variants 5-11
5-9. Variant Descriptor Bytes for the 4778 Commands 5-11
5-10. Example of a Master Key Conversion 5-12
5-11. Hexadecimal-to-Keystroke Conversion Table 5-13
5-12. Using Message Authentication Codes 5-15
6-1. 4777 MVDM I/O System 6-1
7-1. 4778 MVDM I/O System 7-1

Notices

License

Trademarks

IBM	Personal Computer AT
Operating System/2	Personal System/2
OS/2	PS/2
PC/XT	PS/ValuePoint
Personal Computer XT	TopView

About This Book

This book tells you how to control the IBM[*] 4777 Magnetic Stripe Unit and the IBM[*] 4778 PIN-Pad Magnetic Stripe Reader in an Operating System/2[*] (OS/2[*]) environment. It explains how to customize, load, and use the device driver. This guide also helps you write application programs that access the device driver.

Who Should Read This Book

The information in this book is intended for people who write, or maintain, system and application programs that work with the 4777 and the 4778.

How This Book Is Organized

This book contains the following sections:

Chapter 1, "Introducing the 4777 Magnetic Stripe Unit and the 4778 PIN-Pad Magnetic Stripe Reader," describes the 4777 Magnetic Stripe Unit and 4778 PIN-Pad Magnetic Stripe Reader devices.

Chapter 2, "Loading and Initializing the Device Driver," tells you how to load and initialize the device driver in an OS/2 environment.

Chapter 3, "Understanding the 4777 Application Program Interface," lists function calls that are available when you use the 4777 device driver.

Chapter 4, "Understanding the 4778 Application Program Interface," lists function calls that are available when you use the 4778 device driver.

Chapter 5, "Data Formats," describes the data stream formats used by the 4778 PIN-Pad Magnetic Stripe Reader, including the use of cryptographic keys, personal identification number (PIN) formats, and message authentication codes.

Chapter 6, "4777 Multiple-Virtual-DOS-Machine I/O System," tells you how to load and use the 4777 multiple-virtual-DOS-machine (MVDM) I/O system.

Chapter 7, "4778 Multiple-Virtual-DOS-Machine I/O System," tells you how to load and use the 4778 multiple-virtual-DOS-machine (MVDM) I/O system.

Chapter 8, "Messages and Status Codes," lists the installation messages for the 4777 and 4778 devices and provides an explanation of each message. It also includes a list of the application program status codes and an explanation for each code.

[*] Trademark of IBM

Related Publications

You might need additional information from one or more of the following publications:

- *OS/2 Programming Guide*, SA34-2194
- *4777 Magnetic Stripe Unit and 4778 PIN-Pad Magnetic Stripe Reader DOS Programming Guide*, SA34-2206
- *4700 Finance Communication System: System Summary*, GC31-2016
- *4700 Finance Communication System, Controller Programming Library; Volume 5: Cryptographic Programming*, GC31-2070
- *4700 Financial I/O Planning Guide*, GC31-3762
- *4777 Magnetic Stripe Unit Installation Guide*

Chapter 1. Introducing the 4777 Magnetic Stripe Unit and the 4778 PIN-Pad Magnetic Stripe Reader

This chapter describes the IBM 4777 Magnetic Stripe Unit and the IBM 4778 PIN-Pad Magnetic Stripe Reader.

The 4777 and 4778 are compatible with the IBM 4700 Finance Communication System family of programs and products. For more information about the 4700 Finance Communication System, see the *4700 Finance Communication System: System Summary*.

The 4777 Magnetic Stripe Unit

The 4777 Magnetic Stripe Unit, available in four models, is a countertop magnetic-stripe unit (see Figure 1-1). The 4777 reads and encodes magnetic-stripe documents that are manually passed through the device.

The 4777 is based on the IBM 4717 Magnetic Stripe Unit. You can attach the 4777 to the serial port or the auxiliary port of an IBM Personal System/2* (PS/2*) workstation or a PS/ValuePoint* workstation. You can also attach both the 4777 and the 4778 PIN-Pad Magnetic Stripe Reader to your workstation with a special connector. For more information about installing both devices on the same workstation, see the *4777 Magnetic Stripe Unit Installation Guide*.

Note: For serial port attachment the 4777 cannot be installed on the same system with a 4717 device. Although they are similar in operation, the device drivers for the 4717 devices are not compatible with the 4777 device.

Warning: The 4777 does not differentiate one medium from another. If you try to encode a previously encoded medium, the previously data can be destroyed. Because of this and other operations that require encoding expertise, only authorized banking personnel should use the 4777 to encode data.

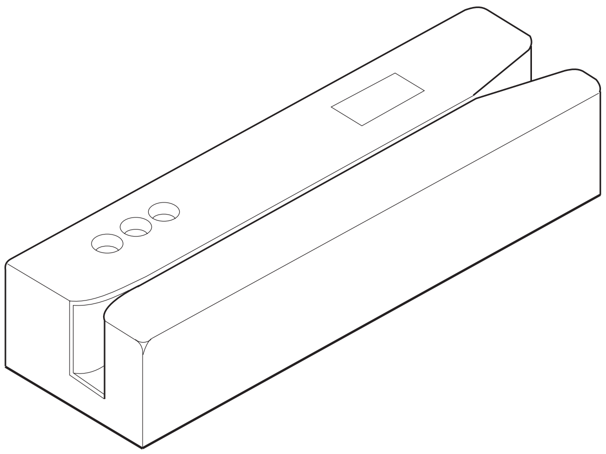

Figure 1-1. 4777 Magnetic Stripe Unit

* Trademark of IBM

Model 001	Reads, on a single pass, tracks 1 and 2 on credit cards and identification cards. This model reads track 1 and track 2 at 75 or 210 bits per inch (bpi), in accordance with American National Standards Institute (ANSI) standards X4.16-1983 and the International Standards Organization (ISO) standards 7810 and 7811/2-5.
Model 002	Reads tracks 2 and 3 on credit cards and identification cards. It also reads and encodes passbooks. This model reads tracks 2 and 3 of credit cards and identification cards at 75 or 210 bpi. It encodes passbooks using the 4700 specifications or in accordance with the ISO standard 8484. It reads passbooks that are encoded by the IBM 3604 Keyboard Display, by the IBM 4704 Keyboard Display, or in accordance with the ISO standard 8484.
Model 003	Reads and encodes tracks 1 and 2 on credit cards and identification cards in accordance with the ISO and the ANSI specifications. This model reads tracks 1 and 2 at 75 or 210 bpi, and encodes track 1 at 210 bpi and track 2 at 75 bpi. The 4777 Model 003 is useful in an administrative work area of a financial institution that creates the identification cards for personal banking machines when a customer opens an account.
Model 004	Reads tracks 2 and 3 on credit cards and identification cards and reads passbooks. This model reads tracks 2 and 3 at 75 or 210 bpi. It reads passbooks that are encoded by the IBM 3604 Keyboard Display, by the IBM 4704 Keyboard Display, or in accordance with the ISO Standard 8484. **Note:** This model is available only in Europe, the Middle East, and Asia.

The 4778 PIN-Pad Magnetic Stripe Reader

The 4778 PIN-Pad Magnetic Stripe Reader, available in three models, is a countertop keypad with or without a magnetic-stripe reader (MSR) (see Figure 1-2 on page 1-3). The three models of the 4778 are described below:

Model 001	Reads tracks 1 and 2 on credit and ID cards on a single pass. Supports application programs requiring a 12-key PIN pad.
Model 002	Supports application programs requiring a 12-key PIN pad.
Model 003	Reads tracks 1, 2, and 3 on credit and ID cards. Supports application programs requiring a 12-key PIN pad.

The keypad is used to enter a personal identification number (PIN) for validating financial transactions. The 4778 keypad accepts and encrypts PINs with enhanced security. The keypad has 10 numeric keys, 2 special keys, and a 16-character display. The magnetic-stripe reader lets your applications read data from magnetic stripes on credit cards or ID cards.

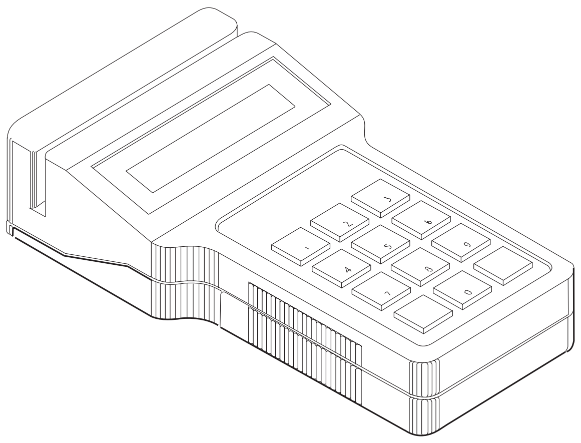

Figure 1-2. 4778 PIN-Pad Magnetic Stripe Reader

You can attach both the 4778 PIN-Pad Magnetic Stripe Reader and the 4777 Magnetic Stripe Unit to your workstation with a special connector. For more information about installing both devices on the same workstation, see the *4778 PIN-Pad Magnetic Stripe Reader Installation Guide*. The magnetic-stripe reader in the 4778 provides the same function as the IBM 4777 Model 001 Magnetic Stripe Unit.

The 4778 PIN keypad is based on the IBM 4718 PIN Keypad. You can attach the 4778 to the serial port or to the auxiliary port of an IBM Personal System/2 (PS/2) workstation or a PS/ValuePoint workstation.

Note: For serial port attachment the 4778 cannot be installed on the same system with a 4718 device. Although they are similar in operation, the device drivers for the devices are not compatible.

Operations available with the 4778 PIN-Pad Magnetic Stripe Reader:

- Reads data encoded in formats of either the American National Standards Institute (ANSI) X4.16-1983 or the International Organization for Standardization (ISO) 7810 and 7811/2-5.
- Operates in clear or encrypted mode
- Uses the Data Encryption Standard (DES) algorithm
- Uses a master key that can be a single-length key or a double-length key
- Encrypts PINs using the American National Standards Institute (ANSI) X9.8, IBM 3624 Keyboard Display, or IBM 4704 Keyboard Display formats within the keypad
- Lets you download master keys or enter them using the keypad
- Generates or verifies a message authentication code (MAC)
- Verifies a PIN or creates PIN offset data using the IBM 3624 Consumer Transaction Facility algorithm

Chapter 2. Loading and Initializing the Device Driver

This chapter describes how to load and initialize the 4777 and 4778 device drivers and how to configure your operating environment. It includes the following sections:

- Automatically loading and initializing the device driver
- Manually loading and initializing the device driver
- Loading the dynamic link libraries
- Loading the message files

For more information about using the 4777 and 4778 in a multiple-virtual-DOS-machine (MVDM) environment, see Chapter 6, “4777 Multiple-Virtual-DOS-Machine I/O System” and Chapter 7, “4778 Multiple-Virtual-DOS-Machine I/O System.”

Installation Notes:

- If you installed the OS/2 device driver for a 4777, do not install the device driver for a 4778. One device driver supports both the 4777 and 4778; install the device driver only once for either the 4777 or the 4778.
- If you are using a virtual-machine-boot (VMB) Disk Operating System (DOS) session, you must add the device-driver statements to the CONFIG.SYS file that is used to start the DOS session.

Prerequisites

The 4777, 4778, and device-driver software can be used on the following systems:

- Personal System/2
- PS/ValuePoint
- OS/2 Version 1.3 (or higher) operating system

  If you use the multiple-virtual-DOS-machine (MVDM) I/O system, you need the OS/2 Version 2.0 (or higher) operating system.

Automatically Loading the Device Driver

To load and initialize the device driver automatically, do the following:

1. Insert the OS/2 device-drivers diskette in the A: drive.
2. Type **A: FINSTALL** at the system prompt.

Select the configuration you want on the series of panels that the installation program displays. Existing device support specified in the CONFIG.SYS file for the 4777 and 4778 is remarked out in the CONFIG.SYS file. The program updates the files that contain the device-support code and copies them to the target drive that you specify.

Manually Loading the Device Driver

To manually load and initialize the 4777 and 4778 device driver, copy the FIOSERDD.SYS or FIOAUXDD.SYS file from the OS/2 device-drivers diskette to your target drive. Include the following DEVICE statement in your CONFIG.SYS file:

```
DEVICE = [d:[path]]namedd.SYS
[/Cx/M/P/I/W/S]
```

Note: The brackets [ ] indicate optional parameters.

The device statement parameters are:

d:path
The identifier for the disk drive or the diskette drive (d:) and the directory-search sequence to locate the FIOSERDD.SYS or FIOAUXDD.SYS file.

namedd.SYS
The name of the device-driver data set, either FIOSERDD.SYS for serial port attach or FIOAUXDD.SYS for auxiliary port attach. Copy this file to the directory specified in the path parameter.

[/C*x*/M/P/I/W/S]
The optional parameters define the following:

A serial port other than the default value (COM1)
Devices attached to the auxiliary or serial port
Use of delay operations when an error occurs
Use of a 4778 for magnetic-stripe functions

At least one blank must precede the options list. You can specify the optional parameters in any combination and sequence: `/Cx/M/P/I/W/S`.

/C*x* (Serial only) Indicates the COM (serial) port to which the devices are attached, where *x* can be 1, 2, 3, or 4. If you do not specify this option, the default value is to use the COM1 port.

/M Indicates that a 4777 is attached. The device driver tests for a 4777. If the device driver does not find a 4777, it displays an error message. If it finds a 4777 and it passes the self-test, the application program can access the device driver.

If the device driver finds a 4777 and you do not specify the /M parameter, the device driver displays a specification error and the device driver is installed.

Parameters M and S are mutually exclusive.

/P Indicates that a 4718 or 4778 model 2 is attached. The device driver tests for a PIN pad. If the device driver does not find a PIN pad, it displays an error message. If it finds a 4778 or 4718 and the device passes the self-test, the application program can access the device driver.

If the device driver finds a 4718 or 4778 PIN pad and you do not specify the /P parameter, the device driver displays a specification error and the device driver is installed.

Parameters P and I are mutually exclusive.

/I	This parameter is only applicable to the FIOAUXDD.SYS driver used for attaching a 4778 model 1 or model 3 to the mouse port (auxiliary port). The device driver tests for a 4778. If the device driver does not find a 4778, it displays an error message. If it finds a 4778 or a 4718 and the device passes the self-test, the application program can access the device driver. Parameters P and I are mutually exclusive. If the device driver finds a 4778 and you do not specify the /I parameter, the device driver displays a specification error and the device driver is installed.
/S	This option lets you attach a 4778 device and use it as though it was a 4777 Model 001 device. This lets you change a system configured with a 4717 Model 001 and a 4718 keypad so that it can use a single 4778 device. This parameter is ignored if either a 4717 or 4777 is attached or a 4778 MSR is not attached. In either case, an error message is displayed. Parameters S and M are mutually exclusive.
/W	Using this option causes the device-driver operation to pause after displaying installation messages. The device driver waits indefinitely; you must press **Enter** to continue. The device driver displays installation messages when it detects an error in the operational environment (such as an error in the device or an error in the options that are specified). **Note:** If the device driver detects a *critical error* during initialization, the device driver might not load into storage or it might not establish communication with the application program.

Loading the Dynamic Link Libraries

Loading the 4777 Dynamic Link Library

To load the 4777 dynamic link library (DLL), copy the MAGCALLS.DLL dynamic-link-library file to the root directory or to a user-defined library directory. Identify this library directory to the operating system by modifying the LIBPATH statement in the CONFIG.SYS file as follows:

```
SET LIBPATH = d: PATH ;
```

The parameters for the SET LIBPATH statement are:

d:	The identifier for the disk drive or the diskette drive
PATH	The directory-search sequence for the disk or the diskette that contains the MAGCALLS.DLL file

Loading the 4778 Dynamic Link Library

To load the 4778 dynamic link libraries, copy the PINCALLS.DLL and PINMSR.DLL files to the root directory or to a user-defined library directory. Identify this library directory to the operating system by modifying the LIBPATH statement in the CONFIG.SYS file as follows:

```
SET LIBPATH = d: PATH  ;
```

The descriptions of the parameters for the SET LIBPATH statement are as follows:

d:	This parameter specifies the identifier for the disk drive or the diskette drive.
PATH	This parameter specifies the directory-search sequence for the disk or the diskette that contains the PINCALLS.DLL and PINMSR.DLL files.

Loading the Message Files

The OS/2 device-support code uses the FIO.MSG and FIOH.MSG message files to generate the installation messages. If you loaded the message files when you installed another device, do not load them again. If you did not load the message files, copy the FIO.MSG and FIOH.MSG message files to the root directory or to a user-defined library directory. Identify this library directory to the device-support code by modifying the DPATH statement in the CONFIG.SYS file as follows:

```
SET DPATH = d: PATH ;
```

The descriptions of the SET DPATH statement parameters are as follows:

d:	This parameter specifies the identifier for the disk drive or the diskette drive.
PATH	This parameter specifies the directory-search sequence for the disk or the diskette that contains the FIO.MSG and FIOH.MSG message files.

Chapter 3. Understanding the 4777 Application Program Interface

This chapter describes the application program interface (API) for the 4777 device-support code. The device-support code supplies the API to the OS/2 operating system and supports the attachment of all four models of the 4777 Magnetic Stripe Unit.

Your application program can use the function calls of the 4777 device-support code to pass data to and from the 4777. The 4777 device-support code enables your application program to do the following:

- Select which tracks the 4777 reads or encodes
- Specify the track parameters for a read or an encode operation
- Read or encode the magnetic-stripe data
- Check the validity of the data that is read or encoded

The 4777 OS/2 device-support code includes the following components and data files:

File Name	**Description**
FIOSERDD.SYS	This file contains the device driver that supports serial attachment of both the 4777 and the 4778.
FIOAUXDD.SYS	This file contains the device driver that supports mouse port (auxiliary) attachment of both the 4777 and the 4778.
MAGCALLS.DLL	This file contains the 4777 device-support code that the dynamic link library (DLL) function calls implement for the API.
FIO.MSG	This file contains the object file for the error messages.
FIOH.MSG	This file contains the object file for the help messages.

Using the 4777 Device-Support Code

This section describes the function calls for the 4777 device-support code. The dynamic-link-library name for the 4777 I/O DLL file is MAGCALLS.DLL. This section describes the following function calls:

Function Call	Used For
MagOpen	Opening the 4777, see page 3-2
MagClose	Closing the 4777, see page 3-3
MagLoadDevParms	Loading the device parameters, see page 3-4
MagSetOperationMode	Setting the multitrack-read operation mode, see page 3-7
MagEncodeData	Encoding magnetic data, see page 3-8
MagResetDevice	Resetting the 4777, see page 3-9
MagAbort	Stopping an I/O operation, see page 3-9
MagReadDevParms	Reading the device parameters, see page 3-10
MagReadErrorStats	Reading the error statistics, see page 3-11
MagReadConfigStatus	Reading the configuration status, see page 3-12
MagReadData	Reading magnetic data, see page 3-13

Programming Note: This section describes the 4777 function calls in mixed case for readability, but they are only recognized as uppercase character strings. When you use a compiler that generates mixed-case external names, code the function calls in uppercase letters. For example, the function call **MagOpen** is actually the external name **MAGOPEN**.

Opening the 4777 (MagOpen)

The MagOpen function call opens the 4777 for the current session. When the application program issues subsequent function calls, the function calls use the 4777 device-driver handle that they receive from this MagOpen function call.

MagOpen (MagHandle, *rc*)

Parameters

The parameters for the MagOpen call are:

MagHandle
This parameter is a long pointer to a word value where the 4777 device-support code returns a value that represents the 4777 device-driver handle to the application program. Your application program uses the 4777 device-driver handle as the controlling identifier for all subsequent function calls to the 4777 device-support code.

rc This parameter is a word value that represents the return code from the MagOpen function call. The valid return codes are:

0	No error.
110	The 4777 device driver failed to open.
1551	The 4777 device-support code ended the operation.

Remarks

To request services from the 4777 device-support code, the application program issues the MagOpen function call to generate a 4777 device-driver handle. The handle is a 16-bit binary value that the 4777 device-support code returns to the application program after the MagOpen function call completes.

Following a successful MagOpen function call, the 4777 device-support code parameters and operating modes are set to their default values. For more information about the default values, see “Loading the Device Parameters (MagLoadDevParms)” on page 3-4 and “Setting the Multitrack-Read Operation Mode (MagSetOperationMode)” on page 3-7. To establish the correct operational environment, the application program issues additional function calls.

This function call opens the 4777 device driver for the following functions:

- Read and write operations
- Deny-all-sharing mode
- Errors not reported to the system critical-error handler

When the application program issues subsequent function calls, the function calls use the MagHandle value that they receive from this MagOpen function call. If your application program issues a function call with a different MagHandle value, the function call fails.

Only one application program at a time can use the 4777 device-support code. A second application program cannot use the 4777 until the controlling application program issues a MagClose function call.

Closing the 4777 (MagClose)

The MagClose function call closes the 4777 for the current session.

MagClose (MagHandle, *rc*)

Parameters

The parameters for the MagClose call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

rc This parameter is a word value that represents the return code from the MagClose function call. The valid return codes are:

0	No error.
1540	The 4777 MagHandle parameter is not valid.

Remarks

When the 4777 device-support code is closed, it is available to other application programs. When the application program issues a MagClose function call, the 4777 device-support code closes the parameters for the 4777 device-support code, the 4777 is disabled, and all the indicators are switched off.

Loading the Device Parameters (MagLoadDevParms)

The MagLoadDevParms function call loads the operating parameters of the 4777.

MagLoadDevParms (MagHandle, TrackParms, *rc*)

Parameters

The parameters for the MagLoadDevParms call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

TrackParms
This parameter is a long pointer to a 24-byte buffer that contains the track parameters. Bytes 1 through 8 contain the track-1 parameters, bytes 9 through 16 contain the track-2 parameters, and bytes 17 through 24 contain the track-3 parameters.

Byte 1 - Data and longitudinal redundancy check (LRC) parity definition
This parameter defines the magnetic-character parity definition for both the data characters and the LRC character as follows:

Value	Meaning
00H	Odd data parity and odd LRC parity
01H	Even data parity and even LRC parity
02H	Odd data parity and even LRC parity
03H	Even data parity and odd LRC parity

During the MagReadData function call, this parameter verifies that the parity definition is correct for each magnetic character read from the stripe. During the MagEncodeData function call, this parameter adds the parity definition for each magnetic character that is received. This parameter also assigns parity to the LRC value that is generated for the encode data stream.

Byte 2 - Character definition
This parameter indicates the size of the magnetic character in bits. The fields for this parameter are as follows:

Value	Meaning
05H	5 bits per character
06H	6 bits per character
07H	7 bits per character

The size value includes the magnetic character and its parity bit. For example, a character definition of 5 bits per character correlates to a 4-bit character with one parity bit to form a complete magnetic character.

The read-data or encode-data characters that pass between the application program and the 4777 device-support code are always in an 8-bit byte and do not include the parity bit. The 4777 device-support code adds and checks the parity of the characters.

Byte 3 - Primary start-of-message (PSOM) character

When the application program issues a MagReadData function call, this parameter specifies the character that indicates the beginning of a magnetic record. When the application program issues a MagEncodeData function call, this parameter specifies the start-of-message (SOM) character that it adds to the encode data stream.

Byte 4 - Alternate start-of-message (ASOM) character

When the application program issues a MagReadData function call, this parameter specifies the alternate character that indicates the beginning of a magnetic record. When the 4777 device-support code looks for the beginning of a magnetic record, it accepts either the PSOM or the ASOM character.

The MagEncodeData function call does not use this parameter.

Byte 5 - Primary end-of-message (PEOM) character

When the application program issues a MagReadData function call, this parameter specifies the character that indicates the end of a magnetic record. When the application program issues a MagEncodeData function call, this parameter specifies the EOM character that the 4777 device-support code adds to the encode data stream.

Byte 6 - Alternate end-of-message (AEOM) character

When the application program issues a MagReadData function call, this parameter specifies the alternate character that indicates the end of a magnetic record. When the 4777 device-support code looks for the end of a magnetic record, it accepts either the PEOM or the AEOM character.

The MagEncodeData function call does not use this parameter.

Byte 7 - Encode format control definition

When the application program issues a MagEncodeData function call, this parameter defines whether the 4777 device-support code encodes on the magnetic medium one copy or two copies of the encode data stream. If this parameter requests two copies of the data stream, it also specifies the number of interrecord zero (IRZ) characters that the 4777 device-support code encodes onto the medium between the two records.

Encoding two copies of the record usually results in a more reliable reading of the magnetic medium. When requesting the double-encode option, ensure that the medium is long enough to contain the two records, including the leading zeros and the interrecord zeros.

Note: If bit 7 is equal to zero, byte 7 indicates a single record. If bit 7 is equal to one, byte 7 indicates a double record. Bits 0 through 6 indicate the number of IRZ characters between the two records.

The MagReadData function call does not use this parameter.

Byte 8 - Number of leading zero characters

This parameter specifies the number of leading zero characters that the 4777 device-support code encodes on the medium before it encodes the first record.

The MagReadData function call does not use this parameter.

For more information about the default values of the track-1 parameters, see Figure 3-1.

Bytes 9 through 16
These 8 bytes contain the track-2 parameters. Their definitions are identical to the track-1 parameters except that they apply to track 2. For more information about the default values of the track-2 parameters, see Figure 3-1.

Bytes 17 through 24
These 8 bytes contain the track-3 parameters. Their definitions are identical to the track-1 parameters except that they apply to track 3. For more information about the default values of the track-3 parameters, see Figure 3-1.

rc This parameter is a word value that represents the return code from the MagLoadDevParms function call. The valid return codes are:

0	No error.
1540	The 4777 MagHandle parameter is not valid.
1541	The 4777 is not available.
1545	The track-1 parameter is not valid.
1546	The track-2 parameter is not valid.
1547	The track-3 parameter is not valid.

Figure 3-1. Default Values for the MagLoadDevParms Track Parameters

Track	Byte	Description	Default
Track 1	Byte 1	Data/LRC parity definition	02H
	Byte 2	Character definition	07H
	Byte 3	Primary SOM	05H
	Byte 4	Alternate SOM	05H
	Byte 5	Primary EOM	1FH
	Byte 6	Alternate EOM	1FH
	Byte 7	Format control	00H (single record)
	Byte 8	Leading zero characters	0AH
Track 2	Byte 9	Data/LRC parity definition	02H
	Byte 10	Character definition	05H
	Byte 11	Primary SOM	0BH
	Byte 12	Alternate SOM	0DH
	Byte 13	Primary EOM	0FH
	Byte 14	Alternate EOM	0CH
	Byte 15	Format control	85H (Model 2 only) 00H (Model 3 only)
	Byte 16	Leading zero characters	3CH (Model 2 only) 05H (Model 3 only)
Track 3	Byte 17	Data/LRC parity definition	02H
	Byte 18	Character definition	05H
	Byte 19	Primary SOM	0BH
	Byte 20	Alternate SOM	0DH
	Byte 21	Primary EOM	0FH
	Byte 22	Alternate EOM	0CH
	Byte 23	Format control	85H
	Byte 24	Leading zero characters	3CH

Remarks

When the application program issues a MagOpen function call, the parameters are set to the default values defined in Figure 3-1 on page 3-6. The application program then issues a MagReadDevParms function call (see "Reading the Device Parameters (MagReadDevParms)" on page 3-10) to determine the current operational characteristics. The application program issues this function call to change any of the parameters to match the requirements of the application program.

If a parameter is not valid, the MagReadDevParms function call fails. The function call performs the following checks:

- The data and LRC parity is 00, 01, 02, or 03.
- The character definition is 05, 06, or 07.
- The SOM value does not equal 00.
- The alternate SOM value does not equal 00.
- The EOM value does not equal 00.
- The alternate EOM value does not equal 00.
- The IRZ value of the encode control does not equal 00 if it is a double record.
- The encode leading zeros do not equal 00.

Setting the Multitrack-Read Operation Mode (MagSetOperationMode)

The MagSetOperationMode function call sets the multitrack-read operation mode of the 4777.

MagSetOperationMode (MagHandle, OperationMode, *rc*)

Parameters

The parameters for the MagSetOperationMode call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

OperationMode
This parameter is a word value that defines how the 4777 device-support code performs a multitrack-read operation. This parameter specifies the fields as follows:

Bit	**Description**
15–1	These bits are reserved and must be set to zero.
0	If this bit is set to zero, the data returns only if all the requested track data is valid. If this bit is set to one, the data returns if the data for at least one of the requested tracks is valid.

rc This parameter is a word value that represents the return code from the MagSetOperationMode function call. The valid return codes are:

0	No error.
1540	The 4777 MagHandle parameter is not valid.
1541	The 4777 is not available.

Remarks

This function call sets the operational mode for multitrack-read operations. If all the data that is read from the requested tracks is valid, one mode returns the data to the application program. If the data from at least one of the requested tracks is valid, the other mode returns data to the application program (this is the default).

For more information about the read-data format, see Chapter 5, “Data Formats.”

Encoding Magnetic Data (MagEncodeData)

The MagEncodeData function call encodes data on the magnetic media.

MagEncodeData (MagHandle, DataLength, Data, *rc*)

Parameters

The parameters for the MagEncodeData call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

DataLength
This parameter is a word value that specifies the length of the encode data stream that is input in the data buffer. The minimum data length is 2 bytes, which includes the length byte and the TRK byte for encoding a track with only an SOM value and an EOM value and no data bytes.

Data
This parameter is a long pointer to the data buffer that contains the data that will be encoded.

rc This parameter is a word value that represents the return code from the MagEncodeData function call. The valid return codes are:

0	No error.
1538	The 4777 is not attached.
1539	A hardware error occurred in the 4777.
1540	The 4777 MagHandle parameter is not valid.
1541	The 4777 is not available.
1543	The 4777 does not support encoding for the requested track.
1544	An error occurred in the encode data stream.
1548	An error occurred in the buffer size.
1550	The application program ended the operation.
1551	The 4777 device-support code ended the operation.

Remarks

This function call encodes the data onto the track that is specified in the TRK identification field of the encode data stream. For more information about the encode data stream, see Chapter 5, “Data Formats.”

Until the operator performs a successful operation or the application program issues a MagAbort function call to end this MagEncodeData function call, the

4777 device-support code blocks the thread for this function call. For more information about how the application program can end this operation, see "Stopping an I/O Operation (MagAbort)" on page 3-9.

Resetting the 4777 (MagResetDevice)

The MagResetDevice function call resets the 4777.

```
MagResetDevice (MagHandle, rc)
```

Parameters

The parameters for the MagResetDevice call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

rc This parameter is a word value that represents the return code from the MagResetDevice function call. The valid return codes are:

0	No error.
1538	The 4777 is not attached.
1540	The 4777 MagHandle parameter is not valid.
1541	The 4777 is not available.
1549	Errors occurred in the 4777 self-test.
1551	The 4777 device-support code ended the operation.

Remarks

The device self-tests run as a result of this function call. The return code indicates the completion status of the device self-tests.

Stopping an I/O Operation (MagAbort)

The MagAbort function call cancels a previous MagReadData or MagEncodeData function call that left the 4777 in the active state.

```
MagAbort (MagHandle, rc)
```

Parameters

The parameters for the MagAbort call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

rc This parameter is a word value that represents the return code from the MagAbort function call. The valid return codes are:

0	No error.
1540	The 4777 MagHandle parameter is not valid.
1551	The 4777 device-support code ended the operation.

Remarks

This function call disables the 4777 and turns off the device indicators. Before your application program issues a MagReadData or a MagEncodeData function call, it must start another thread to support this MagAbort function call. Until the operator passes the medium through the 4777, the thread for the MagReadData or the MagEncodeData function call is blocked. You can cancel this blocked thread with the MagAbort function call.

Reading the Device Parameters (MagReadDevParms)

The MagReadDevParms function call reads the operational parameters of the 4777.

MagReadDevParms (MagHandle, StatusBuffer, TransferCount, *rc*)

Parameters

The parameters for the MagReadDevParms call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

StatusBuffer
This parameter is a long pointer to a 24-byte buffer where the device parameter status returns. The device parameter status is 8 bytes for the track-1, track-2, and track-3 parameters.

TransferCount
This parameter is a long pointer to a word value that indicates the size of the StatusBuffer when the application program issues the function call. When this parameter returns, it is modified to indicate the actual number of bytes that were transferred to the StatusBuffer. The maximum transfer count is 24 bytes.

rc This parameter is a word value that represents the return code from the MagReadDevParms function call. The valid return codes are:

0	No error.
1540	The 4777 MagHandle parameter is not valid.
1548	The data buffer size is too small for the requested data.

Remarks

This function call returns the device parameter information to the application program. If the MagReadDevParms function call requests the parameters for all three tracks, the application program must ensure that the buffer is at least 24 bytes in length. For more information about the parameters that this function call returns, see “Loading the Device Parameters (MagLoadDevParms)” on page 3-4.

Reading the Error Statistics (MagReadErrorStats)

The MagReadErrorStats function call reads the error statistics for the 4777.

MagReadErrorStats (MagHandle, StatBuffer, TransferCount, *rc*)

Parameters

The parameters for the MagReadErrorStats call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

StatBuffer
This parameter is a long pointer to a 6-byte buffer where the device error statistics are returned. The format for the error statistics is as follows:

Error Statistic	**Number of Bytes**
Track-1 read error count	1 byte
Track-2 read error count	1 byte
Track-3 read error count	1 byte
Track-1 encode error count	1 byte
Track-2 encode error count	1 byte
Track-3 encode error count	1 byte

TransferCount
This parameter is a long pointer to a word value that indicates the size of the StatBuffer when the application program issues the function call. When this parameter returns, it is modified to indicate the actual number of bytes that were transferred to the StatBuffer. The maximum transfer count is 6 bytes.

rc This parameter is a word value that represents the return code from the MagReadErrorStats function call. The valid return codes are:

0	No error.
1540	The 4777 MagHandle parameter is not valid.
1548	The data buffer size is too small for the requested data.

Remarks

This function call reads the 6-byte error statistics for the 4777. Each error count can have a maximum value of decimal 255. To receive all the error statistics, the StatBuffer parameter must be at least 6 bytes in length.

The 4777 device-support code maintains the error statistics while it is active (that is, during the time period between the MagOpen and MagClose function calls). When the application program issues a MagOpen function call, the error statistics are set to zero. If you want to maintain the error statistics over a longer period of time, the application program must read the values before it issues a MagClose function call and must save the values within the application program data area.

Reading the Configuration Status (MagReadConfigStatus)

The MagReadConfigStatus function call reads the device read and encode capabilities of the attached 4777, and it reads the results of the power-on test for the 4777 device driver.

MagReadConfigStatus (MagHandle, ConfigStatusBuff, *rc*)

Parameters

The parameters for the MagReadConfigStatus call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

ConfigStatusBuff
This parameter is a long pointer to the 16-bit word buffer where the configuration status returns. The format for the configuration status is as follows:

Bit	Description
15	If this bit is set, there is a specification error in an optional parameter in the CONFIG.SYS file.
14	If this bit is set, a 4777 is not attached.
13	If this bit is set, there are power-on errors in the 4777.
12	If this bit is set, there is a load error or an installation error for the 4777 device driver.
11–7	Reserved; do not use.
6	This bit indicates whether the function call can read track 1.

Value	Meaning
0	The function call cannot read track 1.
1	The function call can read track 1.

Bit	Description
5	This bit indicates whether the function call can read track 2.

Value	Meaning
0	The function call cannot read track 2.
1	The function call can read track 2.

Bit	Description
4	This bit indicates whether the function call can read track 3.

Value	Meaning
0	The function call cannot read track 3.
1	The function call can read track 3.

Bit	Description
3	Reserved; do not use.
2	This bit indicates whether the function call can encode track 1.

Value	Meaning
0	The function call cannot encode track 1.
1	The function call can encode track 1.

1	This bit indicates whether the function call can encode track 2.

Value	Meaning
0	The function call cannot encode track 2.
1	The function call can encode track 2.

0	This bit indicates whether the function call can encode track 3.

Value	Meaning
0	The function call cannot encode track 3.
1	The function call can encode track 3.

rc This parameter is a word value that represents the return code from the MagReadConfigStatus function call. The valid return codes are:

0	No error.
1540	The 4777 MagHandle parameter is not valid.

Remarks

The application program uses this function call to determine the capabilities of the 4777 and to detect errors when the device is powered on.

Reading Magnetic Data (MagReadData)

The MagReadData function call reads the magnetic-stripe data from the magnetic media.

MagReadData (MagHandle, ReadTracks, DataBuffer, TransferCount, *rc*)

Parameters

The parameters for the MagReadData call are:

MagHandle
This parameter is the word value 4777 device-driver handle that was obtained from the MagOpen function call.

ReadTracks
This parameter specifies a 16-bit word value that identifies the tracks that the function call will read. The format for the ReadTracks parameter is as follows:

Bit	Description
15	Setting this bit returns control to the application program after a magnetic stripe read error; otherwise the device is reset for another read try.
14–3	Reserved. These bits must be set to zero.
2	The function call reads track 1.
1	The function call reads track 2.
0	The function call reads track 3.

DataBuffer
This parameter is a long pointer to the data buffer where the magnetic read data returns.

TransferCount
This parameter is a long pointer to a word value that indicates the size of the DataBuffer when the application program issues the function call. When this

parameter returns, it is modified to indicate the actual number of bytes that were transferred to the DataBuffer. If an error occurs and the read data is not processed, this parameter returns with a value of zero.

rc This parameter is a word value that represents the return code from the MagReadData function call. The valid return codes are:

0	No error.
1538	The 4777 is not attached.
1539	An error occurred in the 4777 hardware.
1540	The 4777 MagHandle parameter is not valid.
1541	The 4777 is not available.
1542	The 4777 does not support the track that is requested for the read operation.
1548	The data buffer size is too small for the requested data.
1550	The application program ended the operation.
1551	The 4777 device-support code ended the operation.
1552	The 4777 experienced a magnetic stripe read error.

Remarks

This function call reads the magnetic-stripe data from the magnetic media. Until the operator performs a successful operation or the application program issues a MagAbort function call to end this MagReadData function call, the thread for this function call is blocked. For more information about how the application program can end this function call, see “Stopping an I/O Operation (MagAbort)” on page 3-9.

The ReadTracks parameter must reflect the actual read capabilities of the 4777 or an error returns. For more information about how the read capabilities are determined, see “Reading the Configuration Status (MagReadConfigStatus)” on page 3-12.

If the ReadTracks parameter indicates a multitrack-read operation, the current state of the multitrack-read operation mode determines how the read data returns to the application program. For more information about how the read data returns to the application program, see “Setting the Multitrack-Read Operation Mode (MagSetOperationMode)” on page 3-7.

Chapter 4. Understanding the 4778 Application Program Interface

This chapter describes the application program interface (API) for the 4778 device-support code. The 4778 device-support code supplies the application program interface to the OS/2 system. This interface is supported only in the OS/2 environment.

Your application program can use the function calls of the 4778 device-support code to pass data to and from the 4778. The 4778 device-support code enables your application program to do the following:

- Read magnetic-stripe data
- Write information to the keypad display
- Read nonencrypted data
- Read encrypted data
- Read and set device information
- Send control data to the 4778
- Receive data from the 4778

Note: The 4778 MSR and PIN keypad devices cannot be enabled concurrently.

Programming Note: This chapter describes the 4778 function calls in mixed case for readability, but they are only recognized as uppercase character strings. For example, the function call **PinOpen** is actually the external name **PINOPEN**. When you use a compiler that generates mixed-case external names, code the function calls in uppercase letters.

The 4778 PIN keypad API (PINCALLS.DLL) provides the following function calls:

Function Call	Used For
PinOpen	Opening the 4778 PIN keypad. See page 4-3.
PinClose	Closing the 4778 PIN keypad. See page 4-5.
PinAbort	Stopping an I/O operation. See page 4-7.
PinReadConfigStatus	Reading the configuration status. See page 4-9.
PinSetModeClr	Setting the nonencrypted mode. See page 4-22.
PinSetModeEnc	Setting the encrypted mode. See page 4-23.
PinEnterMasterKey	Entering the master key manually. See page 4-24.
PinLoadMasterKey	Loading the master encryption key. See page 4-26.
PinLoadKey	Loading the encryption key. See page 4-28.
PinLoadICV	Loading the initial-chaining value. See page 4-29.
PinLoadVerifParms	Loading the PIN verification parameters. See page 4-30.
PinReadPin4704	Creating the 4704 PIN block. See page 4-31.
PinReadPin3624	Creating the 3624 PIN block. See page 4-32.
PinReadPinAnsi98	Creating the ANSI X9.8 PIN block. See page 4-34.
PinVerifyPin	Verifying the PIN block. See page 4-36.
PinCreateOffsetData	Creating the offset data. See page 4-38.
PinGenerateMac	Generating the message authentication code. See page 4-40.

Function Call	Used For
PinVerifyMac	Verifying the message authentication code. See page 4-43.
PinExecDevDiag	Running the device diagnostic test. See page 4-45.
PinReadSN	Reading the device serial number. See page 4-47.
PinReadClearData	Reading the nonencrypted data. See page 4-48.
PinWriteDisplay	Writing data to the PIN keypad display. See page 4-49.

The name for the 4778 magnetic-stripe reader API dynamic-link-library (DLL) file is PINMSR. DLL. This API provides the following function calls:

Function Call	Used For
PinMagOpen	Opening the 4778 magnetic-stripe reader. See page 4-4.
PinMagClose	Closing the 4778 magnetic-stripe reader. See page 4-6.
PinMagLoadDevParms	Loading the magnetic-stripe device parameters. See page 4-13.
PinMagSetOperationMode	Setting the multitrack-read operation mode. See page 4-16.
PinMagResetDevice	Resetting the 4778 magnetic-stripe reader. See page 4-17.
PinMagAbort	Stopping an I/O operation. See page 4-8.
PinMagReadDevParms	Reading the magnetic-stripe device parameters. See page 4-18.
PinMagReadErrorStats	Reading the magnetic-stripe error statistics. See page 4-19.
PinMagReadConfigStatus	Reading the magnetic-stripe-reader configuration status. See page 4-11.
PinMagReadData	Reading magnetic data. See page 4-20.

To request services from the 4778 device-support code, the application program must issue the PinOpen or PinMagOpen function call. This function call opens the device driver and, when the function call completes successfully, it identifies a device-driver handle. The parameters for the 4778 and the operating modes are set to their default values. To establish the correct operational environment, the application program issues additional function calls.

Note: The 4778 and the 4777 share the same set of device-support codes.

The 4778 device-support code includes the following components and data files:

File Name	Description
FIOSERDD.SYS	This file contains the device driver that supports serial attachment of both the 4777 and the 4778.
FIOAUXDD.SYS	This file contains the device driver that supports auxiliary attachment of both the 4777 and the 4778.
PINCALLS.DLL	This file contains the 4778 device-support code that the DLL function calls implement for the PIN keypad API.
PINMSR.DLL	This file contains the 4778 device-support code that the DLL function calls implement for the magnetic-stripe-reader API.
FIO.MSG	This file contains the object file for the error messages.
FIOH.MSG	This file contains the object file for the help messages.

Opening the Device

These function calls open the 4778 device. If you are using the 4778 magnetic-stripe reader, use the PinMagOpen call to access the magnetic-stripe functions.

PinOpen

The PinOpen function call opens the 4778 PIN keypad for the current session. When the application program issues subsequent function calls, the function calls use the device-driver handle that they receive from this PinOpen function call.

PinOpen (PinHandle, *rc*)

Parameters

The PinOpen function call uses the following parameters:

PinHandle
This parameter is a long pointer to a word value where the 4778 device-support code returns a value that represents the 4778 device-driver handle to the application program. Your application program uses the 4778 device-driver handle as the controlling identifier for all subsequent function calls to the 4778 device-support code.

rc This parameter is a word value that represents the return code from the PinOpen function call. The valid return codes are:

0	No error.
110	The 4778 device driver failed to open.
1798	An error occurred in the PINCALLS.DLL file.

Remarks

To request services from the 4778 PIN keypad device-support code, the application program issues the PinOpen function call to generate a 4778 device-driver handle.

This function call opens the 4778 device driver for the following functions:

- Read and write operations
- Deny-all-sharing mode
- Errors are not reported to the system critical-error handler

When the application program issues subsequent function calls, the function calls use the PinHandle value that they receive from this PinOpen function call. If your application program issues a function call with a different PinHandle value, the function call fails.

Only one application program at a time can use the 4778 PIN-Pad Magnetic Stripe Reader device-support code. A second application program cannot use the 4778 until the controlling application program issues a PinClose function call.

PinMagOpen

The Open function call opens the 4778 magnetic-stripe reader (MSR) for the current session. When the application program issues subsequent function calls, the function calls use the 4778 device-driver handle that they receive from this PinMagOpen function call.

```
PinMagOpen (MagHandle, rc)
```

Parameters

The PinMagOpen function call uses the following parameters:

MagHandle
This parameter is a long pointer to a word value where the 4778 device-support code returns a value that represents the 4778 device-driver handle to the application program. Your application program uses the 4778 device-driver handle as the controlling identifier for all subsequent function calls to the 4778 device-support code.

rc This parameter is a word value that represents the return code from the PinMagOpen function. The valid return codes are:

0	No error.
110	The 4778 device driver failed to open.
1551	The 4778 device-support code ended the operation.

Remarks

To request services from the 4778 MSR device-support code, the application program issues the PinMagOpen function call to generate a 4778 device-driver handle. The handle is a 16-bit binary value that the 4778 device-support code returns to the application program after the PinMagOpen function call completes.

The 4778 device-support code parameters and operating modes are set to their default values. For more information about the default values, see “Loading the Device Parameters (MagLoadDevParms)” on page 3-4 and “Setting the Multitrack-Read Operation Mode (MagSetOperationMode)” on page 3-7. To establish the correct operational environment, the application program issues additional function calls.

This function call opens the 4778 device driver for the following functions:

Read operations
Deny-all-sharing mode
Errors not reported to the system critical-error handler

When the application program issues subsequent function calls, the function calls use the MagHandle value that they receive from this PinMagOpen function call. If your application program issues a function call with a different MagHandle value, the function call fails.

Only one application program can use the 4778 MSR device-support code at a time. A second application program cannot use the 4778 until the controlling application program issues a PinMagClose function call.

Closing the Device

These function calls close the device after an operation is complete. If you are using the 4778 magnetic-stripe functions, use the PinMagClose call to end the operation.

PinClose

The PinClose function call closes the 4778 PIN keypad for the current session.

PinClose (PinHandle, *rc*)

Parameters

The PinClose function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

rc This parameter is a word value that represents the return code from the PinClose function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1798	An error occurred in the PINCALLS.DLL file.
1801	The 4778 device driver is not open.

Remarks

When the 4778 PIN keypad device-support code is closed, it is available to other application programs. When the application program issues a PinClose function call, the 4778 device-support code closes the parameters for the 4778 PIN keypad device-support code, disables the 4778 PIN keypad, and switches off all the indicators.

PinMagClose

The PinMagClose function call closes the 4778 MSR for the current session.

PinMagClose (MagHandle, *rc*)

Parameters

The PinMagClose function call uses the following parameters:

MagHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinMagOpen function call.

rc This parameter is a word value that represents the return code from the PinMagClose function call. The valid return codes are:

0	No error.
1540	The 4778 MagHandle parameter is not valid.

Remarks

When the 4778 MSR device-support code is closed, it is available to other application programs. When the application program issues a PinMagClose function call, the 4778 MSR device-support code closes the parameters for the 4778 MSR device-support code, the 4778 MSR is disabled, and all the indicators are switched off.

Stopping an I/O Operation

These function calls stop any I/O operations that are processing or are waiting to process. For the 4778 magnetic-stripe functions, use the PinMagAbort call to stop I/O processing.

PinAbort

The PinAbort function call ends a PIN keypad I/O operation that is currently processing or a pending I/O operation.

PinAbort (PinHandle, *rc*)

Parameters

The PinAbort function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

rc This parameter is a word value that represents the return code from the PinAbort function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1801	The 4778 device driver is not open.

Remarks

This function call disables the 4778 PIN keypad and turns off the device indicators. Before your application program issues a PIN function call, it must start another thread to support the PinAbort function call. Until the operator completes the PIN keypad operation, the thread for the PIN function call is blocked. You can cancel the blocked thread with the PinAbort function call.

PinMagAbort

The PinMagAbort function call ends a previous PinMagReadData function call that left the 4778 MSR in the active state.

```
PinMagAbort (MagHandle, rc)
```

Parameters

The PinMagAbort function call uses the following parameters:

MagHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinMagOpen function call.

rc This parameter is a word value that represents the return code from the PinMagAbort function call. The valid return codes are:

0	No error.
1540	The 4778 MagHandle parameter is not valid.
1551	The 4778 device-support code ended the operation.

Remarks

This function call disables the 4778 MSR and turns off the device indicators. Before your application program issues a PinMagReadData, it must start another thread to support this PinMagAbort function call. Until the operator passes the medium through the 4778 MSR, the thread for the PinMagReadData function call is blocked. You can cancel this blocked thread with the PinMagAbort function call.

Reading the Configuration Status

The PinReadConfigStatus and the PinMagReadConfigStatus function calls let your application determine the configuration status of the 4778.

PinReadConfigStatus

The PinReadConfigStatus function call reads the configuration status of the 4778 PIN keypad and the configuration status of the device driver for the application program.

```
PinReadConfigStatus (PinHandle, StatusBuffer, rc)
```

Parameters

The PinReadConfigStatus function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

StatusBuffer
This parameter is a long pointer to a word buffer where the configuration status returns. The format for the configuration status is as follows:

Bit	Description
15	This bit indicates whether the optional parameters in the CONFIG.SYS file are correct.

Value	Meaning
0	The specification of the optional parameters in the CONFIG.SYS file is correct.
1	There is an error in the specification of an optional parameter in the CONFIG.SYS file.

Bit	Description
14	This bit indicates whether the 4778 is attached.

Value	Meaning
0	A 4778 device is attached.
1	A 4778 device is not attached.

Bit	Description
13	This bit indicates whether the 4778 PIN keypad passed the device self-test.

Value	Meaning
0	The 4778 PIN keypad device passed the device self-test.
1	The 4778 PIN keypad device failed the device self-test.

Bit	Description
12	This bit indicates whether the 4778 device driver was initialized.

Value	Meaning
0	The 4778 device driver initialized successfully.
1	The 4778 device driver failed to initialize.

Bit	Description
11–1	These bits are reserved.

0 This bit indicates whether the 4778 device driver is in the nonencrypted mode.

Value	**Meaning**
0	The 4778 device driver is in the nonencrypted mode.
1	The 4778 device driver is in the encrypted mode.

Note: If any of the bits 15 through 11 are on, bit 0 is unpredictable.

rc This parameter is a word value that represents the return code from the PinReadConfigStatus function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1801	The 4778 device driver is not open.
1802	The 4778 is not available.

Remarks

The application program uses this function call to determine the capabilities of the 4778 PIN keypad and to detect errors when the device is powered on.

PinMagReadConfigStatus

The PinMagReadConfigStatus function call reads the status of the read capabilities of the attached 4778 magnetic-stripe reader and reads the results of the power-on test for the 4778 device driver.

PinMagReadConfigStatus (MagHandle, ConfigStatusBuff, *rc*)

Parameters

The PinMagReadConfigStatus function call uses the following parameters:

MagHandle
This word value parameter is the 4778 device-driver handle obtained from the PinMagOpen function call.

ConfigStatusBuff
This parameter is a long pointer to the 16-bit word buffer where the configuration status returns. The format for the configuration status is as follows:

Bit **Description**

15 This bit indicates whether there is a specification error in an optional parameter in the CONFIG.SYS file.

Value	Meaning
0	The specification of the optional parameters in the CONFIG.SYS file is correct.
1	There is an error in the specification of an optional parameter in the CONFIG.SYS file.

14 This bit indicates whether the 4778 is attached.

Value	Meaning
0	A 4778 device is attached.
1	A 4778 device is not attached.

13 This bit indicates whether there are power-on errors in the 4778 MSR.

Value	Meaning
0	There are no power-on errors in the 4778 MSR.
1	There are power-on errors in the 4778 MSR.

12 This bit indicates whether there is a load error or an installation error for the 4778 device driver.

Value	Meaning
0	There are no load errors or installation errors in the 4778 device driver.
1	There is a load error or an installation error in the 4778 device driver.

11–7 These bits are reserved; do not use these bits.

6	This bit indicates whether the function call can read track 1.	
	Value	**Meaning**
	0	The function call cannot read track 1.
	1	The function call can read track 1.
5	This bit indicates whether the function call can read track 2.	
	Value	**Meaning**
	0	The function call cannot read track 2.
	1	The function call can read track 2.
4–0	These bits are reserved; do not use these bits.	

rc This parameter is a word value that represents the return code from the PinMagReadConfigStatus function call. The valid return codes are:

0	No error.
1540	The 4778 MagHandle parameter is not valid.

Remarks

The application program uses this function call to determine the capabilities of the 4778 MSR and to detect errors when the device is powered on.

Magnetic-Stripe Functions

The following function calls are valid for magnetic-stripe functions of the 4778.

Loading the Device Parameters (PinMagLoadDevParms)

The PinMagLoadDevParms function call loads the operating parameters of the 4778 MSR.

PinMagLoadDevParms (MagHandle, TrackParms, *rc*)

Parameters

The PinMagLoadDevParms function call uses the following parameters:

MagHandle

This word value parameter is the 4778 device-driver handle that was obtained from the PinMagOpen function call.

TrackParms

This parameter is a long pointer to a 24-byte buffer that contains the track parameters. Bytes 1 through 8 contain the track-1 parameters.
Bytes 9 through 16 contain the track-2 parameters. The individual track parameters are defined as follows:

Byte 1 - Data and longitudinal redundancy check (LRC) parity definition

This parameter defines the magnetic-character parity definition for both the data characters and the LRC character as follows:

Value	Meaning
00H	Odd data parity and odd LRC parity
01H	Even data parity and even LRC parity
02H	Odd data parity and even LRC parity
03H	Even data parity and odd LRC parity

During the PinMagReadData function call, this parameter verifies that the parity definition is correct for each magnetic character that is read from the stripe.

Byte 2 - Character definition

This parameter indicates the size of the magnetic character in bits. The fields for this parameter are as follows:

Value	Meaning
05H	5 bits per character
06H	6 bits per character
07H	7 bits per character

The size value includes the magnetic character and its parity bit. For example, a character definition of 5 bits per character correlates to a 4-bit character with one parity bit to form a complete magnetic character.

The read-data characters that pass between the application program and the 4778 MSR device-support code are always in an 8-bit byte and do not include the parity bit. The 4778 device-support code adds and checks the parity of the characters.

Byte 3 - Primary start-of-message (PSOM) character
When the application program issues a PinMagReadData function call, this parameter specifies the character that indicates the beginning of a magnetic record.

Byte 4 - Alternate start-of-message (ASOM) character
When the application program issues a PinMagReadData function call, this parameter specifies the alternate character that indicates the beginning of a magnetic record. When the 4778 MSR device-support code looks for the beginning of a magnetic record, it accepts either the PSOM or the ASOM character.

Byte 5 - Primary end-of-message (PEOM) character
When the application program issues a PinMagReadData function call, this parameter specifies the character that indicates the end of a magnetic record.

Byte 6 - Alternate end-of-message (AEOM) character
When the application program issues a PinMagReadData function call, this parameter specifies the alternate character that indicates the end of a magnetic record. When the 4778 MSR device-support code looks for the end of a magnetic record, it accepts either the PEOM or the AEOM character.

Byte 7
The PinMagReadData function call does not use this parameter.

Byte 8
The PinMagReadData function call does not use this parameter.

For more information about the default values of the track-1 parameters, see Figure 4-1 on page 4-15.

Bytes 9 through 16
These 8 bytes contain the track-2 parameters. Their definitions are identical to the track-1 parameters except that they apply to track 2. For more information about the default values of the track-2 parameters, see Figure 4-1 on page 4-15.

rc This parameter is a word value that represents the return code from the PinMagLoadDevParms function call. The valid return codes are:

0	No error.
1540	The 4778 MagHandle parameter is not valid.
1541	The 4778 is not available.
1545	The track-1 parameter is not valid.
1546	The track-2 parameter is not valid.

Figure 4-1. Default Values for the PinMagLoadDevParms Track Parameters

Track	Byte	Description	Default
Track 1	Byte 1	Data/LRC parity definition	02H
	Byte 2	Character definition	07H
	Byte 3	Primary SOM	05H
	Byte 4	Alternate SOM	05H
	Byte 5	Primary EOM	1FH
	Byte 6	Alternate EOM	1FH
	Byte 7	Not used	
	Byte 8	Not used	
Track 2	Byte 9	Data/LRC parity definition	02H
	Byte 10	Character definition	05H
	Byte 11	Primary SOM	0BH
	Byte 12	Alternate SOM	0DH
	Byte 13	Primary EOM	0FH
	Byte 14	Alternate EOM	0CH
	Byte 15	Not used	
	Byte 16	Not used	

Remarks

When the application program issues a PinMagOpen function call, the parameters are set to the default values defined in Figure 4-1. The application program then issues a PinMagReadDevParms function call (see "Reading the Device Parameters (MagReadDevParms)" on page 3-10) to determine the current operational characteristics. The application program issues this function call to change any of the parameters to match the requirements of the application program.

If a parameter is not valid, the PinMagReadDevParms function call fails. The function call performs the following checks:

- The data and LRC parity is 00, 01, 02, or 03.
- The character definition is 05, 06, or 07.
- The SOM value does not equal 00.
- The alternate SOM value does not equal 00.
- The EOM value does not equal 00.
- The alternate EOM value does not equal 00.

Setting the Multitrack-Read Operation Mode (PinMagSetOperationMode)

The PinMagSetOperationMode function call sets the multitrack read-operation mode of the 4778 MSR.

PinMagSetOperationMode (MagHandle, OperationMode, *rc*)

Parameters

The PinMagSetOperationMode function call uses the following parameters:

MagHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinMagOpen function call.

OperationMode
This word value parameter specifies a field that defines how the 4778 MSR device-support code performs a multitrack-read operation. This parameter specifies the fields as follows:

Bit	Description
15–1	These bits are reserved and must be set to zero.
0	If this bit is set to zero, the data returns only if all the requested track data is valid. If this bit is set to one, the data returns if the data for at least one of the requested tracks is valid.

rc This parameter is a word value that represents the return code from the PinMagSetOperationMode function call. The valid return codes are:

0	No error.
1540	The 4778 MagHandle parameter is not valid.
1541	The 4778 is not available.

Remarks

This function call sets the operational mode for multitrack-read operations. If all the data that is read from the requested tracks is valid, one mode returns the data to the application program. If the data from at least one of the requested tracks is valid, the other mode returns data to the application program (this is the default).

For more information about the read-data format, see Chapter 5, “Data Formats.”

Resetting the 4778 (PinMagResetDevice)

The PinMagResetDevice function call resets the 4778 MSR.

```
PinMagResetDevice (MagHandle, rc)
```

Parameters

The PinMagResetDevice function call uses the following parameters:

MagHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinMagOpen function call.

rc This word value parameter represents the return code from the PinMagResetDevice function call. The valid return codes are:

0	No error.
1538	The 4778 is not attached.
1540	The 4778 MagHandle parameter is not valid.
1541	The 4778 MSR is not available.
1549	Errors occurred in the 4778 MSR self-test.
1551	The 4778 MSR device-support code ended the operation.

Remarks

The device self-tests run as a result of this function call. The return code indicates the completion status of the device self-tests.

Reading the Device Parameters (PinMagReadDevParms)

The PinMagReadDevParms function call reads the operational parameters of the 4778 MSR.

PinMagReadDevParms (MagHandle, StatusBuffer, TransferCount, *rc*)

Parameters

The PinMagReadDevParms function call uses the following parameters:

MagHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinMagOpen function call.

StatusBuffer
This parameter is a long pointer to a 16-byte buffer where the device parameter status returns. The device parameter status is 8 bytes for the track-1 and track-2 parameters.

TransferCount
This parameter is a long pointer to a word value that indicates the size of the StatusBuffer when the application program issues the function call. When this parameter returns, it is modified to indicate the actual number of bytes that were transferred to the StatusBuffer. The maximum transfer count is 16 bytes.

rc This parameter is a word value that represents the return code from the PinMagReadDevParms function call. The valid return codes are:

0	No error.
1540	The 4778 MagHandle parameter is not valid.
1548	Data buffer size is too small for the amount of data requested.

Remarks

This function call returns the 4778 MSR device parameter information to the application program. If the PinMagReadDevParms function call requests the parameters for two tracks, the application program must ensure that the buffer is at least 16 bytes in length. For more information about the parameters that this function call returns, see “Loading the Device Parameters (MagLoadDevParms)” on page 3-4.

Reading the Error Statistics (PinMagReadErrorStats)

The PinMagReadErrorStats function call reads the error statistics for the 4778 MSR.

PinMagReadErrorStats (MagHandle, StatBuffer, TransferCount, *rc*)

Parameters

The PinMagReadErrorStats function call uses the following parameters:

MagHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinMagOpen function call.

StatBuffer
This parameter is a long pointer to a 2-byte buffer where the device error statistics are returned. The format for the error statistics is as follows:

Error Statistic	**Number of Bytes**
Track-1 read error count	1 byte
Track-2 read error count	1 byte

TransferCount
This parameter is a long pointer to a word value that indicates the size of the StatBuffer when the application program issues the function call. When this parameter returns, it is modified to indicate the actual number of bytes that were transferred to the StatBuffer. The maximum transfer count is 2 bytes.

rc This parameter is a word value that represents the return code from the PinMagReadErrorStats function call. The valid return codes are:

0	No error.
1540	The 4778 MagHandle parameter is not valid.
1548	Data buffer size is too small for the amount of data requested.

Remarks

This function call reads the 2-byte error statistics for the 4778 MSR. Each error count can have a maximum value of decimal 255. To receive all of the error statistics, the StatBuffer parameter must be at least 2 bytes in length.

The 4778 MSR device-support code maintains the error statistics while it is active (that is, during the time period between the PinMagOpen and PinMagClose function calls). When the application program issues a PinMagOpen function call, the error statistics are set to zero. If you want to maintain the error statistics over a longer period of time, the application program must read the values before it issues a PinMagClose function call and must save the values within the application program data area.

Reading Magnetic Data (PinMagReadData)

The PinMagReadData function call reads the magnetic-stripe data from the magnetic media.

```
PinMagReadData (MagHandle, ReadTracks, DataBuffer, TransferCount, rc)
```

Parameters

The PinMagReadData function call uses the following parameters:

MagHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinMagOpen function call.

ReadTracks
This parameter specifies a 16-bit word value that identifies the tracks that the function call reads. The ReadTracks parameter specifies the fields as follows:

Bit	Description
15–3	Reserved. These bits must be set to zero.
2	The function call reads track 1.
1	The function call reads track 2.
0	Not used.

DataBuffer
This parameter is a long pointer to the data buffer where the magnetic read data returns.

TransferCount
This parameter is a long pointer to a word value that indicates the size of the DataBuffer parameter when the application program issues the function call. When this parameter returns, it is modified to indicate the actual number of bytes transferred to the DataBuffer parameter. If an error occurs, the magnetic read data is not processed and the parameter returns with a value of zero.

rc This parameter is a word value that represents the return code from the PinMagReadData function call. The valid return codes are:

0	No error.
1538	The 4778 is not attached.
1539	An error occurred in the 4778 MSR hardware.
1540	The 4778 MagHandle parameter is not valid.
1541	The 4778 MSR is not available.
1542	The 4778 MSR does not support the track that is requested for the read operation.
1548	The data buffer size is too small for the amount of data requested.
1550	The application program ended the operation.
1551	The 4778 MSR device-support code ended the operation.

Remarks

This function call reads the magnetic-stripe data from the magnetic media. Until the operator performs a successful operation or the application program issues a PinMagAbort function call to end this PinMagReadData function call, the thread for this function call is blocked. For more information about how the application program can end this function call, see “PinMagAbort” on page 4-8.

The ReadTracks parameter must reflect the actual read capabilities of the 4778 or an error returns. For more information about how the read capabilities are determined, see “Reading the Configuration Status (MagReadConfigStatus)” on page 3-12.

If the ReadTracks parameter indicates a multitrack-read operation, the current state of the multitrack-read operation mode determines how the read data returns to the application program. For more information about how the read data returns to the application program, see “Setting the Multitrack-Read Operation Mode (MagSetOperationMode)” on page 3-7.

4778 PIN Keypad Function Calls

The following function calls are used for 4778 PIN keypad functions.

Setting the Nonencrypted Mode (PinSetModeClr)

The PinSetModeClr function call sets the 4778 PIN keypad device-support code in the nonencrypted mode.

PinSetModeClr (PinHandle, *rc*)

Parameters

The PinSetModeClr function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

rc This parameter is a word value that represents the return code from the PinSetModeClr function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

The PinSetModeClr function call sets the 4778 PIN keypad device-support code in nonencrypted mode. When the 4778 PIN keypad device-support code is in the nonencrypted mode, it denies requests to create PIN blocks and returns an error code to the application program. When the 4778 PIN keypad device-support code is in the encrypted mode, it denies requests for nonencrypted data and returns an error code to the application program. If valid encryption keys are loaded in the 4778, all other cryptographic operations work in both the nonencrypted and the encrypted mode of the 4778.

Setting the Encrypted Mode (PinSetModeEnc)

The PinSetModeEnc function call sets the 4778 PIN keypad device-support code in the encrypted mode.

Warning: This function call invalidates all previously-loaded encryption keys. When this function call sets the device-support code in the encrypted mode, you must reload all the Data Encryption Standard (DES) encryption keys.

```
PinSetModeEnc (PinHandle, rc)
```

Parameters

The PinSetModeEnc function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

rc This parameter is the word value that represents the return code from the PinSetModeEnc function call. The valid return calls are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

When the 4778 PIN keypad device-support code is in the encrypted mode, it denies requests for nonencrypted data and returns an error code to the application program. When the 4778 device-support code is in the nonencrypted mode, it denies requests to create PIN blocks and returns an error code to the application program. If valid encryption keys are loaded in the 4778, all other cryptographic operations work in both the nonencrypted and the encrypted mode of the 4778.

Entering the Master Key Manually (PinEnterMasterKey)

The PinEnterMasterKey function call sets the 4778 PIN keypad for the manual entry of the master key.

PinEnterMasterKey (PinHandle, KeyType, SNBuffer, *rc*)

Parameters

The PinEnterMasterKey function call uses the following parameters:

PinHandle

This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

KeyType

This word value parameter specifies the type of master key to enter manually. The fields for this parameter are as follows:

Bit	Description
15–5	These bits are reserved and must be set to zero.
4	This bit indicates the size of the master key.
3–1	These bits are reserved and must be set to zero.
0	This bit indicates whether the master key is a single-entry or a dual-entry key.

Value	Meaning
0	This value indicates that the master key is a 16-byte key.
1	This value indicates that the master key is an 8-byte key.

Value	Meaning
0	This value indicates that the master key is a single-entry key.
1	This value indicates that the master key is a dual-entry key.

SNBuffer

This parameter is a long pointer to an 8-byte buffer where the triple-encrypted device serial number is returned. The device serial number is encrypted using the entered master key.

Note: The keypad keystrokes used to manually enter the master key are in 3-3-2 format. The valid keystrokes are `mmn` where:

`m` has a range of 0 through 7.
`n` has a range of 0 through 2.

For more information about manually entering the master key through the PIN keypad, see “Converting a Master Key to the Keypad-Entry Format” on page 5-12.

rc This word value parameter represents the return code from the PinEnterMasterKey function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1796	A key was pressed that is not valid.
1798	An error occurred in the PINCALLS.DLL file.
1799	The application program ended the operation.
1800	A parameter error occurred.
1801	The 4778 PIN keypad device driver is not open.
1802	The 4778 is not available.
1804	The key parity is not valid for the requested function call.
1805	An error occurred for the PinOpen function call.

Remarks

The PinEnterMasterKey function call reads the keystrokes from the PIN keypad, converts the keystrokes to hexadecimal, and stores the results as the master key in the 4778. If you enter an 8-byte master key, this function call duplicates the key as the second 8 bytes of the double-length master key.

Loading the Master Encryption Key (PinLoadMasterKey)

The PinLoadMasterKey function call uses the PS/2 communication interface to load a master encryption key into the 4778 PIN keypad.

PinLoadMasterKey (PinHandle, KeyType, KeyBuffer, SNBuffer, *rc*)

Parameters

The PinLoadMasterKey function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

KeyType
This word value parameter specifies the type of master key that the device driver loads into the 4778 PIN keypad. The fields for this parameter are as follows:

Bit	Description
15–5	These bits are reserved and must be set to zero.
4	This bit indicates the size of the master key.
	Value / **Meaning**
	0 — This value indicates that the master key is a 16-byte key.
	1 — This value indicates that the master key is an 8-byte key.
3–1	These bits are reserved and must be set to zero.
0	This bit indicates whether the master-key buffer contains a nonencrypted key.
	Value / **Meaning**
	0 — This value indicates that the master-key buffer contains a nonencrypted key.
	1 — This value indicates that the master-key buffer contains an encrypted key.

KeyBuffer
This parameter is a long pointer to the buffer that contains the master key.

SNBuffer
This parameter is a long pointer to an 8-byte buffer where the triple-encrypted device serial number is returned. The device serial number is encrypted under the loaded key.

rc This parameter is a word value that represents the return code from the PinLoadMasterKey function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1804	The key parity is not valid for the requested function call.
1805	An error occurred for the PinOpen function call.

Remarks

The master-key buffer contains either an 8-byte or a 16-byte encryption key — the KeyType parameter specifies the value. If you load the master key in encrypted form, set the bit in the KeyType parameter to indicate that the 4778 PIN keypad device-support code must decrypt the key before it places the key in storage. (If the key is encrypted, the key is always encrypted under the resident master key in the 4778.) If you load an 8-byte key, this function call duplicates the master key as the second 8 bytes of the double-length master key.

Loading the Encryption Key (PinLoadKey)

The PinLoadKey function call uses the PS/2 communication interface to load an 8-byte encryption key into the 4778 PIN keypad.

```
PinLoadKey (PinHandle, KeyIdentifier, VariantDescriptor, KeyBuffer,
            SNBuffer, rc)
```

Parameters

The PinLoadKey function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

KeyIdentifier
This parameter is a word value that specifies the destination index of the key in the range of 00H to FFH.

VariantDescriptor
This parameter is a word value that specifies the master-key variant used to decrypt the key. The fields for this parameter are as follows:

Value	Meaning
00H	Variants are not used.
03H–06H	The function determines whether the master-key variant is 03H, 04H, 05H, or 06H.

KeyBuffer
This parameter is a long pointer to the buffer that contains the 8-byte encryption key to be loaded.

SNBuffer
This parameter is a long pointer to an 8-byte buffer where the triple-encrypted device serial number is returned. The device serial number is encrypted under the loaded key.

rc This parameter is a word value that represents the return code from the PinLoadKey function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1804	The key parity is not valid for the requested function call.
1805	An error occurred for the PinOpen function call.

Remarks

The 4778 can store 256 encryption keys. When the keys pass to the 4778, they are triple-encrypted under a variant of the master key. The 4778 PIN keypad decrypts the master keys and checks the keys for valid parity. If the parity is valid, the key (in encrypted form) is placed in nonvolatile storage.

Loading the Initial-Chaining Value (PinLoadICV)

The PinLoadICV function call uses the PS/2 communication interface to load the initial-chaining value (ICV) into the 4778 PIN keypad.

PinLoadICV (PinHandle, VariantDescriptor, ICVBuffer, *rc*)

Parameters

The PinLoadICV function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

VariantDescriptor
This word value parameter specifies the master-key variant used to decrypt the ICV. The fields for this parameter are as follows:

Value	Meaning
00H	Do not use a variant.
02H	Use the master-key variant 2.

ICVBuffer
This parameter is a long pointer to a 16-byte buffer that contains the initial-chaining value.

rc This parameter is a word value that represents the return code from the PinLoadKey function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

The ICV is used to generate and verify the message-authentication-code functions. The ICV is encrypted under a variant of the master key.

Loading the PIN Verification Parameters (PinLoadVerifParms)

The PinLoadVerifParms function call loads the verification parameters that the 4778 PIN keypad requires to verify the PIN blocks and to generate the PIN offset data.

PinLoadVerifParms (PinHandle, PinLength, DecTblBuffer, *rc*)

Parameters

The PinLoadVerifParms function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

PinLength
This parameter is a word value that specifies the PIN length. When the 4778 PIN keypad verifies the PIN blocks and generates the PIN offset data, it checks the PIN length.

DecTblBuffer
This parameter is a long pointer to the 16-byte table that the 4778 uses to translate hexadecimal digits to decimal digits. Translation occurs at the same time that the 4778 verifies the PIN blocks and generates the offset data.

rc This parameter is a word value that represents the return code from the PinLoadVerifParms function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected a function call error.
1798	An error occurred in the PINCALLS.DLL file.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

This function call loads the verification parameters in nonvolatile storage. The 4778 uses these parameters to verify PIN blocks and generate offset data until the application program specifies new parameters.

Creating the 4704 PIN Block (PinReadPin4704)

The PinReadPin4704 function call reads a 4704-formatted PIN block from the 4778 PIN keypad and returns it to the application program.

PinReadPin4704 (PinHandle, PinBuffer, *rc*)

Parameters

The PinReadPin4704 function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

PinBuffer
This parameter is a long pointer to an 8-byte buffer where the encrypted PIN block returns.

rc This parameter is a word value that represents the return code from the PinReadPin4704 function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1799	The application program ended the operation.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

The 4704 PIN block is always encrypted under the resident master key. If the 4778 PIN keypad is currently operating in a nonencrypted mode, this function call results in an error. The pad character is hexadecimal F.

Creating the 3624 PIN Block (PinReadPin3624)

The PinReadPin3624 function call reads a 3624-formatted PIN block from the 4778 PIN keypad and returns it to the application program.

```
PinReadPin3624 (PinHandle, PinKeyInfo, PadCharacter, VariantDescriptor,
                PinKey, PinBuffer, rc)
```

Parameters

The PinReadPin3624 function call parameters are as follows:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

PinKeyInfo
This word value parameter describes the encryption key format used to create the PIN block. The fields for this parameter are as follows:

Bit	Description
15–2	These bits are reserved and must be set to zero.
1–0	These bits indicate the format of the encryption key.

Value	Meaning
00	Use the master key.
01	The encryption-key field is a 1-byte offset-key pointer.
10	This value is reserved.
11	Use the 8-byte encryption-key field.

PadCharacter
This parameter is a hexadecimal word value with a valid range of hexadecimal 0 through hexadecimal F.

Variant Descriptor
This word value parameter specifies the master-key variant used to decrypt the key. The fields for this parameter are as follows:

Value	Meaning
00H	Do not use a variant.
03H	Use the master-key variant 3.

PinKey
This parameter is a long pointer to a 1-byte or an 8-byte buffer. The buffer contains either a 1-byte key offset or an 8-byte encryption key (also called the session key) that is used to generate the PIN block.

Notes:

1. An encryption key that passes to the 4778 for a PIN block function must be encrypted under a variant of the master key.
2. A PIN block that is to be encrypted under the resident master key requires a PinKey parameter (used as a dummy pointer). The parameter must be on the stack as a place holder.

PinBuffer
This parameter is a long pointer to an 8-byte buffer where the encrypted PIN block returns.

rc This parameter is a word value that represents the return code from PinReadPin3624. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1799	The application program ended the operation.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

The PinReadPin3624 function call encrypts the formatted PIN block that returns to the application. To do this, the function call uses one of the following keys:

- The master key
- An internal 8-byte encryption key
- An 8-byte session key

When the PinReadPin3624 function call uses an internal 8-byte encryption key, the PinKey parameter points to the 1-byte buffer that indicates the offset of the key in the 4778. When it uses an 8-byte session key, it passes the session key to the 4778 only for the requested PIN block. If the 4778 PIN keypad is currently operating in the nonencrypted mode, this function call results in an error.

Creating the ANSI X9.8 PIN Block (PinReadPinAnsi98)

The PinReadPinAnsi98 function call reads an ANSI X9.8-formatted PIN block from the 4778 PIN keypad and returns it to the application program.

```
PinReadPinAnsi98 (PinHandle, PinKeyInfo, VariantDescriptor, PinKey,
                  PanBuffer, PinBuffer, rc)
```

Parameters

The PinReadPinAnsi98 function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

PinKeyInfo
This parameter is a word value that describes the encryption key format used to create the PIN block. The fields for this parameter are as follows:

Bit	Description
15–2	These bits are reserved and must be set to zero.
1–0	These bits indicate the format of the encryption key.

Value	Meaning
00	Use the master key.
01	The encryption-key field is a 1-byte offset-key pointer.
10	This value is reserved.
11	Use the 8-byte encryption-key field.

VariantDescriptor
This parameter is a word value that specifies the master-key variant used to decrypt the key. The fields for this parameter are as follows:

Value	Meaning
00H	Do not use a variant.
03H	Use the master-key variant 3.

PinKey
This parameter is a long pointer to a 1-byte buffer or an 8-byte buffer. The buffer contains either a 1-byte key offset or an 8-byte encryption key (session key) that is used to generate the PIN block.

Notes:

1. An encryption key that passes to the 4778 for a PIN block function must be encrypted under a variant of the master key.
2. If the PIN block is encrypted under the resident master key, the PinKey parameter is a dummy pointer and must be present on the stack as a place holder.

PanBuffer
This parameter is a long pointer to a 6-byte buffer that contains the personal account number.

PinBuffer
This parameter is a long pointer to an 8-byte buffer where the encrypted PIN block returns.

rc This parameter is a word value that represents the return code from PinReadPinAnsi98. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1799	The application program ended the operation.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

The PinReadPinAnsi98 function call uses one of the following to encrypt the formatted PIN block that is returned to the application.

The master key
An internal 8-byte encryption key
An 8-byte session key

When the PinReadPinAnsi98 function call uses an internal 8-byte encryption key, the PinKey parameter points to the 1-byte buffer. The 1-byte buffer indicates the offset of the key in the 4778.

The pad character is hexadecimal F.

When the PinReadPinAnsi98 function call uses an 8-byte session key, it passes the session key to the 4778 only for the requested PIN block. If the 4778 PIN keypad is currently operating in nonencrypted mode, the function call results in an error.

Verifying the PIN Block (PinVerifyPin)

The PinVerifyPin function call verifies a PIN entered from the 4778 PIN keypad. To perform the verification, the function call uses the offset data and verification data that is read from the customer's magnetic card.

```
PinVerifyPin (PinHandle, VerifyInfo, VariantDescriptor,
              PinKey, VerifData, OffsetData, rc)
```

Parameters

The PinVerifyPin function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

VerifyInfo
This parameter is a word value that specifies the encryption key format used to verify the PIN block and specifies whether to use offset data in the verification process. The fields for this parameter are as follows:

Bit	Description
15–5	These bits are reserved and must be set to zero.
4	This bit indicates whether the offset data should be used.

Value	Meaning
0	Do not use offset data in the verification process.
1	The encryption key is a 1-byte offset-key pointer.

Bit	Description
3–2	These bits are reserved and must be set to zero.
1–0	These bits indicate the format of the encryption key.

Value	Meaning
00	Use the master key.
01	The encryption-key field is a 1-byte offset-key pointer.
10	Reserved.
11	Use the 8-byte encryption-key field.

VariantDescriptor
This parameter is a word value that specifies the master-key variant used to decrypt the key. The fields for this parameter are as follows:

Value	Meaning
00H	Do not use a variant.
04H	Use the master-key variant 4.

PinKey
This parameter is a long pointer to a 1-byte or an 8-byte buffer. The buffer contains either a 1-byte key offset or an 8-byte encryption key (session key) that is used to generate the PIN block.

Notes:

1. An encryption key that passes to the 4778 for a PIN block function must be encrypted under a variant of the master key.
2. If the PIN block is encrypted under the resident master key, the PinKey parameter is a dummy pointer and must be present on the stack as a place holder.

VerifData
This parameter is a long pointer to an 8-byte buffer that contains the verification data.

OffsetData
This parameter is a long pointer to an 8-byte buffer that contains the offset data.

rc This parameter is a word value that represents a return code from the PinVerifyPin function call. The valid values are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1795	An error occurred in the data length.
1798	An error occurred in the PINCALLS.DLL file.
1799	The application program ended the operation.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1803	The PIN entry was not verified.
1805	A PinOpen function call error occurred.

Remarks

The PinVerifyPin function call verifies the PIN entries at the 4778 using the offset and verification data read from the customer's magnetic card. The 4778 uses the 3624 algorithm to verify the PIN block. This function call uses one of the following key formats in the verification process:

The master key
An internal 8-byte encryption key
An 8-byte session key

When this function call uses an internal 8-byte encryption key, the PinKey parameter points to the 1-byte buffer that indicates the offset of the key in the 4778. When it uses an 8-byte session key, it passes the session key to the 4778 only for the requested PIN block. If the 4778 is currently operating in a nonencrypted mode, this function call results in an error.

Creating the Offset Data (PinCreateOffsetData)

The PinCreateOffsetData function call generates PIN offset data for an entered PIN.

```
PinCreateOffsetData (PinHandle, PinKeyInfo, VariantDescriptor, PinKey,
                     VerifData, OffsetData, rc)
```

Parameters

The PinCreateOffsetData function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

PinKeyInfo
This parameter is a word value that describes the encryption key format used to create the PIN block. The fields for this parameter are as follows:

Bit	Description
15–2	These bits are reserved and must be set to zero.
1–0	These bits indicate the format of the encryption key.

Value	Meaning
00	Use the master key.
01	The encryption-key field is a 1-byte offset-key pointer.
10	Reserved.
11	Use the 8-byte encryption-key field.

VariantDescriptor
This parameter is a word value that specifies the master-key variant used to decrypt the key. The fields for this parameter are as follows:

Value	Meaning
00H	Do not use a variant.
04H	Use the master-key variant 4.

PinKey
This parameter is a long pointer to a 1-byte or an 8-byte buffer. The buffer contains a 1-byte key offset or an 8-byte encryption key (session key) used to generate the PIN block.

Notes:

1. An encryption key that passes to the 4778 for a PIN block function must be encrypted under a variant of the master key.
2. If the PIN block is encrypted under the resident master key, the PinKey parameter is a dummy pointer and must be present on the stack as a place holder.

VerifData
This parameter is a long pointer to an 8-byte buffer that contains the verification data.

OffsetData
This parameter is a long pointer to an 8-byte buffer that contains the offset data.

rc This parameter is a word value that represents the return code from the PinCreateOffsetData function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1795	An error occurred in the data length.
1797	The requested PIN mode is not valid.
1798	An error occurred in the PINCALLS.DLL file.
1799	The application program ended the operation.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

The PinCreateOffsetData function call generates PIN offset data. The 4778, using the 3624 algorithm, uses the PIN offset data to verify the PIN blocks. This function call creates the offset data by cryptographically combining the verification data with an entered PIN. It uses one of the following to encrypt the formatted PIN block:

The master key
An internal 8-byte encryption key
An 8-byte session key

When this function call uses an internal 8-byte encryption key, the PinKey parameter points to the 1-byte buffer, which indicates the offset of the key. If the offset data is less than 16 bytes long, the offset data that is returned is padded with FH characters.

When the function call uses an 8-byte session key, it passes the session key to the 4778 only for the requested PIN block. If the 4778 PIN keypad is currently operating in the nonencrypted mode, this function call results in an error.

Generating the Message Authentication Code (PinGenerateMac)

The PinGenerateMac function call generates a message authentication code from the input data string.

```
PinGenerateMac (PinHandle, MacInfo, KeyVariantDescriptor,
                IcvVariantDescriptor, MacKey, MacIcv, MacData,
                DataLen, Mac, rc)
```

Parameters

The PinGenerateMac function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

MacInfo
This parameter is a word value that describes the encryption key format and the ICV format that are used to generate the message authentication code. The fields for this parameter are as follows:

Bit	Description
15–8	These bits are reserved and must be set to zero.
7–4	These bits indicate the ICV used to generate the message authentication code.

Value	Meaning
0100	Use the resident ICV.
0110	Use the ICV that is in the MacIcv buffer.
xxxx	All other values are reserved.

Bit	Description
3–2	These bits are reserved and must be set to zero.
1–0	These bits indicate the encryption key format used to generate the message authentication code.

Value	Meaning
00	Use the master key.
01	The encryption key is a 1-byte offset-key pointer.
10	Reserved.
11	Use the 8-byte encryption-key field.

KeyVariantDescriptor
This parameter is a word value that specifies the master-key variant used to decrypt the key. The fields for this parameter are as follows:

Value	Meaning
00H	Do not use a variant.
05H	Use the master-key variant 5.

IcvVariantDescriptor
This parameter is a word value that specifies the master-key variant used to decrypt the ICV. The fields for this parameter are as follows:

Value	**Meaning**
00H	Do not use a variant.
02H	Use the master-key variant 2.

MacKey
This parameter is a long pointer to either a 1-byte or an 8-byte buffer. The buffer contains a 1-byte key offset or an 8-byte encryption key (session key) that is used to generate the message authentication code.

Notes:

1. An encryption key that passes to the 4778 for a message-authentication-code function must be encrypted under a variant of the master key.
2. If the message authentication code is generated using the master key, the MacKey parameter is a dummy parameter and must be present on the stack as a place holder.

MacIcv
This parameter is a long pointer to an 8-byte buffer that contains an 8-byte ICV (session ICV). The 8-byte ICV is used to generate the message authentication code (MAC).

Note: If a session ICV passes to the 4778 for a MAC function, the ICV must be encrypted under a variant of the master key. If the MAC function uses the resident ICV, this parameter must be on the stack as a place holder.

MacData
This parameter is a long pointer to the input data buffer that contains the data to be used to generate the message authentication code.

DataLen
This parameter is a word value that specifies the length of the input data buffer. The input buffer length must be a multiple of eight.

Mac
This parameter is a long pointer to an 8-byte buffer where the message authentication code returns.

rc This parameter is a word value that represents the return code from the PinGenerateMac function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

The PinGenerateMac function call generates a message authentication code from a data string that is up to 65 528 bytes (FFF8H) long, in multiples of 8 bytes. The function call uses one of the following key formats in the message-authentication-code generation process:

- The master key
- An internal 8-byte encryption key
- An 8-byte session key

When this function call uses an internal 8-byte encryption key, the MacKey parameter points to the 1-byte buffer that indicates the offset of the key in the 4778. When it uses an 8-byte session key, it passes the session key to the 4778 to generate the message authentication code.

Verifying the Message Authentication Code (PinVerifyMac)

The PinVerifyMac function call verifies a MAC from an input data string.

```
PinVerifyMac (PinHandle, MacInfo, KeyVariantDescriptor,
              IcvVariantDescriptor, MacKey, MacIcv, MacData, DataLen, rc)
```

Parameters

The PinVerifyMac function call uses the following parameters:

PinHandle

This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

MacInfo

This word value parameter describes the encryption key format and the ICV used to verify the message authentication code. The fields for this parameter are as follows:

Bit	Description
15–8	These bits are reserved and must be set to zero.
7–4	These bits indicate the ICV used to verify the message authentication code.

Value	Meaning
0100	Use the resident ICV.
0110	Use the ICV that is in the MacIcv buffer.
xxxx	All other values are reserved.

Bit	Description
3–2	These bits are reserved and must be set to zero.
1–0	These bits indicate the encryption key used to verify the message authentication code.

Value	Meaning
00	Use the master key.
01	The encryption key field is a 1-byte offset-key pointer.
10	Reserved.
11	Use the 8-byte encryption-key field.

KeyVariantDescriptor

This parameter is a word value that specifies the master-key variant used to decrypt the key. The fields for this parameter are as follows:

Value	Meaning
00H	Do not use a variant.
05H	Use the master-key variant 5.

IcvVariantDescriptor

This parameter is a word value that specifies the master-key variant used to decrypt the ICV. The fields for this parameter are as follows:

Value	Meaning
00H	Do not use a variant.
02H	Use the master-key variant 2.

MacKey

This parameter is a long pointer to a 1-byte or an 8-byte buffer. The buffer contains either a 1-byte key offset or an 8-byte encryption key (session key) used to verify the message authentication code.

Notes:

1. An encryption key that passes to the 4778 for a MAC function must be encrypted under a variant of the master key.
2. If the message-authentication-code function is encrypted using the master key, the MacIcv parameter is a dummy pointer and must be present on the stack as a place holder.

MacIcv

This parameter is a pointer to an 8-byte buffer. The buffer contains an 8-byte ICV (which is also called a session ICV) that is used to generate the message authentication code.

Note: If a session ICV passes to the 4778 for a MAC function, the ICV must be encrypted under a variant of the master key. If the message-authentication-code function uses the resident ICV, this parameter must be on the stack as a place holder.

MacData

This parameter is a long pointer to the input data buffer that contains the data used to verify the MAC.

DataLen

This parameter is a word value that specifies the input data buffer length. The input data buffer length must be a multiple of four.

rc This parameter is a word value that represents the return code from the PinVerifyMac function. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1803	The PIN entry was not verified.
1805	An error occurred for the PinOpen function call.

Remarks

The PinVerifyMac function call verifies a message authentication code that was generated elsewhere (such as at a different system node). The message authentication code is the last 4- or 8-bytes of the data that is pointed to by the MacData pointer. The data must be a multiple of four and a maximum of 65 532 bytes (FFFCH). This function call uses one of the following key formats in the message-authentication-code verification process to verify the data string:

The master key
An internal 8-byte encryption key
An 8-byte session key

When this function call uses an internal 8-byte encryption key, the MacKey parameter points to the 1-byte buffer that indicates the offset of the key in the 4778. When it uses an 8-byte session key, it passes the session key to the 4778 device to verify the message authentication code.

Running the Device Diagnostic Test (PinExecDevDiag)

The PinExecDevDiag function call runs the 4778 PIN keypad diagnostic tests. The results of the 4778 PIN keypad diagnostic tests are returned in the status buffer.

PinExecDevDiag (PinHandle, DiagTest, StatusBuffer, *rc*)

Parameters

The PinExecDevDiag function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

DiagTest
This parameter is a word value that identifies the 4778 diagnostic test to be performed. The fields for this parameter are as follows:

Value	Meaning
00H	Return the power-on diagnostic test status.
01H	Run and return the power-on diagnostic test status.
02H	Run and return the PIN keypad test status.
03H	Return the 4778 device-support code version information.
04H–FFFFH	These values are reserved.

StatusBuffer
This parameter is a long pointer to the status buffer that the diagnostic test returns. The DiagTest parameters return the following status fields:

DiagTest Values	Status Fields
00H and 01H	1 byte
02H	8 byte
03H	14 byte

The information in the status buffer has the following format:

DiagTest Command Byte	Returned Status Byte	Description
00H/01H	Bit 0	This bit is set to 1 if any of the keys on the keyboard are closed.
	Bit 1	This bit is set to 1 if the EEPROM code test failed.
	Bit 2	This bit is set to 1 if the RAM test failed.
	Bits 3–6	These bits are reserved.
	Bit 7	This bit is set to 0 if the 4778 device driver is in the nonencrypted mode. This bit is set to 1 if the 4778 device driver is in the encrypted mode.
02H	8 bytes	`FEB7B9253F35EB D` if the PIN keypad test is correct.
03H	14 bytes	ASCII string (*v.vv,mm/dd/yy*) where *v.vv*=version, *mm/dd/yy*=month/day/year.

rc This parameter is a word value that represents the return code from the PinExecDevDiag function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1799	The application program ended the operation.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

The PinExecDevDiag function call runs the device diagnostic tests. Each time the workstation power is switched on or the workstation is reset, the 4778 device driver uses this function call to force the application program to run the power-on diagnostic tests. The host diagnostics also use this function call to run the diagnostic tests, to run the keyboard test, and to read the device-support code information for field problem determination.

Reading the Device Serial Number (PinReadSN)

The PinReadSN function call reads the serial number of the 4778 PIN keypad.

PinReadSN (PinHandle, SNBuffer, *rc*)

Parameters

The PinReadSN function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

SNBuffer
This parameter is a long pointer to an 8-byte buffer where the serial number for the 4778 returns. The serial number format is as follows:

```
477841 sssssssFFH
```

The values for the serial number are as follows:

Value	Meaning
4778H	The machine type
41H	The location of the physical plant of control for the device
0*sssssss*H	The serial number that is unique to the device
FFH	The flag byte

rc This parameter is a word value that represents the return code from the PinReadSN function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1798	An error occurred in the PINCALLS.DLL file.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

All encryption-key management functions return the serial number of the device encrypted under the loaded (or entered) key; this function call lets the application program verify that the key was loaded correctly.

Reading the Nonencrypted Data (PinReadClearData)

The PinReadClearData function call reads nonencrypted data from the 4778.

PinReadClearData (PinHandle, DataLength, TransferCount, DataBuffer, *rc*)

Parameters

The PinReadClearData function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

DataLength
This parameter is a word value that indicates the amount of data to be read from the 4778 PIN keypad.

TransferCount
This parameter is a long pointer to a word buffer where the actual transfer count (the number of data bytes read from the device) returns.

DataBuffer
This parameter is a long pointer to the buffer where the nonencrypted data returns. The buffer can be a maximum of 32 digits; therefore, the data buffer should be large enough to hold all data that is entered on the 4778 (up to 32 bytes). If the data buffer is smaller than the actual amount of data that the 4778 returns, the device driver returns only the amount of data that the application program requests and then returns an error code.

rc This parameter is a word value that represents the return code from the PinReadClearData function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1795	An error occurred in the data length.
1797	The requested PIN mode is not valid.
1798	An error occurred in the PINCALLS.DLL file.
1799	The application program ended the operation.
1800	A parameter error occurred.
1801	The 4778 device driver is not open.
1802	The 4778 PIN keypad is not available.
1805	An error occurred for the PinOpen function call.

Remarks

The PinReadClearData function call reads the nonencrypted data from the 4778 PIN keypad. If the 4778 is currently operating in the encrypted mode, the function call results in an error.

Writing to the Display (PinWriteDisplay)

The PinWriteDisplay function call writes data to the display on the 4778 PIN keypad.

PinWriteDisplay (PinHandle, DisplayData, DataLength, *rc*)

Parameters

The PinWriteDisplay function call uses the following parameters:

PinHandle
This word value parameter is the 4778 device-driver handle that was obtained from the PinOpen function call.

DisplayData
This parameter is a long pointer to a buffer that contains the data to be displayed on the 4778.

DataLength
This parameter is a word value that indicates the number of bytes in the DisplayData buffer.

rc This parameter is a word value that represents the return code from the PinWriteDisplay function call. The valid return codes are:

0	No error.
1793	The 4778 PinHandle parameter is not valid.
1794	The 4778 PIN keypad detected an error in the requested function call.
1795	A data length error occurred.
1798	An error occurred in the PINCALLS.DLL file.
1799	The application program ended the operation.
1800	A parameter error occurred.
1806	The display data string was truncated.

Remarks

The 4778 has a 16-character, single-line display. If the DataLength parameter value is less than 16, the data is padded with ASCII blank (20H) characters to a length of 16. If the length is greater than 16, only the first 16 bytes are displayed and an error code is returned to the application.

Chapter 5. Data Formats

This chapter describes the data stream formats for:

- Magnetic-stripe data
- Nonencrypted PIN data
- Encrypted PIN data
- Personal identification numbers using 4704, 3624, and ANSI X9.8 formats

This chapter also includes information about managing cryptographic keys and using message authentication codes.

Magnetic-Stripe Data Format

This section describes the data formats that are used by the 4778 PIN-Pad Magnetic Stripe Reader magnetic-stripe-reader (MSR) feature.

Read Format

The 4778 PIN-Pad Magnetic Stripe Reader MSR device-support code reads the data for a single-track request or a double-track request and passes the data to the application program in the data buffer. Figure 5-1 shows the format of the single-track data and Figure 5-2 shows the format of the double-track data.

```
Lx      Sx      DATAx
```

Figure 5-1. Single-Track Read-Data Format

```
Lx      Sx      DATAx      Ly      Sy      DATAy
```

Figure 5-2. Double-Track Read-Data Format

The descriptions of the fields for the single-track and the double-track read-data formats are as follows:

Field	Description
L	Each L field represents a 1-byte field that defines the length of the respective S and DATA fields. When the 4778 MSR device-support code reads valid data from the magnetic stripe, the value in the L field is 03 or greater; at a minimum, it includes the SOM, the EOM, and the S field values. If the value in the L field is 01, the corresponding S field contains the status that indicates why the track was not read.

S Each S field represents a 1-byte status field that indicates the track associated with the respective DATA field. If the track did not contain a valid record, this field also contains the error status. The definitions of the bits in the S field are as follows:

Bit	Description
7	This bit indicates that an SOM value was not found and the 4778 device-support code interprets the track as a blank.
6	This bit indicates that an SOM value was found and an error was detected in the parity, the LRC, or the EOM value.
5–3	These bits are reserved.
2	This bit indicates that the data is from track 1.
1	This bit indicates that the data is from track 2.
0	Not used.

DATA Each DATA field represents the magnetic data that is read, including the SOM and EOM characters; the LRC value is excluded and is not returned to the application program. Each magnetic character read from the stripe returns as a single hexadecimal byte. If the L field is 01, the DATA field is not present; this indicates that valid data was not found.

If different PSOM and ASOM characters are defined in the parameter table, the 4778 device-support code accepts either character as the start-of-message character. If different PEOM and AEOM characters are defined, the 4778 device-support code accepts either character as the end-of-message character. For more information about the PSOM, ASOM, PEOM, and AEOM parameters, see “Loading the Device Parameters (MagLoadDevParms)” on page 3-4.

Single-Track Format

The 4778 device-support code checks the read data for validity. If the validity check is successful, the data returns to the application program in the data buffer. If the validity check fails, the Up arrow (↑) on the liquid crystal display (LCD) comes on. The operator can continue making read attempts until one succeeds, or the application program can issue the PinMagAbort function call to cancel the PinMagReadData function call.

The 4778 device-support code accumulates the number of read failures. To read this number, the application program issues the PinMagReadErrorStats function call. For more information about reading the error statistics for the 4778 MSR, see “Reading the Error Statistics (MagReadErrorStats)” on page 3-11.

Multiple-Track Format

When more than one track is specified in the ReadTracks parameter of the PinMagReadData function call, the OperationMode parameter of the PinMagSetOperation function call determines how the 4778 MSR device-support code processes a PinMagReadData function call. One or more tracks must contain valid data before the data can return to the application program. For more information about setting the parameters for the multitrack mode, see “Setting the Multitrack-Read Operation Mode (MagSetOperationMode)” on page 3-7.

If the PinMagSetOperation function call specifies that *all* tracks must have valid data, the 4778 MSR device-support code checks the data of each requested track for validity. If the validity check succeeds, the 4778 MSR device-support code returns the data to the application program in the data buffer. If one or more tracks fail the validity check, the read operation fails and the Up arrow indicator on the LCD is switched on. The operator can continue making read attempts until one succeeds, or the application program can issue the PinMagAbort function call to cancel the PinMagReadData function call.

If the PinMagSetOperation function call specifies that any *one* track must have valid data, control passes to the application program after the validity check completes and at least one of the requested tracks contains valid data. The 4778 MSR device-support code returns the data in the data buffer; the data includes an error status that indicates why the validity check of one or more of the tracks failed.

Reading Data

This section describes the default device-track parameters for a read operation and the device indicators that indicate the operating condition of the 4778 MSR. The 4778 device-support code automatically sets the default device-track parameters for a read operation when the application program issues a PinMagOpen function call. This enables the 4778 MSR to read any of the supported tracks. However, the 4778 MSR device-support code enables the application program to issue the PinMagLoadDevParms function call. This function call then loads user-specified parameters.

For the 4778 device driver, the *defaults* for the PinMagReadData function call and their corresponding tracks are as follows:

Track-1 Read

SOM = 05H
EOM = 1FH
Bits per character = 07H
Odd data parity
Even LRC parity

Track-2 Read

SOM = 0BH or 0DH
EOM = 0FH or 0CH
Bits per character = 05H
Odd data parity
Even LRC parity

User-Specified Operation

Your application program can override the default parameters and set new parameters in the 4778 MSR device-support code. To load the 4778 MSR parameters, the application program issues the PinMagLoadDevParms function call. This function call sets the operational parameters of the 4778 MSR and determines how the data passes from each track. For more information about the PinMagLoadDevParms function call, see “Loading the Device Parameters (MagLoadDevParms)” on page 3-4.

Indicators for Read Operations

On the 4778, the Up arrow indicator on the liquid crystal display (LCD) indicates the operating condition of the device.

For single-track read requests, the Up Arrow indicator display comes on (1) when the 4778 MSR initially prepares for a read operation and (2) when the validity check fails.

For multiple-track read requests, the PinMagSetOperation specifies one of the following options:

- All tracks must contain valid data.
- At least one track must contain valid data.

The Up arrow indicator comes on to indicate a read operation failure based on the option that was selected.

The Up arrow indicator remains on until:

- A subsequent, successful read occurs.
- The application program issues a PinMagAbort function call to cancel the operation.

PIN Data Formats

This section describes the PIN formats that the 4778 PIN keypad supports. It includes the following topics:

- Nonencrypted PIN format
- Encrypted PIN format
- 4704 encrypting-PIN-pad (EPP) PIN format
- 3624 PIN format
- Verifying the 3624 PINs
- ANSI X9.8 format

Nonencrypted PIN Data Format

For nonencrypted data, the maximum data length that the 4778 PIN keypad can store in a buffer and return is 32 digits. The 4778 device driver stores the ASCII data, along with the 7FH header and trailer bytes, in the data buffer of the application program. The 4778 device driver translates digits 0 through 9 into ASCII codes as follows:

PIN Keypad Key	**Output to Application**
0	30H
1	31H
2	32H
3	33H
4	34H
5	35H
6	36H
7	37H
8	38H
9	39H

Figure 5-3. Translations for Nonencrypted PIN Keypad Data

Encrypted PIN Data Format

For encrypted data, the application program issues the Create PIN Block command to allow you to enter an encrypted PIN at the PIN keypad. The 4778 PIN keypad transfers the data to the 4778 device driver. When the customer presses the **End** key on the 4778 PIN keypad, the data is transferred to the 4778 PIN keypad device-support code. If the customer presses the **Erase** key, the 4778 PIN keypad purges the keyed data and restarts the 4778 PIN keypad entry.

The 4778 PIN keypad returns 8 bytes of encrypted data to the 4778 device driver, unless the customer presses the **End** key without entering PIN data. Encrypted data returns to the application program in the application-supplied buffer.

The 4778 supports three different formats of encrypted PINs:

4704 EPP
ANSI X9.8
3624

The following sections describe these formats in detail.

4704 EPP Format

The format for the 4704 EPP PIN is as follows:

```
              8 bytes

     LEN PIN .......PAD SEQ

     A   123456789  FFF

         encipher using KP
```

Figure 5-4. 4704 EPP PIN Format

LEN	Number of PIN characters entered; a 4-bit value from 1H to DH
PIN	From 1 to 13 PIN characters; each is a 4-bit value from 0H to 9H
PAD	From 0 to 12 pad characters (13 minus the number of PIN characters); each is a 4-bit value, always FH
SEQ	A 1-byte sequence number, from 00H to FFH

ANSI X9.8 Format

The format for the ANSI X9.8 is as follows:

```
                 8 bytes


            LEN  PIN.........PAD   Plain text PIN



                 PAN............   Primary Account Number



            encipher using Kp

  Example:
          6 12 34 56 FF FF FF FF   (Customer PIN: 123456)


               22 23 33 44 45 55   (PAN: 111 222 333 444 555)


          6 12 16 75 CC BB BA AA   Formatted PIN (PIN XOR PAN)
```

Figure 5-5. ANSI X9.8 PIN Format

0	A 4-bit control field; always 0H.
LEN	Number of PIN characters entered; a 4-bit value from 4H to CH.
PIN	From 4 to 12 PIN characters; each is a 4-bit value from 0H to 9H.
PAD	From 2 to 10 keypad characters (14 minus the number of PIN characters). Each 4-bit character must be set to FH.
0000	A 2-byte field; always 0000H.
PAN	Twelve 4-bit digits representing the rightmost 12 digits of the primary account number (PAN).
XOR	An exclusive-OR of the plain text PIN and the PAN yields the formatted PIN.

3624 PIN Format

The format for the 3624 PIN is as follows:

```
              8 bytes

       PIN ...............PAD

        1 23 45 6E EE EE EE EE

        ††encipher using Kp†† |
```

Figure 5-6. 3624 PIN Format

PIN From 1 to 16 PIN characters; each is a 4-bit value from 0H to 9H.

PAD From 0 to 15 pad characters (16 minus the number of PIN characters). Each is a 4-bit value, always 0H to FH; all values must be the same.

Verifying the 3624 PINs

The PIN verification process compares the PIN that the customer enters with the validation data encrypted on the customer's identification card. This determines whether the customer entered the correct PIN. The 4778 PIN keypad uses a verification algorithm that is identical to the algorithm used in the IBM 3624 Consumer Transaction Facility and other IBM products.

To verify PINs, the 4778 PIN keypad requires the following information:

Item	**Description**
Validation Data	The data on the customer's card that the 4778 PIN keypad compares to the PIN that the customer enters.
EPINKEY	The key that the 4778 PIN keypad uses to encrypt the validation data.
Offset Data	The optional data that the 4778 PIN keypad requires if random or customer-selected PINs are used.
DECTAB	The decimalization table that the 4778 PIN keypad uses to translate hexadecimal numbers to decimal numbers. The DECTAB is used to compare the PIN digits.
PINMINL	The number of PIN digits that the EPP is to check.

The method that the 4778 PIN keypad uses to verify PINs is shown in Figure 5-7 on page 5-8 and works as follows:

1. The application, using the 4778 MSR device-support code, reads the validation data from the identification card, pads the data to 8 bytes (16 digits) if required, and passes it to the EPP along with the EPINKEY location (or encrypted key) and the PINMINL.
2. The 4778 PIN keypad encrypts the validation data with the EPINKEY and converts the data to decimal using the DECTAB.
3. The 4778 PIN keypad reads the customer's PIN input.

4. The *n* leftmost characters (*n* is the length of the entered PIN) of the decimalized validation data form the intermediate PIN.
5. To form the PIN check number, the *m* rightmost digits (*m* is PINMINL) of the intermediate PIN are added to the offset data, modulo 10 (without carry).
6. The 4778 PIN keypad compares the *m* rightmost digits of the entered PIN with the PIN check number and returns the results of the comparison to the application.

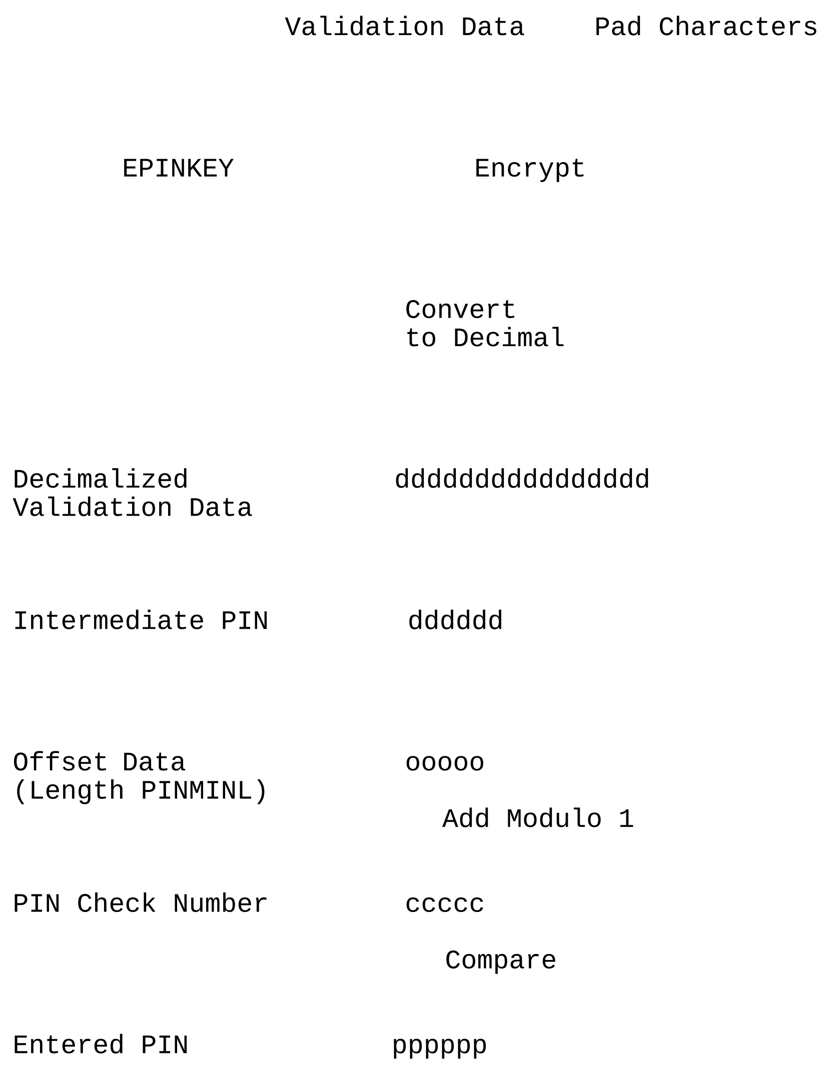

Figure 5-7. Verifying the 3624 PINs

When you create the PIN offset data for the magnetic stripe of a new customer or when a consumer changes a PIN, you use the same algorithm. The only difference is that, instead of adding the offset data modulo 10 to the intermediate PIN to compute the PIN check number, the entered PIN subtracts modulo 10 from the intermediate PIN to compute the offset data to place on the customer's card.

Managing the Cryptographic Keys

This section describes the cryptographic functions that the 4778 PIN keypad supports and how you use them. It includes the following topics:

- Loading keys
- Using key variants
- Converting a master key to the keypad-entry format

To use cryptography with a 4700 Finance Communication System, you should be familiar with the *IBM 4700 Finance Communication System, Controller Programming Library; Volume 5: Cryptographic Programming*.

Because the data-encryption algorithm (DEA) is in the public domain, the security of the functions of the 4778 PIN keypad that use the DEA depends on the security of the key that is used in processing the algorithm. Therefore, after you load or enter cryptographic keys into the 4778 PIN keypad, the keys cannot be read. They are placed in nonvolatile EEPROM storage that resides in a tamper-resistant security processor.

You can design a secure method for handling your keys when you are isolated from the PIN keypad, using the provisions for loading the keys. Randomly generate your keys, and store and distribute your keys in a secure, controlled manner that you can audit.

Loading Keys

The first key that you load into the keypad is the master key. You must load (or enter) the master key into the keypad before you can use any cryptographic operations. This is the only key that is loaded into the keypad in a nonencrypted form. For protection, this key should be 128 bits (16 bytes) long. However, for compatibility with the existing 4704 encrypting-PIN-pad feature, you can load a 64-bit (8-byte) master key. When you load an 8-byte master key, the 8 bytes are duplicated. This ensures that a full 16 bytes are available for the key management functions. These functions can then use your master key.

Verifying a DES key

After the successful entry or the loading of a DES key, the triple-encrypted serial number is returned to the application. You should decrypt the serial number to verify that the key was loaded correctly. The encrypted result of the serial number remains displayed until you press a key on the PS/2 keyboard.

Warning: Placing the 4778 PIN keypad into the encrypted mode destroys all the loaded DES keys. Following the PinSetModeEncrypt request, you must reload all the DES keys into the 4778 PIN keypad.

Triple-Encrypted Keys

After you load the master key, you can load additional 8-byte keys into the keypad (if you desire). You load these keys after they are triple-encrypted, under the master key or the variant of the master key (for an explanation of variants, see "Using Key Variants"). Triple encryption is a cryptographic process in which you do the following:

1. Encrypt the 8 bytes of data with the first 8 bytes of a double-length key.
2. Decrypt the result with the second 8 bytes of the double-length key.
3. Encrypt the result again, using the first 8 bytes of the double-length key.

If you use the same 8 bytes for the encryption and the decryption steps (for an 8-byte master key), the final result is the same as if a single encryption step is performed with a single-length (8-byte) key.

The 4778 can store 256 keys (in addition to the master key). These keys are placed in the nonvolatile EEPROM storage and triple encrypted under the appropriate master-key variant until they are used.

Using Key Variants

A variant of a cryptographic key is a new key that is formed by combining the original key with a nonsecret quantity. In the 4778 PIN keypad, the nonsecret quantity is called a variant descriptor byte (VDB). To produce the new key, each byte of the original key is combined in an exclusive-OR operation with the VDB.

For example:

```
Original key:       1 23 45 67 89 AB CD EF    (hexadecimal)
VDB:               55

Perform an exclusive-OR operation with the VDB with each byte
of the original key to obtain the variant key as follows:

                 1 23 45 67 89 AB CD EF
            XOR  55 55 55 55 55 55 55 55

Variant key:       54 76 1  32 DC FE 98 BA
```

The 4778 PIN keypad contains a fixed table (Figure 5-8) of variant-descriptor bytes. The table is organized as 16 sets of four VDBs. Only six variant bytes are defined, corresponding to the six PIN commands that require them. Whenever you use a variant with a command, you must specify a variant descriptor that designates which of the VDB sets to use.

Figure 5-8. Key Variants

Index	Variant a	Variant b	Variant c	Variant d
1	12H	xxH	xxH	xxH
2	90H	xxH	xxH	xxH
3	06H	xxH	xxH	xxH
4	2EH	xxH	xxH	xxH
5	44H	xxH	xxH	xxH
6	82H	xxH	xxH	xxH
7–16	xxH	xxH	xxH	xxH

Note: `'00H'` is a special case. It indicates that variants are not to be used at all. This is equivalent to using a variant of 00H.

Each command can include only certain variants as shown in Figure 5-9.

Figure 5-9. Variant Descriptor Bytes for the 4778 Commands

Command	Variant use
Load Key	Use variant a3, a4, a5, or a6 to decrypt the key that is being loaded.
Load ICV	Use variant a2 to decrypt the ICV that is being loaded.
Create PIN Block	Use variant a3 to decrypt the PIN key.
Verify PIN	Use variant a4 to decrypt the PIN verification key.
Generate MAC	Use variant a5 to decrypt the MAC key.
Verify MAC	Use variant a6 to decrypt the MAC key.

Variants ensure that a key can only be used for its intended function. For example, a security problem could result if a MAC verification key could also be used for the MAC generation function. To prevent this problem, each key is stored in encrypted form (encrypted under a variant of the master key by using one of the variant descriptor bytes). The VDB is specified with the Load Key command. When you use the VDB for the intended function, the key is decrypted using the correct variant of the master key (and is successfully recovered). If you use the VDB for a different function, the wrong variant of the master key is used, resulting in an incorrect key.

Converting a Master Key to the Keypad-Entry Format

Master keys are generated as either 8-byte or 16-byte values. To enter the key into the 4778 PIN keypad with the Enter Master Key command, first convert the key to a format that contains only digits 0 through 9. In this process, convert each byte of the key into three keystrokes. For example, convert an 8-byte key to 24 keystrokes and a 16-byte key to 48 keystrokes.

To express the key in the form of keystrokes, write the key in hexadecimal, then use the tables in Figure 5-11 on page 5-13 to convert each pair of hexadecimal digits to 3-keystroke values.

For example:

```
      73 A  11 C3 8  6F CE 22       Key




      Convert each pair,
      using the hexadecimal
      to keystroke tables.



   3 4 3
     5   1
         4
          6   2
             4
                3 3 2
                  6 3 2
                     1   3

Enter this sequence on the
keypad     3 4 3 5   1   4   6   2 4     3 3 2 6 3 2 1   3
```

Figure 5-10. Example of a Master Key Conversion

When you generate keys by a random process, any hexadecimal character is possible. Because the keypad has only decimal characters, you must translate the Enter Master Key command bytes (which appear as two hexadecimal characters) into the 3-3-2 decimal format. To translate these bytes, use the table in Figure 5-11 on page 5-13.

When you read the table, notice that multiple hexadecimal bytes result in identical decimal input (00H and 01H both result in 001D). This is because each byte is required to have odd parity (the parity bit is the least-significant bit). This means that if you entered a byte such as 000D, a parity error occurs. If keys are generated by an automatic process, the process corrects the parity. This ensures that identical keys of the correct parity reside at all the nodes.

Note: Do not use this table for routine data conversions to and from the 3-3-2 format, because the parity is accounted for in the table. Use this table only to encrypt and generate the key and the keystroke.

1H			1	21H	1			41H	2			61H	3		1
2H			2	22H	1		3	42H	2		3	62H	3		2
3H			2	23H	1		3	43H	2		3	63H	3		2
4H	1			24H	1	1	1	44H	2	1	1	64H	3	1	
5H	1			25H	1	1	1	45H	2	1	1	65H	3	1	
6H	1	3		26H	1	1	2	46H	2	1	2	66H	3	1	3
7H	1	3		27H	1	1	2	47H	2	1	2	67H	3	1	3
8H	2			28H	1	2	1	48H	2	2	1	68H	3	2	
9H	2			29H	1	2	1	49H	2	2	1	69H	3	2	
AH	2	3		2AH	1	2	2	4AH	2	2	2	6AH	3	2	3
BH	2	3		2BH	1	2	2	4BH	2	2	2	6BH	3	2	3
CH	3	1		2CH	1	3		4CH	2	3		6CH	3	3	1
DH	3	1		2DH	1	3		4DH	2	3		6DH	3	3	1
EH	3	2		2EH	1	3	3	4EH	2	3	3	6EH	3	3	2
FH	3	2		2FH	1	3	3	4FH	2	3	3	6FH	3	3	2
1 H	4			3 H	1	4	1	5 H	2	4	1	7 H	3	4	
11H	4			31H	1	4	1	51H	2	4	1	71H	3	4	
12H	4	3		32H	1	4	2	52H	2	4	2	72H	3	4	3
13H	4	3		33H	1	4	2	53H	2	4	2	73H	3	4	3
14H	5	1		34H	1	5		54H	2	5		74H	3	5	1
15H	5	1		35H	1	5		55H	2	5		75H	3	5	1
16H	5	2		36H	1	5	3	56H	2	5	3	76H	3	5	2
17H	5	2		37H	1	5	3	57H	2	5	3	77H	3	5	2
18H	6	1		38H	1	6		58H	2	6		78H	3	6	1
19H	6	1		39H	1	6		59H	2	6		79H	3	6	1
1AH	6	2		3AH	1	6	3	5AH	2	6	3	7AH	3	6	2
1BH	6	2		3BH	1	6	3	5BH	2	6	3	7BH	3	6	2
1CH	7			3CH	1	7	1	5CH	2	7	1	7CH	3	7	
1DH	7			3DH	1	7	1	5DH	2	7	1	7DH	3	7	
1EH	7	3		3EH	1	7	2	5EH	2	7	2	7EH	3	7	3
1FH	7	3		3FH	1	7	2	5FH	2	7	2	7FH	3	7	3

Figure 5-11 (Part 1 of 2). Hexadecimal-to-Keystroke Conversion Table

8 H	4			A H	5		1	C H	6		1	E H	7		
81H	4			A1H	5		1	C1H	6		1	E1H	7		
82H	4		3	A2H	5		2	C2H	6		2	E2H	7		3
83H	4		3	A3H	5		2	C3H	6		2	E3H	7		3
84H	4	1	1	A4H	5	1		C4H	6	1		E4H	7	1	1
85H	4	1	1	A5H	5	1		C5H	6	1		E5H	7	1	1
86H	4	1	2	A6H	5	1	3	C6H	6	1	3	E6H	7	1	2
87H	4	1	2	A7H	5	1	3	C7H	6	1	3	E7H	7	1	2
88H	4	2	1	A8H	5	2		C8H	6	2		E8H	7	2	1
89H	4	2	1	A9H	5	2		C9H	6	2		E9H	7	2	1
8AH	4	2	2	AAH	5	2	3	CAH	6	2	3	EAH	7	2	2
8BH	4	2	2	ABH	5	2	3	CBH	6	2	3	EBH	7	2	2
8CH	4	3		ACH	5	3	1	CCH	6	3	1	ECH	7	3	
8DH	4	3		ADH	5	3	1	CDH	6	3	1	EDH	7	3	
8EH	4	3	3	AEH	5	3	2	CEH	6	3	2	EEH	7	3	3
8FH	4	3	3	AFH	5	3	2	CFH	6	3	2	EFH	7	3	3
9 H	4	4	1	B H	5	4		D H	6	4		F H	7	4	1
91H	4	4	1	B1H	5	4		D1H	6	4		F1H	7	4	1
92H	4	4	2	B2H	5	4	3	D2H	6	4	3	F2H	7	4	2
93H	4	4	2	B3H	5	4	3	D3H	6	4	3	F3H	7	4	2
94H	4	5		B4H	5	5	1	D4H	6	5	1	F4H	7	5	
95H	4	5		B5H	5	5	1	D5H	6	5	1	F5H	7	5	
96H	4	5	3	B6H	5	5	2	D6H	6	5	2	F6H	7	5	3
97H	4	5	3	B7H	5	5	2	D7H	6	5	2	F7H	7	5	3
98H	4	6		B8H	5	6	1	D8H	6	6	1	F8H	7	6	
99H	4	6		B9H	5	6	1	D9H	6	6	1	F9H	7	6	
9AH	4	6	3	BAH	5	6	2	DAH	6	6	2	FAH	7	6	3
9BH	4	6	3	BBH	5	6	2	DBH	6	6	2	FBH	7	6	3
9CH	4	7	1	BCH	5	7		DCH	6	7		FCH	7	7	1
9DH	4	7	1	BDH	5	7		DDH	6	7		FDH	7	7	1
9EH	4	7	2	BEH	5	7	3	DEH	6	7	3	FEH	7	7	2
9FH	4	7	2	BFH	5	7	3	DFH	6	7	3	FFH	7	7	2

Figure 5-11 (Part 2 of 2). Hexadecimal-to-Keystroke Conversion Table

Using Message Authentication Codes

The 4778 produces a message authentication code (MAC) using the conventions that are defined in ANSI X9.9. A MAC ensures data integrity when an unprotected communication link transmits a message from one node to another node. The MAC is generated at the sending node and is sent with the message to the other node.

When the other node receives the message authentication code, the code is verified to ensure that it is the same as the code that was transmitted by the sending node. If the MAC is not the same, you can assume that some of the data was either intentionally or unintentionally changed. The algorithm that is shown in Figure 5-12 on page 5-15 is applied either to the entire message or to specific authentication elements that are presented to the 4778 PIN keypad by the workstation. The data must be a multiple of 8 bytes (no padding or element extraction is provided by either the 4778 PIN keypad or the device driver). The algorithm uses the cipher-block-chaining mode of the data encryption standard (DES).

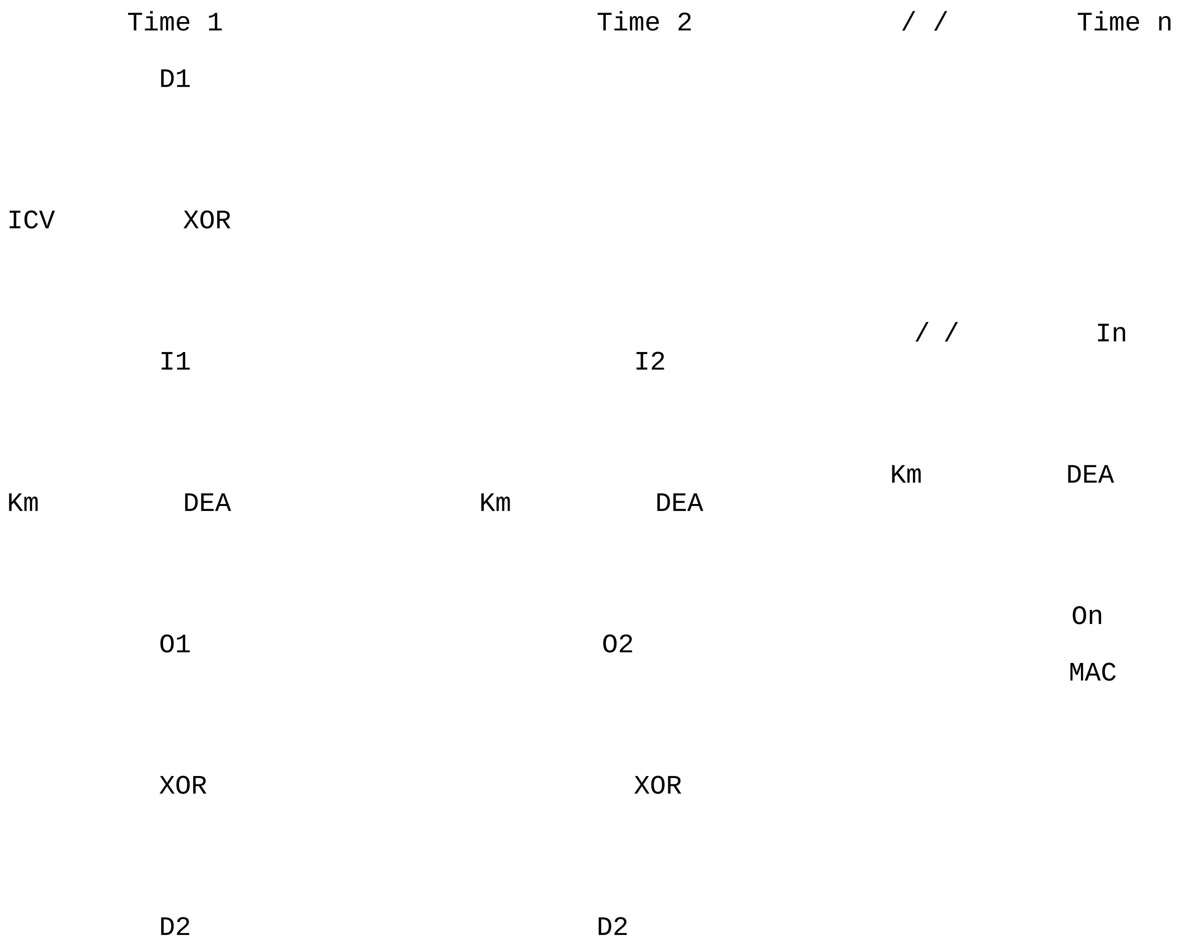

Figure 5-12. Using Message Authentication Codes

The following list defines the terms in Figure 5-12:

ICV	Initial Chaining Vector (8 bytes, preloaded or included with the command request)
D1 – Dn	8-byte data blocks
In	8-byte intermediate value
Km	MAC key
On	8-byte message authentication code (leftmost 4 bytes are used as MAC in ANSI X9.9)

Chapter 6. 4777 Multiple-Virtual-DOS-Machine I/O System

This chapter describes the application program interface (API) for the multiple-virtual-DOS-machine (MVDM) support code 4777 Magnetic Stripe Unit. This API lets you operate the 4777 device on an OS/2 Release 2.0 or higher system using the MVDM environment.

4777 MVDM I/O System

The 4777 MVDM I/O system consists of the following:

4777 DOS emulation device driver (4777DD.SYS)
4777 virtual device driver (4777VDD.SYS)
4777 support application (4777SAP.EXE)
4777 dynamic link library (MAGCALLS.DLL)
4777/4778 physical device driver (FIOSERDD.SYS or FIOAUXDD.SYS)

Figure 6-1 shows the relationship of the 4777 MVDM I/O system components.

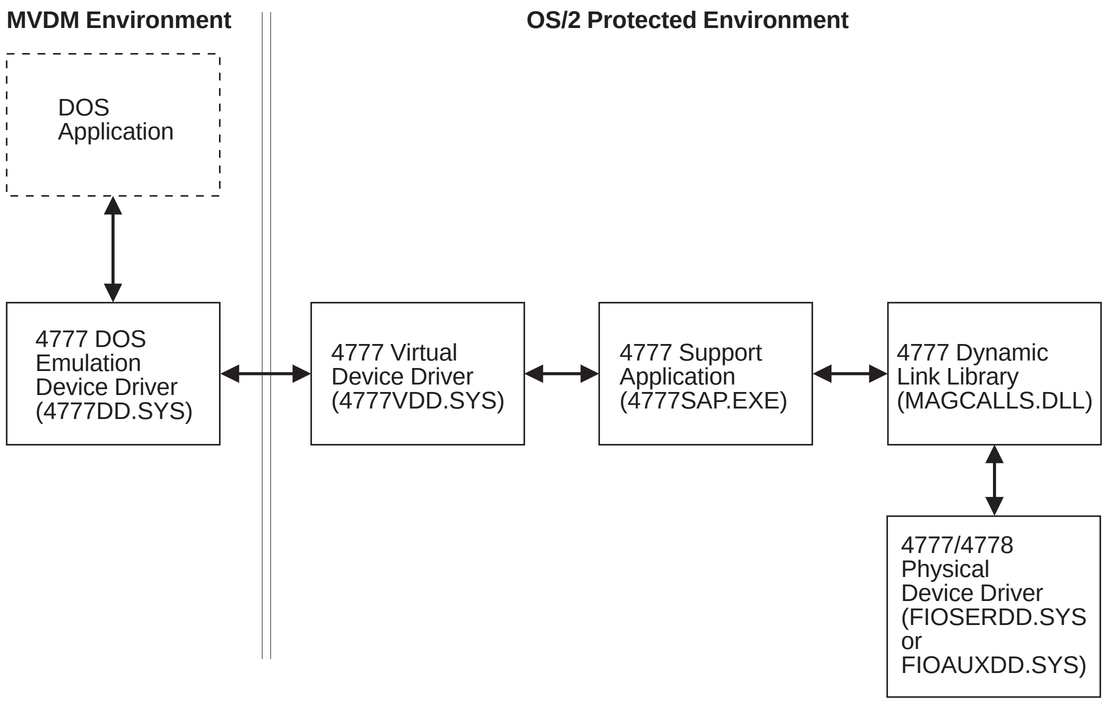
MVDM Environment
OS/2 Protected Environment
DOS Application
4777 DOS Emulation Device Driver (4777DD.SYS)
4777 Virtual Device Driver (4777VDD.SYS)
4777 Support Application (4777SAP.EXE)
4777 Dynamic Link Library (MAGCALLS.DLL)
4777/4778 Physical Device Driver (FIOSERDD.SYS or FIOAUXDD.SYS)

Figure 6-1. 4777 MVDM I/O System

Loading the 4777 MVDM Device Driver

You can load the MVDM device driver using the FINSTALL program or you can install it manually. The automatic and manual procedures are described in the following sections.

Note: For serial port attachment do not attach a 4717 or 4718 to the same workstation used for the 4777. The device driver for the 4777 is not compatible with the 4700 devices.

Automatic Installation

To load and install the 4777 MVDM I/O system using FINSTALL:

1. Insert the OS/2 device-driver diskette into the A: drive.
2. Enter `A:FINSTALL` at the command prompt and press **Enter**.
3. Follow the instructions on the panels displayed by the installation program.

   Note: If you use a virtual machine boot (VMB) DOS session, you must add the device-driver statements to the CONFIG.SYS file that is used to start the DOS session.

When you are finished, the program (FINSTALL) updates the files that contain the device-specific code and copies them to the drive you specified.

Manual Installation

This procedure loads and initializes the 4777 MVDM I/O system without using the FINSTALL program. In the CONFIG.SYS syntax, the disk-drive identifier and directory path are defined as:

d:	The disk-drive identifier
path	The directory path

Brackets [ ] indicate optional parameters.

1. Copy the following files from the device-driver diskette to your workstation hard disk:

   4777DD.SYS
   4777VDD.SYS
   4777SAP.EXE

2. Add the following statements to your CONFIG.SYS file. The device-driver statements must be in the CONFIG.SYS file in the root directory of the hard disk used to start your OS/2 session.

   a. For the 4777 DOS emulation device driver:

   ```
   DEVICE=[d:[path]]4777DD.SYS [/X/Y/K/Z: ;val]
   ```

   The 4777DD device statement parameters are:

   /X Suppresses the messages that are displayed if 4777DD.SYS detects any error conditions. Normally, all messages are displayed and processing stops until you press a key. When using this option, the application program must handle all error messages.

/Y Prevents setting the error bit and returning error codes to DOS. All error status information is blocked. The application program must issue a Read Status request to verify the results of any operations.

/K:*val*
Assigns the Cancel operation to a key that you select, where *val* is a 3-digit decimal value. Use the format `/K:val`, when *val* is the standard ASCII code for the selected key. If the key represents an extended ASCII code, use the format: `/K: ;val`.

The default setting assigns Cancel to the Esc key (ASCII code 27). This key assignment is valid only during synchronous operation.

/Z Postpones any DOS OPEN error codes if the 4777 device is currently controlled by another MVDM session or by an OS/2 session. This lets you see the pop-up message, then go to the owning session and issue a Close function to release the 4777 device. Otherwise, the Open error is passed to the application.

b. For the 4777 virtual device driver:

```
DEVICE=[d:[path]]4777VDD.SYS
```

c. For the 4777 support application:

```
RUN=[d:[path]]4777SAP.EXE
```

3. Load and initialize the 4777 dynamic link library.

   For information about loading the dynamic link library for the 4777 device, see “Loading the 4777 Dynamic Link Library” on page 2-3.

4. Load and initialize the FIOSERDD.SYS device driver if serial attached or the FIOAUXDD.SYS device driver if auxiliary port attached.

5. Load and initialize the message files.

Using the 4777 MVDM Device Driver

The 4777 device driver for OS/2 MVDM sessions provides:

- DOS-defined functions for the MVDM and VMB environments under the OS/2 operating system.
- Serial and interleaved sharing of the 4777 hardware between OS/2 protected-mode applications and DOS applications in the MVDM environment.
- Use of the 4777 MVDM I/O system in a single MVDM session. All subsequent MVDM sessions are prevented from using the 4777 MVDM I/O system.

DOS Application Program Interface

The 4777 MVDM I/O system uses the DOS interrupt 21H application program interface. The interrupt 21H interface is described in the *4777 Magnetic Stripe Unit DOS Programming Guide*. The interface is compatible with the DOS device driver, MSRE2DD.SYS, with the following exceptions:

- The IOCTL Write subfunctions for reading and writing data in the TopView* environment are not supported.
- A pop-up message is displayed when the MVDM application tries to use the 4777 device and it is currently in use by an OS/2 session.

  You can clear this condition by switching to the OS/2 session and issuing a Close request. Then you can request a retry of the Open request or return the error to the application program.

  Note: The pop-up message is not visible during a DOS full-screen session although the workstation sounds a beep when the message is issued. You can view the message by bringing a windowed session to the foreground or by running the application in a DOS window.
- If the application issues a DOS Close request to the MVDM I/O interface when the device is set to read or encode data, the 4777 device indicators are turned off and the device is disabled. This ensures that the 4777 is available in a reset state for any OS/2 protected-mode session application.
- If the application issues multiple Open requests without corresponding Close requests in between, a DOS Open error is returned to the DOS application.
- If another MVDM session is already using the 4777 device, a DOS Open error is returned to the application program. This error causes the FIO0573 error message to appear.

The DOS emulation device driver (4777DD.SYS) processes all 21H interrupts with the MAGDEV handle name.

OS/2 Protected-Mode Operation

The 4777 MVDM I/O system shares a common interface with the 4777 OS/2 protected-mode support programming (see Figure 6-1 on page 6-1). The OS/2 physical device driver determines which mode is in control of the device. When the MVDM environment is in control, Open requests from protected-mode applications receive an ERROR_NOT_READY return code from the OS/2 session.

Although the MVDM and OS/2 protected-mode applications can run concurrently, only one session can own the device at any time. The ownership is on a first-use basis. That is, the first application requesting the device is given control and this determines which API is used. After an application releases control of the device, the I/O system is available to the first application that requests control.

The 4777 MVDM support programs use MAGCALLS.DLL and FIOSERDD.SYS or FIOAUXDD.SYS to perform the actual I/O operations requested by the DOS application.

* Trademark of IBM

Avoiding Unwanted Initialization Error Messages

The 4777 DOS emulation device driver will install by default in each DOS session when the session is started. The device driver will attempt to communicate with the 4777 using the other I/O system components during initialization. If this is not possible, the device driver displays a message. This message could be considered a nuisance, especially if you want to use the I/O system only in the first DOS session to be started and also want to start additional DOS sessions that do not use the I/O system.

To avoid this condition and maintain other DOS sessions that do not use the MVDM I/O system, you can:

- Use the /X option on the DEVICE statement. Use caution with this solution because all error messages are suppressed. This can result in other error conditions being hidden from the user.
- Create separate command prompts in the Command Prompts–Icon View to start DOS sessions with, and without, the device driver in the device-driver DOS settings. There can be a single with-device-driver DOS session started using the appropriate prompt icon.

Chapter 7. 4778 Multiple-Virtual-DOS-Machine I/O System

This chapter describes the application program interface (API) for the multiple-virtual-DOS-machine (MVDM) I/O system for the 4778 PIN-Pad Magnetic Stripe Reader. This API lets you operate the 4778 on an OS/2 Release 2.0 (or higher) system using the MVDM environment.

4778 MVDM I/O System

The 4778 MVDM system consists of the following:

4778 PIN DOS emulation device driver (4778DDP.SYS)
4778 MSR DOS emulation device driver (4778DDM.SYS)
4778 PIN virtual device driver (4778VDDP.SYS)
4778 MSR virtual device driver (4778VDDM.SYS)
4778 PIN support application (4778SAPP.EXE)
4778 MSR support application (4778SAPM.EXE)
4778 PIN dynamic link library (PINCALLS.DLL)
4778 MSR dynamic link library (PINMSR.DLL)
4778 physical device driver (FIOSERDD.SYS) or (FIOAUXDD.SYS).

Figure 7-1 shows the relationship of the 4778 MVDM I/O system components.

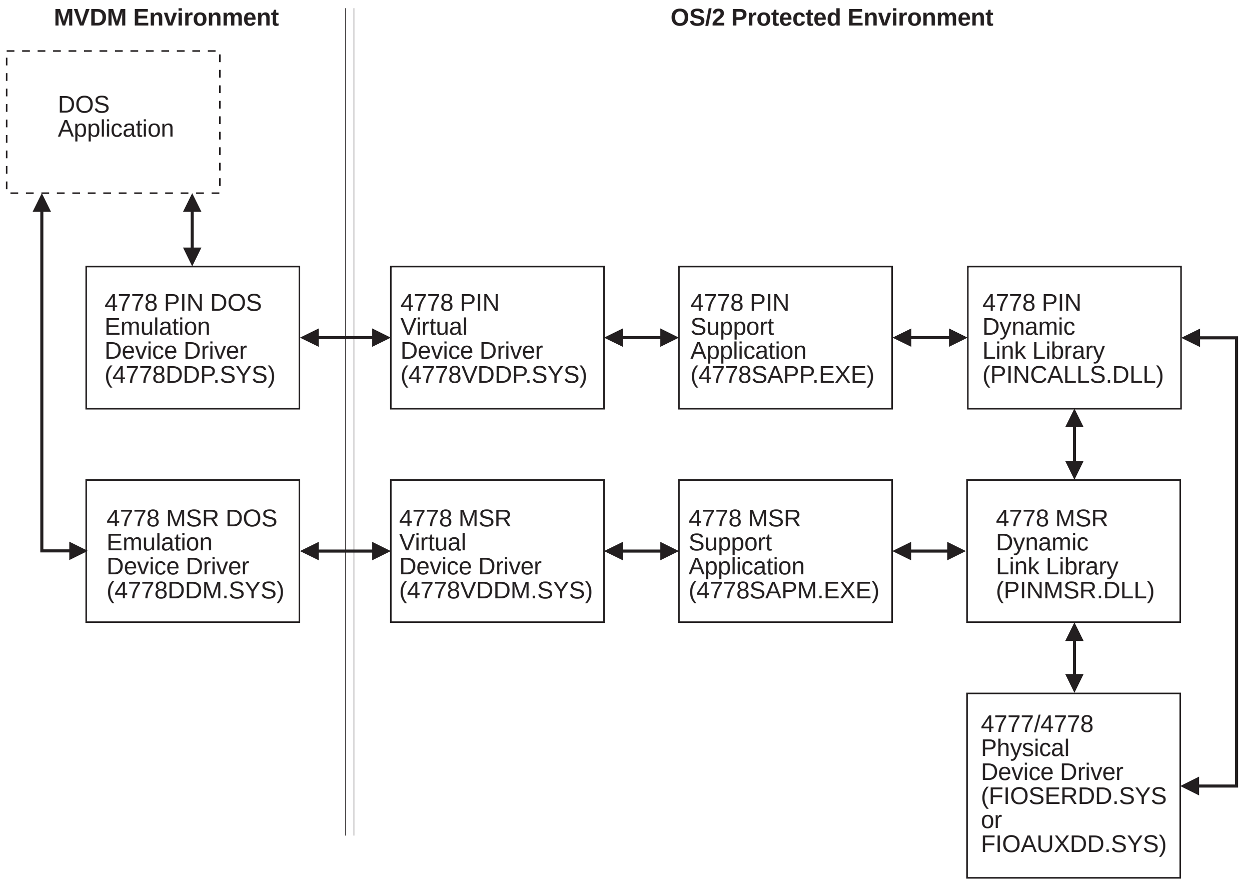
MVDM Environment
OS/2 Protected Environment
DOS Application
4778 PIN DOS Emulation Device Driver (4778DDP.SYS)
4778 PIN Virtual Device Driver (4778VDDP.SYS)
4778 PIN Support Application (4778SAPP.EXE)
4778 PIN Dynamic Link Library (PINCALLS.DLL)
4778 MSR DOS Emulation Device Driver (4778DDM.SYS)
4778 MSR Virtual Device Driver (4778VDDM.SYS)
4778 MSR Support Application (4778SAPM.EXE)
4778 MSR Dynamic Link Library (PINMSR.DLL)
4777/4778 Physical Device Driver (FIOSERDD.SYS or FIOAUXDD.SYS)

Figure 7-1. 4778 MVDM I/O System

Loading the 4778 MVDM Device Driver

You can load the MVDM device drivers using the FINSTALL program or you can install them manually. The automatic and manual procedures are described in the following sections.

Automatic Installation

To load and install the 4778 MVDM I/O system using FINSTALL:

1. Insert the OS/2 device-driver diskette into the A: drive.
2. Enter `A:FINSTALL` at the command prompt and press **ENTER**.
3. Follow the instructions on the panels displayed by the installation program.

   Note: If you use a virtual machine boot (VMB) DOS session, you must add the device-driver statements to the CONFIG.SYS file that is used to start the DOS session.

When you are finished, the program (FINSTALL) updates the files that contain the device-specific code and copies them to the drive you specified.

Manual Installation

This procedure loads and initializes the 4778 MVDM I/O system without using the FINSTALL program. In the CONFIG.SYS syntax, the disk-drive identifier and directory path are defined as:

d:	The disk-drive identifier
path	The directory path

Brackets [ ] indicate optional parameters.

1. Copy the following files from the device-driver diskette to your workstation hard disk:

   4778DDP.SYS
   4778DDM.SYS
   4778VDDP.SYS
   4778VDDM.SYS
   4778SAPP.EXE
   4778SAPM.EXE

2. Add the following statements to your CONFIG.SYS file. The device-driver statements must be in the CONFIG.SYS file in the root directory of the hard disk used to start the OS/2 session.

   a. For the 4778 DOS emulation device drivers:

   ```
   DEVICE=[d:][path]4778DDP.SYS [/X/Y/K:val/Z]

   DEVICE=[d:][path]4778DDM.SYS [/X/Y/K:val/Z]
   ```

   The DOS emulation device statement parameters are defined as follows:

   /X Suppresses the messages that are displayed if the 4778DDP or 4778DDM device driver detects any error conditions. Normally, the messages are displayed and the processing stops until you press any key. When using this option, the application program must handle all error messages.

/Y Prevents setting the error bit and returning error codes to DOS. All error status information is blocked. The application program must issue a READ STATUS request to verify the results of any operations.

/K:val
Assigns the Cancel operation to a key you select, where *val* is a 3-digit decimal value. Use the format `/K:val`, when *val* is the standard ASCII code for the selected key. If the key represents an extended ASCII code, use the format: `/K: ;val`.

The default setting assigns Cancel to the Esc key (ASCII code 27). This key assignment is valid only during synchronous operation.

/Z This option postpones any DOS OPEN error codes if the 4778 device is currently controlled by another MVDM session or by an OS/2 session. This lets you see the pop-up message, then go to the owning session and issue a Close function to release the 4778 device. Otherwise, the Open error is passed to the application.

b. For the 4778 virtual device drivers:

```
DEVICE=[d:][path]4778VDDP.SYS
DEVICE=[d:][path]4778VDDM.SYS
```

c. For the 4778 support applications:

```
RUN=[d:][path]4778SAPP.EXE
RUN=[d:][path]4778SAPM.EXE
```

3. Load and initialize the 4778 dynamic link library. For information about using the dynamic link library for the 4778 device, see “Loading the 4778 Dynamic Link Library” on page 2-3.
4. Load and initialize the 4778 physical device driver. For information about using the physical device driver for the 4778 device, see Chapter 2, “Loading and Initializing the Device Driver.”
5. Load the message files. For information about loading the message files, see “Loading the Message Files” on page 2-4.

Using the 4778 MVDM Device Driver

The 4778 device driver for OS/2 MVDM sessions provides:

- DOS-defined functions for the MVDM and VMB environments under the OS/2 operating system.
- Serial and interleaved sharing of the 4778 hardware between OS/2 protected-mode applications and DOS applications in the MVDM environment.
- Use of the 4778 MVDM I/O system by a single MVDM session. All subsequent MVDM sessions are prevented from using the 4778 MVDM I/O system.

DOS Application Program Interface

The 4778 MVDM I/O system uses the DOS interrupt 21H application program interface. The interrupt 21H interface is described in the *4778 DOS Programming Guide*. The interface is compatible with the DOS device driver, PIN2DD.SYS, with the following exceptions:

- The IOCTL Write subfunctions for reading and writing data in the TopView* environment are not supported.
- A pop-up message is displayed when the MVDM application tries to use the 4778 device and it is currently in use by an OS/2 session.

  You can clear this condition by switching to the OS/2 session and issuing a Close request. Then you can request a retry of the Open request or return the error to the application program.

  Note: The pop-up message is not visible during a DOS full-screen session although the workstation sounds a beep when the message is issued. You can view the message by bringing a windowed session to the foreground or by running the application in a DOS window.
- If the DOS application issues a DOS Close request to the 4778 MVDM interface when the 4778 device is enabled, the 4778 device indicator is turned off and the unit is disabled.

  This ensures that the 4778 is available in a reset state for any OS/2 protected-mode session application.
- If the application issues multiple open requests without corresponding closes in between, a DOS Open error is returned to the DOS application.
- If another MVDM session is already using the 4778 device, a DOS Open error is returned to the application program. This error causes either the FIO0593 or FIO0598 error message to appear.

The DOS emulation device driver (4778DDP.SYS) processes all 21H interrupts with the PINDEV handle name. The 4778DDM.SYS device driver processes all 21H interrupts with the PINMSR$ handle name.

OS/2 Protected Mode

The 4778 MVDM I/O system shares a common interface with the 4778 OS/2 protected-mode support programming (see Figure 7-1 on page 7-1). The OS/2 physical device driver determines which mode is in control of the device. When the MVDM environment is in control, Open requests from protected-mode applications receive an ERROR_NOT_READY return code from the OS/2 session.

Although the MVDM and OS/2 protected-mode applications can all run concurrently, only one session can *own* the device at any time. The ownership is on a first-use basis. That is, the first application that requests the device is given control and this determines which API is used. After an application releases control of the device, the I/O system is available to the first application that requests control.

* Trademark of IBM

The 4778 MVDM support programs use the dynamic link libraries to perform the actual I/O operations requested by the DOS application.

Avoiding Unwanted Initialization Error Messages

The 4778 DOS emulation device drivers will install by default in each DOS session when the session is started. The device drivers will attempt to communicate with the 4778 using the other I/O system components during initialization. If this is not possible, the device drivers will display a message. This message could be considered a nuisance, especially if you want to use the I/O system only in the first DOS session to be started up and you also want to start additional DOS sessions that do not use the I/O system.

To avoid this condition and maintain other DOS sessions that do not use the MVDM I/O system, you can:

- Use the /X option on the DEVICE statement. Use caution with this solution because all error messages are suppressed. This can result in other error conditions being hidden from the user.
- Create separate command prompts in the Command Prompts-Icon View to start DOS sessions with, and without, the device driver in the device-driver DOS settings. There can be a single with-device-driver DOS session started using the appropriate icon prompt.

Chapter 8. Messages and Status Codes

This chapter lists and describes the installation messages and program status codes for the 4777 and 4778.

Installation Messages

FIO0550 4777/4778 Device Interrupt Denied

Explanation: The device driver could not install the device interrupt handler. This is a critical error. The driver is installed but any I/O requests are denied.

User Response: Ensure that no other device driver is installed that might own the serial port exclusively.

FIO0551 4777/4778 CONFIG.SYS Specification Error

Explanation: The device driver detected an error in the optional parameters that are specified in the CONFIG.SYS file on the DEVICE=FIOSERDD.SYS or the DEVICE=FIOAUXDD.SYS command line. This error is not critical; the device driver installs in storage if this is the only error detected.

User Response: Correct the command line in the CONFIG.SYS file that loads the device driver. The valid parameters are /Cx (serial only), /P, /M, /W, and /S.

FIO0552 4777/4778 System Unit Model Not Supported

Explanation: The device driver attempted to install into an unsupported system unit. The 4777 and the 4778 will not install into a Personal Computer AT*, Personal Computer XT* (PC/XT*) 286, PS/2 Model 30, or PC Convertible system. This is a critical error; the device driver is installed but any I/O requests are denied.

User Response: Do not attempt to install the device driver into an unsupported system unit. See Chapter 2, “Loading and Initializing the Device Driver,” for a list of supported systems.

FIO0553 4777/4778 Timer Request Failure

Explanation: The device driver was not given access to the timer interrupt. This is a critical error. The driver is installed but any I/O requests are denied.

User Response: Reduce the number of active timers installed on your system.

* Trademark of IBM

FIO0555 Device driver denied access to serial port

Explanation: The device driver was not given access to the requested serial port. The serial port is in use or not installed.

User Response: Either change the COM port specification in CONFIG.SYS or run setup to add the specified COM port.

FIO0560 4777 Device Not Attached

Explanation: The device driver could not communicate with the device. Either a device is not attached or the device failed. This error is critical only when no device is attached. In this case the device driver is removed from storage.

User Response: Ensure that a device is attached. If a device is attached, either replace the unit or test the device with the customer diagnostic tests.

FIO0561 4777 Diagnostic Test Failure

Explanation: The device driver detected an error with the device; at least one of the power-on diagnostic tests in the device failed. This is a critical error; the device driver is installed but any I/O requests are denied.

User Response: Either replace the unit or test the device with the customer diagnostic tests.

FIO0562 4777 Communication Failure

Explanation: The device driver established initial communication with the device. However, a subsequent communication sequence with the device failed. This is a critical error; the device driver might not install in storage.

User Response: Test the device with the customer diagnostic tests. Replace the device if the communication errors continue.

FIO0570 4777 Device Is Currently Not Available

Explanation: The Open request from the MVDM session cannot be completed because an OS/2 protected-mode application is using the device.

User Response: Close or cancel the device operation in the OS/2 session.

FIO0571 CONFIG.SYS Specification Error

Explanation: One of the optional parameters for the `DEVICE=4777DD.SYS` or the `DEVICE=FIOSERDD.SYS` or the `DEVICE=FIOAUXDD.SYS` statement is not specified correctly.

User Response: Check the statement in the CONFIG.SYS file and correct the error.

FIO0572 4777 MVDM System Component Not Installed

Explanation: The 4777 DOS emulation device driver detected that one or more required system components are not installed or could not be located during startup.

User Response: Ensure that the required files were correctly loaded and that they are on a directory path that is accessible during startup.

FIO0573 4777 MVDM System Unusable

Explanation: The 4777 MVDM system is currently in use.
The 4777 MVDM support allows only one session at a time.

User Response: Use the 4777 MVDM system to close the session.

FIO0580 4778 PIN Device Not Attached

Explanation: The device driver could not communicate with the device. Either a device is not attached or the device failed. This error is critical only when no device is attached.

User Response: Ensure that a device is attached. If a device is attached, either replace the unit or test the device with the customer diagnostic tests.

FIO0581 4778 PIN Diagnostic Test Failure

Explanation: The device driver detected an error with the device; at least one of the power-on diagnostic tests in the device failed. This is a critical error; the device driver is installed but any I/O requests are denied.

User Response: Either replace the unit or test the device with the customer diagnostic tests.

FIO0582 4778 PIN Communication Failure

Explanation: The device driver established initial communication with the device. However, a subsequent communication sequence with the device failed. This is a critical error; the device driver might not install in storage.

User Response: Test the device with the customer diagnostic tests. Replace the device if the communication errors continue.

FIO0583 4778 MSR Device Not Attached

Explanation: The device driver could not communicate with the 4778 magnetic-stripe reader (MSR) component. Either the device is not attached or it has failed. This is a critical error only when no device is attached to the serial port. When no device is attached, the device driver is removed from storage.

User Response: Ensure that a 4778 unit is attached. If the device is attached, either replace the unit or test the device by using the customer diagnostic tests.

FIO0584 4778 MSR Diagnostic Test Failure

Explanation: The device driver detected an error with the device; at least one of the power-on diagnostic tests in the device failed. This is a critical error; the device driver is installed but any I/O requests are denied.

User Response: Test the unit with the customer diagnostic tests. If power-on diagnostic errors continue, replace the unit.

FIO0585 4778 MSR Communication Failure

Explanation: The device driver established initial communication with the device. However, a subsequent communication sequence with the MSR component failed. This is a critical error; the device driver is installed but any I/O requests are denied.

User Response: Test the device with the customer diagnostic tests. Replace the device if the communication errors continue.

FIO0590 4778 PIN Device Is Currently Not Available

Explanation: The Open request from the MVDM session cannot be completed because an OS/2 protected-mode application is using the 4778 PIN keypad device.

User Response: Close or cancel the device operation in the OS/2 session.

FIO0591 CONFIG.SYS Specification Error

Explanation: One of the optional parameters for the `DEVICE=4778DDP.SYS` or the `DEVICE=FIOSERDD.SYS` or the `DEVICE=FIOAUXDD.SYS` statement is not specified correctly.

User Response: Check the statement in the CONFIG.SYS file and correct the error.

FIO0592 4778 PIN MVDM System Component Not Installed

Explanation: The 4778 PIN DOS emulation device driver detected that one or more required system components are not installed or could not be located during the initial program load (IPL).

User Response: Ensure that the required files are correctly loaded and that they are on a directory path that is accessible during IPL.

FIO0593 4778 PIN MVDM System Unusable

Explanation: The 4778 PIN MVDM system is currently in use. The 4778 PIN MVDM support allows only one session at a time.

User Response: Use the 4778 MVDM system to close the session.

FIO0595 4778 MSR Device Is Currently Not Available

Explanation: The Open request from the MVDM session cannot be completed because an OS/2 protected-mode application is using the 4778 MSR device.

User Response: Close or cancel the device operation in the OS/2 session.

FIO0596 CONFIG.SYS Specification Error

Explanation: One of the optional parameters for the `DEVICE=4778DDM.SYS` or the `DEVICE=FIOSERDD.SYS` or the `DEVICE=FIOAUXDD.SYS` statement is not specified correctly.

User Response: Check the statement in the CONFIG.SYS file and correct the error.

FIO0597 4778 MSR MVDM System Component Not Installed

Explanation: The 4778 MSR DOS emulation device driver detected that one or more required system components are not installed or could not be located during the IPL.

User Response: Ensure that the required files were correctly loaded and that they are on a directory path that is accessible during IPL.

FIO0598 4778 MSR MVDM System Unusable

Explanation: The 4778 MSR MVDM system is currently in use. The 4778 MSR MVDM support allows only one session at a time.

User Response: Use the 4778 MVDM system to close the session.

Application Program Status Codes

This section describes the application program status codes for the 4778 PIN-Pad Magnetic Stripe Reader. When the requested function call completes, the OS/2 system returns the status codes to the application program in register AX.

For information about the standard OS/2 return codes that are not described here, use the OS/2 on-line help facility.

PIN Keypad Status Codes

The application program status codes for keypad functions are in decimal form as follows:

0 Pin_No_Error

Explanation: The function call completed successfully; the 4778 PIN keypad device-support code did not detect an error.

110 Device driver failed to open

Explanation: The device driver did not open; the function call did not complete.

1793 Pin_Invalid_Handle

Explanation: The PinHandle that was obtained from the last function call does not match the PinHandle that was returned from the PinOpen function call.

1794 **Pin_Device_Errors**

Explanation: The 4778 detected an error in the requested operation. This message results when one of the following occurs:

A check-sum error
Incorrect parity for the resident encryption key
A communication error
An EEPROM write error

1795 **Pin_Incorrect_Data_Length**

Explanation: The size allocation for the requested data does not match the size of the data that the application program received. This error message also appears when the data length is zero.

1796 **Pin_Invalid_Key_Pressed**

Explanation: During the PinEnterMasterKey function call, a key was pressed that is not valid.

1797 **Pin_Invalid_Mode**

Explanation: The PIN mode is not valid. The 4778 is in the nonencrypted mode and a PinCreateOffsetData function call was requested or the 4778 is in the encrypted mode and a PinReadClearData function call was requested.

1798 **Pin_Dll_Error**

Explanation: The 4778 device-support code detected an unrecoverable error and ended the current function call.

1799 **Pin_Operation_Aborted**

Explanation: The application program issued the PinAbort function call and ended the current I/O operation.

1800 **Pin_Data_Error**

Explanation: The application program detected an error in the parameters that are specified for the current function call, such as the following:

A variant is not valid.
The parity for a variant is not valid.
The data is not valid.
The data length is not valid.

1801 **Pin_Driver_Not_Open**

Explanation: An operation was requested and the 4778 device driver was not open.

1802 **Pin_Busy**

Explanation: The application program issued a function call but the 4778 is not available (another PinHandle owns the device).

1803	**Pin_Not_Verified**
	Explanation: The PIN entry was not verified for a PinVerifyPin or a PinVerifyMac function call.
1804	**Pin_Incorrect_Key_Parity**
	Explanation: The 4778 device-support code detected a key parity that is not valid for the requested function call.
1805	**Pin_Not_Usable**
	Explanation: For a PinOpen function call, this message can result when one of the following conditions occurs: An error occurred in the CONFIG.SYS file. The 4778 is not attached. The 4778 failed the device self-test.
1806	**Pin_LED_Data_Truncated**
	Explanation: The data string to be written to the 4778 display exceeds the 16-character maximum. The application program truncated the data string to 16 characters.

Magnetic-Stripe Operation Status Codes

This section describes the application program status codes that can be returned by magnetic-stripe function calls when you are using a 4778 Model 001. The status codes are in decimal form as follows:

0	**MAG OK**
	Explanation: The function call completed successfully; the 4778 device-support code did not detect an error.
110	**Device driver failed to open**
	Explanation: The device driver did not open; the function call did not complete.
1538	**Request rejected, no 4778 device attached**
	Explanation: The 4778 is not attached; the device-support code rejected the function call.
1539	**Request rejected, 4778 MSR device hardware problems**
	Explanation: An error occurred in the 4778 hardware; the 4778 device-support code rejected the function call.
1540	**MagHandle not valid for current session**
	Explanation: The MagHandle that was used in the last function call does not match the MagHandle that was returned from the PinMagOpen function call.

1541 **Device armed, cannot accept request**

Explanation: The 4778 is enabled for a read operation; the 4778 device-support code cannot process the function call. The 4778 device-support code permits a function call only when the 4778 is disabled and all the indicators are switched off.

1542 **Requested read capability does not exist**

Explanation: The 4778 does not support the track that is specified in the ReadTracks parameter of the PinMagReadData function call.

1545 **Track-1 parameter invalid**

Explanation: The 4778 device-support code rejected the PinMagLoadDevParms function call when it detected a track-1 parameter that is not valid.

1546 **Track-2 parameter invalid**

Explanation: The 4778 device-support code rejected the PinMagLoadDevParms function call when it detected a track-2 parameter that is not valid.

1548 **Application data buffer size too small for requested data**

Explanation: To transfer the requested data, the buffer allocation is insufficient. A function call with a zero-length buffer size generates this message.

1549 **Device self-test failed**

Explanation: The application program issued a PinMagResetDevice function call and the 4778 reported errors when it performed the self-test.

1550 **Operation aborted at request of application**

Explanation: The application program issued a PinMagAbort function call and ended a PinMagReadData function call.

1551 **Operation aborted by 4778 support code**

Explanation: The 4778 MSR device-support code detected an unrecoverable error and ended the current function call.

1552 **Magnetic stripe read error**

Explanation: The application program requested the device not to be re-armed following a read error from the magnetic stripe. ended the current function call.

Index

Numerics

3604 Keyboard Display 1-2
3624 PINs, verifying 5-7
3624-formatted PIN block 5-7
3624-formatted PIN block, creating 4-32
4700 Finance Communication System 1-1
4700 Financial I/O devices 1-1
4704 EPP data format 5-5
4704 Keyboard Display 1-2
4704-formatted PIN block, creating 4-31
4777
- function calls
  - MagAbort 3-9
  - MagClose 3-3
  - MagEncodeData 3-8
  - MagLoadDevParms 3-4
  - MagOpen 3-2
  - MagReadConfigStatus 3-12
  - MagReadData 3-13
  - MagReadDevParms 3-10
  - MagReadErrorStats 3-11
  - MagResetDevice 3-9
  - MagSetOperationMode 3-7
- hardware and software requirements 2-1
- hardware description 1-1
- installation messages 8-1
- physical device driver 2-1
- status messages 8-7

4777 DLL (dynamic link library) 2-3
4777 dynamic link library (DLL), loading 2-3
4778
- API functions 4-1
- cryptographic keys 5-9
- diagnostic test 4-45
- encrypted mode 4-23
- hardware and software requirements 2-1
- input data formats 5-1
- installation messages 8-1
- magnetic-stripe reader API 4-2
- multiple-virtual-DOS-machine (MVDM) API 7-1
- MVDM device driver 7-2
- nonencrypted mode 4-22
- physical device driver 2-1
- PIN keypad API 4-1
- self-test 4-17
- status messages 8-5
- user-specified operation 5-3
- using the device-support code 4-1

4778 DLL (dynamic link library) 2-3
4778 dynamic link library (DLL) 2-3
4778 PIN keypad
- ending I/O operation 4-7
- master key 4-24
- PIN verification 4-36

4778DDM.SYS 7-1
4778DDP.SYS 7-1
4778SAPM.EXE 7-1
4778SAPP.EXE 7-1
4778VDDM.SYS 7-1
4778VDDP.SYS 7-1

A

AEOM (alternate end-of-message) character 3-5, 4-14
alternate end-of-message (AEOM) character 3-5, 4-14
alternate start-of-message (ASOM) character 3-5, 4-14
American National Standards Institute (ANSI) 1-2
ANSI (American National Standards Institute) 1-2
ANSI X9.8 data format 5-6
ANSI X9.8-formatted PIN block, creating 4-34
API
- *See* application program interface (API)

application program interface (API)
- 4777 3-1, 3-2
- 4778 4-1
- DOS interrupt 21H 7-4
- multiple-virtual-DOS-machine I/O system 6-1, 7-1

application program status messages
- 4778 Model 001 8-7
- 4778 PIN keypad 8-5

ASCII translation table (nonencrypted PIN data) 5-4
ASOM (alternate start-of-message) character 3-5, 4-14

C

closing the 4777 3-3
closing the MSR device 4-6
closing the PIN-pad device 4-5
commands, 4778
- Create PIN Block 5-11
- Enter Master Key 5-12
- Generate MAC 5-11
- Load ICV 5-11
- Load Key 5-11
- Verify MAC 5-11
- Verify PIN 5-11

CONFIG.SYS
- 4777 6-2
- determining 4778 configuration status 4-9
- determining MSR configuration status 4-11
- loading the MDVM device driver 7-2

CONFIG.SYS file, device statements for
    4778   2-2
configuration data
    reading status   3-12
conversion (hexadecimal-to-keystroke) table   5-13
Create PIN Block command   5-5, 5-11
cryptographic key management
    describing   5-9
    key variants   5-10
    loading keys   5-9
    triple-encrypted keys   5-10

D

data formats
    3624 PIN   5-7
    4704 EPP   5-5
    ANSI X9.8   5-6
    converting the master key   5-12
    describing   5-1
    encrypted PIN data   5-5
    keypad-entry   5-12
    magnetic-stripe data   5-1
    multitrack read operation   5-2
    nonencrypted PIN data   5-4
    single-track read operation   5-2
determining MSR configuration status   4-11
determining PIN-pad configuration status   4-9
device drivers
    4778 MSR DOS emulation   7-1
    4778 MSR virtual   7-1
    4778 physical   2-1, 7-1
    4778 PIN DOS emulation   7-1
    4778 PIN virtual   7-1
    4778 virtual   7-3
    loading manually   2-2, 7-2
    loading with the FINSTALL program   2-1, 7-2
    magnetic-stripe data   5-1
    MVDM
        automatic installation   6-2
        manual installation   6-2
device serial numbers, reading   4-47
device-support code
    4777   3-1
    using   3-2
device-track parameters
    loading   3-4
    reading   3-10
device-track parameters, read operation   5-3
diagnostic test   4-45
displaying data   4-49
DOS application program interface   7-4
double-track read-data format   5-1

E

encoding magnetic data   3-8
encrypted mode (PIN keypad)   4-23
encrypted PIN data format   5-5
encryption key, loading the   4-28
end-of-message (EOM) character   3-5
ending a PIN-pad I/O operation   4-7
ending an MSR active state   4-8
Enter Master Key command   5-12
entering the PIN-keypad master key manually   4-24
EOM (end-of-message) character   3-5
error statistics   3-11
error statistics, reading   4-19

F

files
    CONFIG.SYS   7-2
    message   2-4
FIOSERDD.SYS   7-1
function calls
    MagAbort   3-9
    MagClose   3-3
    MagEncodeData   3-8
    MagLoadDevParms   3-4
    MagOpen   3-2
    MagReadConfigStatus   3-12
    MagReadData   3-13
    MagReadDevParms   3-10
    MagReadErrorStats   3-11
    MagResetDevice   3-9
    MagSetOperationMode   3-7
    MVDM I/O system   7-1
    PinAbort   4-7
    PinClose   4-5
    PinCreateOffsetData   4-38
    PinEnterMasterKey   4-24
    PinExecDevDiag   4-45
    PinGenerateMac   4-40
    PinLoadICV   4-29
    PinLoadKey   4-28
    PinLoadMasterKey   4-26
    PinLoadVerifParms   4-30
    PinMagAbort   4-8
    PinMagClose   4-6
    PinMagLoadDevParms   4-13
    PinMagOpen   4-4
    PinMagReadConfigStatus   4-11
    PinMagReadData   4-20
    PinMagReadDevParms   4-18
    PinMagReadErrorStats   4-19
    PinMagResetDevice   4-17
    PinMagSetOperationMode   4-16
    PinOpen   4-3
    PinReadClearData   4-48

function calls *(continued)*
PinReadConfigStatus 4-9
PinReadPin3624 4-32
PinReadPin4704 4-31
PinReadPinAnsi98 4-34
PinReadSN 4-47
PinSetModeClr 4-22
PinSetModeEnc 4-23
PinVerifyMac 4-43
PinVerifyPin 4-36
PinWriteDisplay 4-49

G
Generate MAC command 5-11

H
hardware requirements 2-1
hexadecimal-to-keystroke conversion table 5-13

I
initial-chaining value (ICV), loading 4-29
installation
4777/78 physical device driver 2-1
4778 MVDM Device Driver 7-2
CONFIG.SYS file 7-2
messages, generating 2-4
installation messages
4777 and 4778 8-1
generating 2-4
International Standards Organization (ISO) 1-2
interrecord zero (IRZ) characters 3-5
IRZ (interrecord zero) characters 3-5
ISO (International Standards Organization) 1-2

K
key variant, definition 5-10
keypad-entry data format 5-12
keys
master type parameter 4-24
master, manual entry 4-24
requirement to reload 4-23, 5-9
triple-encrypted 5-10

L
Load ICV command 5-11
Load Key command 5-11
loading the device drivers
4777 MVDM I/O system
automatically 6-2
manually 6-2
4778 2-1
automatically 2-1
loading the device drivers *(continued)*
manually 2-2
loading the device parameters 3-4
loading the MSR device parameters 4-13
loading the PIN keypad master key 4-26
longitudinal redundancy check (LRC) 3-4, 4-13
LRC (longitudinal redundancy check) 3-4, 4-13

M
MAC (message authentication code)
generating 4-40
using 5-14
verifying 4-43
MagAbort call 3-9
MagClose call 3-3
MagEncodeData call 3-8
MagLoadDevParms call 3-4
magnetic-stripe data, reading 4-20
magnetic-stripe reader (MSR)
closing 4-6
ending active state 4-8
error statistics 4-19
magnetic-stripe data 4-20
opening 4-4
operational mode 4-16
read capabilities 4-11
magnetic-stripe reader API 4-2
magnetic-stripe unit
hardware description 1-1
models 1-2
magnetic-stripe-reader function calls
PinAbort 4-7
PinClose 4-5
PinCreateOffsetData 4-38
PinEnterMasterKey 4-24
PinGenerateMac 4-40
PinLoadICV 4-29
PinLoadKey 4-28
PinLoadMasterKey 4-26
PinLoadVerifParms 4-30
PinOpen 4-3
PinReadClearData 4-48
PinReadConfigStatus 4-9
PinReadPin3624 4-32
PinReadPin4704 4-31
PinReadPinAnsi98 4-34
PinReadSN 4-47
PinSetModeClr 4-22
PinSetModeEnc 4-23
PinVerifyMac 4-43
PinVerifyPin 4-36
PinWriteDisplay 4-49
MagOpen 3-4
MagOpen call 3-2

MagReadConfigStatus call 3-12
MagReadData call 3-13
MagReadDevParms call 3-10
MagReadErrorStats call 3-11
MagResetDevice call 3-9
MagSetOperationMode call 3-7
master key
  converting to keystrokes 5-12
  loading 4-26, 5-9
  manual entry 4-24
message authentication code (MAC)
  generating 4-40
  using 5-14
  verifying 4-43
message files 2-4
multiple-virtual-DOS-machine I/O system
  4777 device drivers 6-1
  4778 device drivers 7-1
  description 6-1
  DOS application program interface 7-4
  list of system components 7-1
multitrack read-data format 5-2
multitrack-read operation 3-7
MVDM (multiple virtual DOS machine)
  See multiple-virtual-DOS-machine I/O system

N
nonencrypted data 4-48
nonencrypted mode (PIN Keypad) 4-22
nonencrypted PIN data format 5-4
nonvolatile storage 4-29

O
offset data, creating 4-38
opening the 4777 3-2
opening the 4778 MSR device 4-4
opening the 4778 PIN-pad device 4-3
override for device-track default parameters 5-3

P
parameters
  device-track (read operation) 5-3
  overriding device-track defaults 5-3
  PIN block verification 4-30
PEOM (primary end-of-message) character 3-5, 4-14
Personal System/2 (PS/2) workstation 1-1
PIN block
  3624-formatted 4-32, 5-7
  4704-formatted 4-31
  ANSI X9.8-formatted 4-34, 5-6
PIN block verification parameters 4-30
PIN formats, verifying 5-7
PIN keypad API 4-1
PIN keypad function calls
  PinMagAbort 4-8
  PinMagClose 4-6
  PinMagLoadDevParms 4-13
  PinMagOpen 4-4
  PinMagReadConfigStatus 4-11
  PinMagReadData 4-20
  PinMagReadDevParms 4-18
  PinMagReadErrorStats 4-19
  PinMagResetDevice 4-17
  PinMagSetOperationMode 4-16
PinAbort call 4-7
PINCALLS.DLL 7-1
PinClose call 4-5
PinCreateOffsetData call 4-38
PinEnterMasterKey call 4-24
PinExecDevDiag 4-45
PinGenerateMac call 4-40
PinLoadICV call 4-29
PinLoadKey call 4-28
PinLoadMasterKey call 4-26
PinLoadVerifParms call 4-30
PinMagAbort call 4-8
PinMagClose call 4-6
PinMagLoadDevParms call 4-13
PinMagOpen 4-4
PinMagReadConfigStatus call 4-11
PinMagReadData call 4-20
PinMagReadDevParms 4-18
PinMagReadErrorStats call 4-19
PinMagResetDevice call 4-17
PinMagSetOperationMode call 4-16
PINMSR.DLL 7-1
PinOpen call 4-3
PinReadClearData call 4-48
PinReadConfigStatus call 4-9
PinReadPin3624 call 4-32
PinReadPin4704 call 4-31
PinReadPinAnsi98 call 4-34
PinReadSN call 4-47
PinSetModeClr call 4-22
PinSetModeEnc call 4-23
PinVerifyMac call 4-43
PinVerifyPin call 4-36
PinWriteDisplay call 4-49
primary end-of-message (PEOM) character 3-5, 4-14
primary start-of-message (PSOM) character 3-5, 4-13
PS/2 (Personal System/2) workstation 1-1
PSOM (primary start-of-message) character 3-5, 4-13

R
read operation 5-3
reading
  device parameters 3-10

reading *(continued)*
  error statistics 3-11
  magnetic data 3-13
reading the MSR device parameters 4-18
resetting the 4777 3-9
resetting the MSR device 4-17

S
self-test 4-17
setting the multitrack-read operation mode 4-16
single-track read-data format 5-1
software requirements 2-1
SOM (start-of-message) character 5-2
start-of-message (SOM) character 5-2
status codes
  magnetic-stripe reader 8-7
  PIN keypad 8-5
stopping an I/O operation 3-9
storage 4-29

T
tables
  ASCII translation (nonencrypted PIN data) 5-4
  hexadecimal-to-keystroke conversion 5-13
  variant descriptor bytes 5-11
triple-encrypted keys 5-10

U
Up arrow indicator, read operation 5-4
user-specified operation 5-3

V
variant-definition bytes (VDB) 5-10
VDB (variant-definition bytes) 5-10
verification algorithm 5-7
verification, PIN blocks 4-30
Verify MAC command 5-11
Verify PIN command 5-11
verifying the PIN block 4-36
virtual-machine-boot DOS session 7-2

W
workstation 1-1

Communicating Your Comments to IBM

4777 Magnetic Stripe Unit and
4778 PIN-Pad Magnetic Stripe Reader
OS/2 Programming Guide

Publication No. SA34-2205-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - United States & Canada: 1-800-955-5259

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

4777 Magnetic Stripe Unit and
4778 PIN-Pad Magnetic Stripe Reader
OS/2 Programming Guide

Publication No. SA34-2205-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction					

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate					
Complete					
Easy to find					
Easy to understand					
Well organized					
Applicable to your tasks					

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? Yes No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SA34-2205-00

IBM

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
RDS Solutions Development
Department 56I
8501 IBM Drive
Charlotte NC 28262-8563

Fold and Tape **Please do not staple** Fold and Tape

SA34-2205-00

Cut or Fold
Along Line

IBM

Part Number: 07H5084

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

7H5 84

SA34-22 5-