4779 Hybrid Smart Card Device

IBM

Programming Guide

SA34-2360-01

4779 Hybrid Smart Card Device

IBM

Programming Guide

SA34-2360-01

Note!

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

Second Edition (March 1997)

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 56I, 8501 IBM Drive, 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

Figures

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectable rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

Licensees of this program who wish to have information about it for the purpose of enabling:

(i) the exchange of information between independently created programs and other programs (including this one) and
(ii) the mutual use of the information which has been exchanged, should contact: IBM Corporation, Department MG39/201, 8501 IBM Drive, Charlotte, NC 28262-8563, U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY, 10594, USA.

Licensed Innovatron patents may apply to products that are described in this document.

Federal Communications Commission (FCC) Statement

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Properly shielded and grounded cables and connectors must be used in order to meet FCC emission limits. IBM is not responsible for any radio or television interference caused by using other than recommended cables and connectors or by unauthorized changes or modifications to this equipment. Unauthorized changes or modifications could void the user's authority to operate the equipment.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

License

You may use the files which make up Feature Code 3921 (Programs) with the IBM 4779 only in accordance with the IBM System Programs License Agreement that accompanies the Programs.

Trademarks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

Personal Security	Personal System/2	PS/2
IBM	Operating System/2	OS/2

About This Publication

This book tells you how to utilize the IBM* 4779 device in a Disk Operating System (DOS) environment or an OS/2* environment. It explains how to write application programs which communicate with serial-attached 4779 devices using the 4779 DOS and OS/2 device support code.

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 4779.

In order to use this document, the reader must be familiar with background material related to devices of this type. Specifically, the reader should have a working knowledge of the following areas:

Basic cryptography, including the use of DES and RSA

Magnetic stripe cards and their use

Smart cards

How This Book Is Organized

This book contains the following sections:

Chapter 1, “Introducing the 4779 Hybrid Smart Card Device,” describes the capabilities and programming requirements of the 4779 device.

Chapter 2, “DOS Device Driver,” describes how to install and use the 4779 DOS device driver.

Chapter 3, “OS/2 Device Driver and Support Code,” describes how to install and use the 4779 OS/2 device support code.

Chapter 4, “4779 Device Resident Application Function Definitions,” describes the commands provided by the 4779 Device Resident Application Program which may be invoked by PC application programs to perform 4779 non-security functions.

Chapter 5, “4779 Security Functions,” describes the security capabilities of the 4779 and identifies the security functions that a 4779 device resident application program or PC host application program may invoke.

Chapter 6, “4779 Support Programs,” describes the 4779 Support Programs which may be used to perform such functions as downloading keys into the device, reading device information and status, etc.

Appendix A, “4779 DOS and OS/2 Device Driver Diskette Components,” lists the components of the 4779 Device Support Code that may be found on the 4779 Device Support Code Diskette.

* Trademark of IBM

Appendix B, “4779 Device Return Codes,” describes the return codes issued by the 4779 through the IBM default 4779 Device Resident Application Program and Security Functions.

Appendix E, “CCA Verification Pattern Algorithm,” presents a diagram of the algorithm used to compute key verification patterns.

Appendix F, “Machine Information Data Structure,” describes the Machine Information Data Structure (MIDS), which contains information describing the hardware and BIOS features of the 4779 device.

Appendix G, “Character Font Table,” specifies the character font table that is displayable on the 4779 LCD.

Related Publications

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

The *DOS Technical Reference*, for your DOS operating system

OS/2 Programming Guide, SA34-2194

References

The following references may be useful to those readers who are not familiar with cryptography, magnetic stripe technology, or PIN pads.

Transaction Security System, Programming Reference: Volume I, Access Controls and DES Cryptography, IBM publication number SC31-2934.

Applied Cryptograhpy by Bruce Schneier, ISBN 0-471-59756-2.

Security for Computer Networks by D.W. Davies and W.L. Price, ISBN 0-471-92137-8.

Cryptography and Data Security by Dorothy Denning, ISBN 0-201-10150-5.

Transaction Security System, Concepts and Programming Guide: Volume I, Access Controls and DES Cryptography, IBM publication number GC31-3937.

Transaction Security System, Concepts and Programming Guide: Volume II, Public-Key Cryptography, IBM publication number GC31-2889.

4777 Magnetic Stripe Unit and 4778 PIN-Pad Magnetic Stripe Reader: OS/2 Programming Guide, IBM publication number SA34-2205.

4777 Magnetic Stripe Unit and 4778 PIN-Pad Magnetic Stripe Reader: DOS Programming Guide, IBM publication number SA34-2206.

ISO 7811-2: Identification cards - Recording technique - Part 2: Magnetic stripe..

ISO 7816-1: Identification cards - Integrated circuit cards with contacts - Part 1: Physical characteristics.

ISO 7816-2: Identification cards - Integrated circuit cards with contacts - Part 2: Dimensions and location of the contacts.

ISO 7816-3: Identification cards - Integrated circuit cards with contacts - Part 3: Electronic signals and transmission protocols.

Chapter 1. Introducing the 4779 Hybrid Smart Card Device

Smart Card Applications

Financial institutions, retailers and payment systems companies worldwide are moving rapidly towards smart card technology. They are embedding a microprocessor chip in the familiar plastic credit, debit and bank cards which in many cases are replacing, or coexisting with, the magnetic stripe card. Simultaneously, new applications are being developed. Of particular interest today is the *electronic purse* smart card which adds value into the chip, debits from the client's bank account through an ATM or other cash transfer vehicle, and is used to make purchases in place of the use of actual cash.

The requirement for this new technology is to make the transition from the magnetic stripe to the smart card with as little disruption to the client or financial institution user as possible.

The IBM 4779 Hybrid Smart Card Device is a multipurpose card reader/writer device that is flexible enough to cater to future expansion and additional functionality of smart cards. It provides for both banking needs and related payment systems' needs where synergy can be found for future demands. This device combines IBM's experience and expertise in pinpads, magnetic stripe technology, magnetic stripe readers, smart card readers, smart cards, and DES/RSA security into a single product. The IBM 4779 device provides a migration path from existing applications to new applications, ensuring the financial institution or retailer a low cost implementation and protection of his investment for the future.

* Trademark of IBM

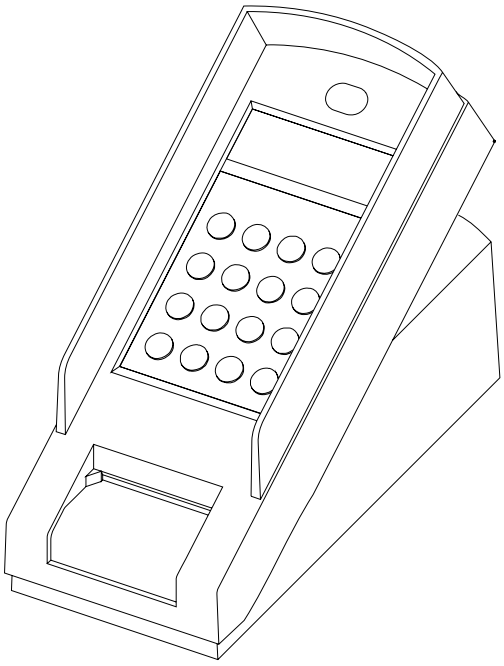


Figure 1-1. 4779 Hybrid Smart Card Device

General Description

The IBM 4779 is available in two models, the 4779-1 (*Model 1*) and the 4779-2 (*Model 2*). The 4779-1 is a motorized hybrid smart card reader/writer and magnetic stripe card reader-only table top device which connects to a PC by way of the serial port. The 4779-1 reads from and writes to a smart card and reads the three standard magnetic stripes on a magnetic card through the same card insertion slot. The card is automatically driven into the slot and seated. The 4779-1 consists of a display, pinpad and function keys and incorporates a programmable microprocessor. DES security within the device may be invoked by host applications to secure transactions with the device. The 4779-2, in addition to the characteristics and functionality of the 4779-1, includes support for encoding magnetic stripe cards. Both the 4779-1 and 4779-2 may be ordered with the Enhanced Security Feature which provides RSA security for key management. A complete description of the Enhanced Security Feature may be found in Chapter 5, “4779 Security Functions” on page 5-1.

Functionality

The IBM 4779 is designed for smart card and magnetic stripe card applications where either card or a card with both technologies might be used. The 4779 can be used in a financial or retail environment as a self-service device or operated by banking or retail personnel. Notable characteristics of 4779 include:

A Single Motorized Card Insertion Slot
Reading From and Writing to a Smart Chip Card
Reading Track 1, 2 and 3 of a Magnetic Stripe Card
Encoding Track 2 of a Magnetic Stripe Card (4779-2 only)
80 Character Liquid Crystal Display
- 20 Characters by 4 Lines

16 Key Keypad
- 10 Numeric Keys
- 2 Special Keys (* and #)
- 4 Programmable Function Keys

Security
- DES algorithm for PIN Verification
- RSA algorithm for Key Management (Enhanced Security Feature only)

Operating System Support
- DOS
- OS/2

4779 Software Components

The following software components are required in order to use the 4779 device.

The 4779 DOS Device Driver is required if the device is attached to a workstation running DOS. The 4779 OS/2 device support code is required if the device is attached to a workstation running OS/2. Both of these software components are provided by IBM on the 4779 DOS and OS/2 Device Drivers Diskette.

An application that resides in the PC system unit uses a 4779 device driver to communicate with the 4779 device. You will need to write this application. The 4779 DOS and OS/2 Device Driver Diskette contains sample files that you can use as a model for this application. Refer to Appendix A, “4779 DOS and OS/2 Device Driver Diskette Components” on page A-1 for a complete list of the components of the 4779 DOS and OS/2 Device Driver Diskette.

An application is device resident and calls the Basic Input/Output System (BIOS) of the 4779 device to control the operations of the device. This application is known as the device resident application program. IBM provides a pre-installed device resident application with each 4779 device. The IBM default device resident application can be modified using 4779 feature 3922, the *4779 Device Resident Application Program Development Kit*.

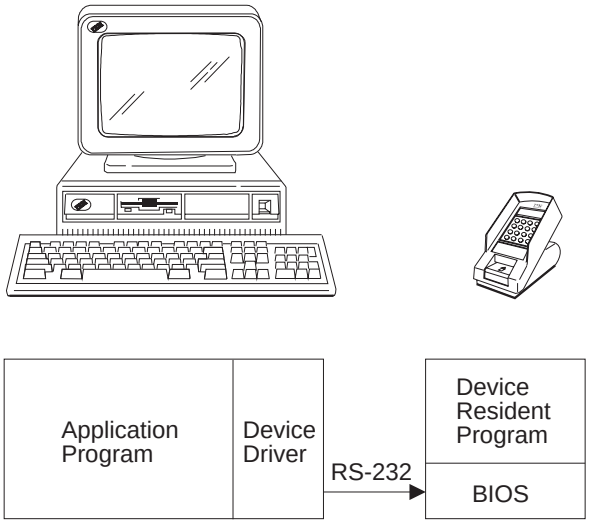


Figure 1-2. 4779 Application Programming Overview

Programmable Micro-Processor

You can design your application to execute partially in the host PC and partially in the 4779 microprocessor. This permits you to split the application function in a way that is most efficient for the overall application. For example, information can be read from a card, combined with device data, encrypted, and transmitted to the PC host application, all with a single function call from the PC application to the 4779 device. The portion of the application that resides in the 4779 programmable micro-processor is known as the device resident application and may be written in C or Assembler. IBM provides with each 4779 unit a default device resident program that has been pre-installed. Chapter 4, “4779 Device Resident Application Function Definitions” on page 4-1 describes the IBM default 4779 device resident program and its application interface.

Software Support

4779 Device Resident Application Program Development Kit, 4779 feature 3922, includes the code for the IBM default device resident application, a run-time library of 4779 high level I/O interface support, and a 4779 Device Resident Program Programming Guide. You may use these items to customize the 4779 default device resident application program or to create your own 4779 device resident application program.

Contact your local IBM marketing representative for more information about the 4779 Device Resident Application Software Development Kit.

Application Changes

Device resident application changes can be downloaded to the programmable micro-processor.

Security

Both models of the 4779 device include DES security. The optional Enhanced Security Feature adds RSA security for key management. The 4779 has a battery backup to maintain the security keys when the system is powered off. In addition, the Enhanced Security Feature provides additional intrusion security. When an intrusion is detected the security keys are immediately erased.

Display

The 4779 LCD display is covered by an optical grade clear window integrated into the main cover such that it will prevent the entry of dust, liquids and static electricity into the device.

Privacy Screen

A privacy screen is integrated into the terminal to provide user PIN entry protection from casual observers.

Card Acceptor

The card acceptor is motorized providing ease of use and optimum magnetic performance. The motor is activated by the presence of a user's card inserted into the entry slot.

Communications Integrity

A watchdog timer circuit is provided to ensure that communications can be regained in case of a communications processing error.

Packaging

The terminal rests on a counter or desk top secured to a surface with a removable base plate. The base plate is an integral part of the terminal. The terminal can be quickly and easily removed from the base plate for servicing without the use of tools.

Product Specifications

The technical specifications of the 4779 are summarized as follows:

Magnetic Stripe Cards: Tracks 1, 2 and 3
Processor ICC: All Cards Complying With ISO 7816
Protocol: ISO T=0, T=1
ICC Contact Method: Landing Contacts
Interface to Controller: RS-232C
Transaction Rate to Controller: 9600 Baud
Keypad: Rubberdome
Power Supply: Wall Mounted Power Transformer (AC to DC Converter) Through a Signal Cable
- Low Voltage Input: 90-132V
- High Voltage Input: 180-264V

Physical Dimensions:

Figure 1-3. Physical Dimensions

	4779-001	**4779-002**
Width	105mm (4.13 inches)	105mm (4.13 inches)
Depth	220mm (8.70 inches)	265mm (10.43 inches)
Height	170mm (6.70 inches)	200mm (7.87 inches)
Weight	1.55kg (3.4 lbs.)	1.55kg (3.4 lbs.)

Compatibilities and Restrictions

The 4779 device may coexist with the IBM 4777 Magnetic Stripe Reader/Encoder.

Please note the following restrictions:

The 4779 cannot be installed on the same system with a 4778 device. Although they are similar in operation, the device drivers for the 4778 devices are not compatible with the 4779 device.

Two 4779 devices cannot be attached to a common system.

The 4779 is not compatible with the IBM Personal Security Card.

ISO Standards Compliance

The 4779 device conforms to the following standards:

ISO/IEC 4909: 1987 Bank Cards - Magnetic Stripe Data Content for Track 3
ISO/IEC 7810: 1993 Identification Cards - Physical Characteristics
ISO/IEC 7811-Part 2: 1993 Identification Cards - Recording Technique: Magnetic Stripe
ISO/IEC 7811-Part 4: 1993 Identification Cards - Recording Technique: Location of Read-Only Magnetic Tracks 1 and 2
ISO/IEC 7811-Part 5: 1993 Identification Cards - Recording Technique: Location of Read-Only Magnetic Tracks 3
ISO/IEC 7816-Part 1: 1987 Identification Cards - Integrated Circuit(s) Cards with Contacts: Physical Characteristics
ISO/IEC 7816-Part 2: 1988 Identification Cards - Integrated Circuit(s) Cards with Contacts: Dimensions and Location of the Contacts
ISO/IEC 7816-Part 3: 1989 Identification Cards - Integrated Circuit(s) Cards with Contacts: Electronic Signals and Transmission Protocols
ISO 9992-Part 1: 1990 Financial Transaction Cards - Messages Between the Integrated Circuits Card and the Card Accepting Device: Concepts and Structures
ISO/IEC 10 373: 1993 Identification Cards - Testing Methods

Chapter 2. DOS Device Driver

DOS device support for the 4779 device consists of a single installable device driver. This device driver has the required logic to coexist with any model of the 4777 MSR/E device family.

Installation

The 4779 device driver is installed via the CONFIG.SYS and supports a single 4779 device attached to one of the COM ports as follows:

DEVICE=*[d:][path]***4779DOS.SYS** *[options]*

where

d: is the disk containing the 4779DOS.SYS file

path is the fully qualified path to the 4779DOS.SYS file

options are the optional parameters that configure the device driver as follows:

/W waits for operator intervention if an error is detected at initialization. The default is no wait.

/C*x* **(Do not select options /Q or /A)** designates the COM port to which the 4779 is attached. *x* may be any value 1 - 4, representing COM1 - COM4. The default is COM1. If this parameter is specified, it overrides the base COM address specified by the /A parameter and the interrupt level specified by parameter /Q parameter.

/A*xxxx* **(also select /Q)** associates the 4779 device with a particular RS232 base port address. *xxxx* specifies the four-digit hexadecimal RS232 base port address. If this parameter is specified, the /Q parameter must be specified as well. If the /C parameter is specified, the /A and /Q parameters are not used.

/Q*xx* **(also select /A)** associates the 4779 device with a particular interrupt request level (IRQ). *xx* specifies the two-digit decimal IRQ level. If this parameter is specified, the /A parameter must be specified as well. If the /C parameter is specified, the /A and /Q parameters are not used.

/Y suppresses the return of all error statuses from the 4779 device driver to DOS. In addition, the carry flag is not set. If this parameter is not specified and an error is detected in a function call or the operation is cancelled, an error bit and an error code are returned to DOS and processed by the INT 24H critical error handler. If you do

not have an error handler, the DOS Abort, Retry, Ignore message appears on the screen.

/T enables the built-in trace capability of the device driver. The default is trace is disabled.

Note: The optional parameters can be specified in any order, with or without spaces, and are not case sensitive. Additionally, the options can be specified with a **/** delimiter or with a **-** delimitter.

Note: The device driver communicates with the 4779 device using line settings of 9600 baud, odd parity, 8 bits per character, and 1 stop bit.

DOS Application Programming Interface

The 4779 DOS API consists of 4 *abstract* functions as follows:

Open Open a handle to the 4779 device
Close Close a handle to the 4779 device
Read Read a user defined response from the 4779 device
Write Write a user defined data block to the 4779 device

You may invoke these abstract functions by issuing the corresponding DOS system call or the appropriate function call for your programming language. The following table identifies the appropriate DOS system calls and IBM C/2 function calls. Other languages may also be used.

Figure 2-1. 4779 DOS API

Abstract Function	DOS System Call	IBM C/2 Function Call
Open	3DH - Open a File	open
Close	3EH - Close a File Handle	close
Read	3FH - Read from a File or Device	read
Write	40H - Write to a File or Device	write

The following paragraphs describe the abstract function calls. Return codes depend upon your implementation method.

Open

The open function call is used by an application to get a handle to the device driver for subsequent I/O functions. The driver processes open function calls and flushes the receive buffer queue when the number of open handles is zero. The 4779 device driver name is **IBMEFT$**. For additional information, see the *DOS Technical Reference*, function call 3DH. Prior to issuing this command, set the driver to binary mode by issuing the DOS function call *I/O Control for Devices* (44H, AL=01H).

Close

The close function call is used by an application to close a handle to the device driver when no further I/O is to be performed. The driver processes close function calls and flushes the receive buffer queue when the number of open handles is zero. For additional information, see the *DOS Technical Reference*, function call 3EH.

Read

The read function call is used by an application to read data from the device driver's receive queue. The driver reads and buffers data from the device and holds that data until an application reads it from the drivers buffer. For additional information, see the *DOS Technical Reference*, function call 3FH.

Since the driver buffers incoming data for an application, the application must be aware that overrun conditions can occur within the driver and must ensure that it reads data from the driver in a timely fashion. Additionally, an application must ensure that it stays in sync with the device and that responses from the device are kept in order with the command causing the response.

Write

The write function call is used by an application to write user defined data (commands) to the 4779 device. Since the data is user defined, the driver has no knowledge of the datastream and simply transmits the data to the 4779 device as per the link level protocol. For additional information, see the *DOS Technical Reference*, function call 40H.

An application must ensure that it stays in sync with the device and that responses from the device are kept in order with the command causing the response.

The maximum length of a command that may be transmitted to the device on a single write is 160 bytes.

Example

The following example shows how an application would use the 4779 DOS device driver to write a user defined command block to the 4779 and then read a response from the 4779

```
#include <dos.h>
#include <stdio.h>
#include <fcntl.h>
#include <share.h>
#include <io.h>
#include <conio.h>

//
// function prototypes
//

void EftRead ( void ) ;
int main ( void ) ;

//
//  data variables and buffers
//

char eftname[] = { "IBMEFT$ " } ;
int efthandle ;
char eftudcb[] =  {  x 4,  x47 } ;    //user-defined command block
unsigned char eftdata[512] ;

void EftRead ( ) {

   register int i ;

   while ( !( read ( efthandle, &eftdata, sizeof(eftdata) ) );

   lseek ( efthandle,  L, SEEK_SET ) ;   //reset the file pointer

}

int main ( ) {

   efthandle = sopen ( "IBMEFT$ ", O_RDWR | O_BINARY, SH_DENYRW ) ;
   lseek ( efthandle,  L, SEEK_SET ) ;   //reset the file pointer

   if ( -1 == write ( efthandle, &eftudcb, sizeof(eftudcb) ) )
      exit ( 1 ) ;
   EftRead ( ) ;

   close ( efthandle ) ;

   return   ;

}
```

Note: The above example assumes the 4779 device "understands" a user defined command block of *0447H* and that the device returns data in response to this user defined command block.

Chapter 3. OS/2 Device Driver and Support Code

The 4779 OS/2 device-support code provides for communication between a PC workstation and a single serially attached 4779 device. The 4779 device-support code also provides for shared serial port access for a 4779 device and a 4777 device.

The primary components of the 4779 OS/2 device-support code are a 16-bit dynamic link library, x4779OS2.DLL, and a 16-bit physical device driver, 4779OS2.SYS. The following table contains a complete list of the components of the 4779 OS/2 device-support code.

Figure 3-1. 4779 OS/2 Device-Support Code Component List

File Name	Description
4779OS2.SYS	4779 OS/2 Physical Device Driver
x4779OS2.DLL	4779 OS/2 Dynamic Link Library (DLL)
4779OS2.H	4779 OS/2 DLL Function Prototypes for IBM C/2
4779OS2V.H	4779 OS/2 DLL Function Prototypes for IBM VisualAge C++
FIO001.MSG	4717/18/77/78/79 OS/2 Support-Code Installation Msg File
FIO001H.MSG	4717/18/77/78/79 OS/2 Support-Code Installation Help File
4779INST.EXE	4779 OS/2 Installation Program
4779INST.MSG	4779 OS/2 Installation Program Message File
4779SUP2.EXE	4779 Support Program for OS/2
4779SUP.PNL	4779 Support Program Panel Library for DOS and OS/2
4779APD2.EXE	4779 Device Resident Application Download Program for OS/2
4779SZIP.EXE	Self-extracting image of the 4779 Sample Application Program Source & Executable Files for DOS and OS/2
MAGCALLS.DLL	4777 OS/2 Dynamic Link Library (DLL)
MAGDLL.H	4777 OS/2 DLL Function Prototypes for IBM C/2
4777DD.SYS	4777 MVDM Device Support Code
4777VDD.SYS	4777 Virtual Device Driver
4777SAP.EXE	4777 MVDM Support Application

There are two ways in which you can install the 4779 OS/2 device support code. The first method is manual and is described in “Manual Installation” on page 3-2 The second method is automatic and is described in “Automatic Installation” on page 3-4.

The remaining sections in the chapter describe how to install the 4779 device-support code and how to communicate with its application interface.

Manual Installation

This section describes how to manually load and initialize the 4779 OS/2 device driver and how to configure your operating environment. It includes the following sections:

Loading and initializing the physical device driver
Loading the dynamic link library
Loading the message files

Note: If a 4779 and 4777 are connected to a common RS232 communication port, the 4779 OS/2 physical device driver, 4779OS2.SYS, is used to support both devices. If the 4777/78 OS/2 physical device driver, FIOSERDD.SYS, has previously been installed, it *must* be removed from the config.sys file.

Loading and Initializing the Physical Device Driver

To load and initialize the 4779 physical device driver, copy file 4779OS2.SYS from the device drivers diskette to your target drive. Include the following DEVICE statement in your CONFIG.SYS file:

```
DEVICE = [d:[path]]4779OS2.SYS [/Cx/M/W/EFT]
```

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 4779OS2.SYS.

[/C*x*/M/W/EFT/Q*xx*/A*xxxx*]
The optional parameters define the following:

A serial port other than the default value (COM1)
Devices attached to the serial port
Use of delay operations when an error occurs
Use of a 4779 and 4777 on a single communication port

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

/C*x* (Do not select options /Q or /A) 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.

/Q*xx* (also select /A) Define the IRQ that the communication adapter is set to acknowledge, where *xx* can be 2,3,4,5,10,11 and 15. Do not specify option /C, an error will result on loading the device driver indicating a parameter error.

/A*xxxx* (also select /Q) Define the I/O address that the communication adapter is set to acknowledge,where *xxxx* can be any hexadecimal 4 digit number. Do not specify option /C, an error will result on loading the device driver indicating a parameter error.

/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.

/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.

/EFT Indicates that a 4779 device is attached. The device driver tests for a 4779 If the device driver does not find a 4779, it displays an error message. If it finds a 4779, and it passes the self-test, the application program can access the device driver.

If the device driver finds a 4779 and you do not specify the /EFT parameter, the device driver displays a specification error and the device driver is installed.

Loading and Initializing the Dynamic Link Libraries

To load the 4779 dynamic link library (DLL), copy file x4779OS2.DLL to the root directory or to a user-defined library directory. If a 4777 will be used, copy file MAGCALLS.DLL to the same directory. Identify this library directory to the operating system by modifying the LIBPATH statement in the CONFIG.SYS file as follows:

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

The parameters for the 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 x4779OS2.DLL file. If a 4777 device is also attached, the 4777 OS/2 DLL, MAGCALLS.DLL, must reside in this path as well.

Loading the Message Files

The OS/2 device-support code uses the FIO001.MSG and FIO001H.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 FIO001.MSG and FIO001H.MSG message files to the root directory or to a user-defined library directory as FIO.MSG and FIOH.MSG respectively. 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 FIO001.MSG and FIO001H.MSG message files.

Automatic Installation

This section describes how to load and initialize the 4779 OS/2 device-support code using the installation program provided. Based upon the options specified, the program will install the following types of support code.

4777 and 4779 OS/2 device-support code.
4777 MVDM device-support code.

To use the installation program, do the following from an OS/2 session.

1. Insert the 4779 DOS and OS/2 Device Driver diskette in the A:drive.
2. Type **A:4779INST**

4779INST displays a series of panels, providing the following options.

Target subdirectory to contain OS/2 device-support code
OS/2 application support
4777 DOS application support (MVDM)
Serial attachment options
Configuration parameters

Based on the selections made, 4779INST copies the required files to the target subdirectory. The workstation CONFIG.SYS file is automatically updated to include the appropriate device statements along with all necessary parameters. The target subdirectory selected is automatically added to the PATH, DPATH, and LIBPATH statements.

Once 4779INST has completed successfully, the workstation must be shutdown and restarted before the OS/2 device-support code can be used.

Application Programming Interface

The 4779 OS/2 Device-Support Code application programming interface consists of the following functions:

EftOpen Open a handle to the 4779 device
EftClose Close a handle to the 4779 device
EftRead Read a user defined response from the 4779 device
EftWrite Write a user defined data block to the 4779 device
EftAbort Terminate the current 4779 operation

Each of these functions is described in detail in the following sections. A simple example of how to invoke each function is included here. Additional examples of how to invoke the 4779 OS/2 Device-Support Code are provided in the 4779 Sample Application Program source code which may be found on the 4779 Device-Support Code Diskette. Refer to “Interpreting the API Return Codes” on page 3-14 for a description of the return codes associated with each function call.

EftOpen

The EftOpen function call opens a handle to the 4779 device.

```
USHORT APIENTRY _saveregs _loadds EftOpen(PHFILE EftHandle);
```

Parameters

The descriptions of the function call parameters are as follows:

EftHandle (PHFILE) *input*
This parameter is a far pointer to the variable where the device driver handle is returned. Your application uses this handle as the 4779 identifier for all subsequent 4779 function calls prior to and including the next EftClose command.

Returns

EftOpen returns the following word values:

0	No Error
2,5,6	OS/2 System Error (on DosOpen)
21	OS/2 4779 failed initialization when device driver loaded
1798	Unrecoverable Error
1801	Device PDD Not Open

Remarks

For each 4779 session, an application program must issue the EftOpen function call to access the 4779 device. EftOpen generates a new handle each time it is invoked. Subsequent calls by the application program to x4779OS2.DLL will reference the EftHandle value generated by EftOpen. At the end of each 4779 session, the application program should issue the EftClose function call.

Example Code

This example opens the 4779 device driver.

```
#include <os2.h>
#include "4779OS2.H"    /  4779 DLL function prototypes  /

HFILE  EftHandle;      /  4779  handle  /
USHORT rc;            /  Return code  /

rc = EftOpen((PHFILE)&EftHandle);
```

EftClose

The EftClose function call closes the 4779 device driver and releases it for use by other 4779 sessions or applications.

USHORT APIENTRY _saveregs _loadds EftClose(HFILE EftHandle);

Parameters

The descriptions of the function call parameters are as follows:

EftHandle (HFILE) *input*

This parameter is the device handle returned by EftOpen.

Returns

EftClose returns the following word values:

0	No Error
2,5,6	OS/2 System Error (on DosClose)
1793	Invalid Handle
1798	Unrecoverable Error
1801	Device PDD Not Open

Remarks

When an application program issues an EftClose function call, the 4779 device-support code closes the 4779 device driver and releases the device to other sessions and applications.

Example Code

This example closes the 4779 device and releases its ownership to other sessions or applications.

```
#include <os2.h>
#include "4779OS2.H"    /  4779  DLL function prototypes  /

HFILE  EftHandle;      /  4779  handle  /
USHORT rc;            /  Return code  /

rc = EftClose(EftHandle);
```

EftRead

The EftRead function call reads data from the 4779 device.

```
USHORT APIENTRY _saveregs _loadds EftRead(HFILE EftHandle, PVOID Buffer,
                                          USHORT BufferLength, PUSHORT BytesRead);
```

Parameters

The descriptions of the function call parameters are as follows:

EftHandle (HFILE) *input*
This parameter is the device handle returned by EftOpen.

Buffer (PVOID) *input*
This parameter is a far pointer to the buffer into which data is to be read.

BufferLength (USHORT) *input*
This parameter is the length in bytes of the data to be read. The maximum value of this parameter is 512. The minimum value is 1.

BytesRead (PUSHORT) *output*
This parameter is a far pointer to the number of bytes read.

Returns

EftRead returns the following values:

0	No Error.
1793	Invalid Handle
1795	Invalid BufferLength
1798	Unrecoverable Error
1799	Operation Aborted
1801	Device PDD Not Open
1802	Device Busy
1807	Sequence Error

Remarks

This function call reads data from the 4779 device. The request (thread) will be blocked until a response is received from the 4779, or until an EftAbort API request is issued.

An application must ensure that it stays in sync with the device and that responses from the device are kept in order with the command causing the response.

Example

This example reads data from the 4779 device to the application buffer.

```
#include <os2.h>
#include "4779OS2.H"    /  4779  DLL function prototypes  /

HFILE  EftHandle;       /  4779  handle  /
UCHAR  Buffer[1 5];     /  Data buffer  /
USHORT BufferLength;    /  Data buffer length  /
USHORT BytesRead;       /  Bytes read   /
USHORT rc;             /  Return code  /

BufferLength = sizeof(Buffer);
rc = EftRead(EftHandle, (PVOID)Buffer, BufferLength, (PUSHORT)&BytesRe
```

EftWrite

The EftWrite function call transfers data to the 4779 device.

```
USHORT APIENTRY _saveregs _loadds EftWrite(HFILE EftHandle, PVOID Buffer,
                                           USHORT BufferLength, PUSHORT BytesWritten);
```

Parameters

The descriptions of the function call parameters are as follows:

EftHandle (HFILE) *input*
This parameter is the device handle returned by EftOpen.

Buffer (PVOID) *input*
This parameter is a far pointer to the buffer which contains data to be transferred to the 4779 device.

BufferLength (USHORT) *input*
This parameter is the length in bytes of the data being transferred to the 4779 device. The maximum value of this parameter is 512. The minimum value is 1.

BytesWritten (PUSHORT) *output*
This parameter is required. A zero will always be returned in this parameter. A No Error return code indicates that all data was sent to the 4779 device.

Returns

EftWrite returns the following values:

0	No Error.
1793	Invalid Handle
1795	Invalid BufferLength
1798	Unrecoverable Error
1799	Operation Aborted
1801	Device PDD Not Open
1802	Device Busy
1807	Sequence Error

Remarks

This function call transfers data to the 4779 device.

An application must ensure that it stays in sync with the device and that responses from the device are kept in order with the command causing the response.

Example

This example transfers data to the 4779 device.

```
#include <os2.h>
#include "4779OS2.H"    /  4779  DLL function prototypes  /

HFILE  EftHandle;       /  4779  handle  /
UCHAR  Buffer[1 5];     /  Data buffer  /
USHORT BufferLength;    /  Data buffer length  /
USHORT BytesWritten;    /  Bytes written   /
USHORT rc;             /  Return code  /

BufferLength = sizeof(Buffer);
rc = EftWrite(EftHandle, (PVOID)Buffer, BufferLength, (PUSHORT)&BytesW
```

EftAbort

The EftAbort function call ends a 4779 I/O operation that is currently processing or that is pending.

```
USHORT APIENTRY _saveregs _loadds EftAbort(HFILE EftHandle);
```

Parameters

The descriptions of the function call parameters are as follows:

EftHandle (HFILE) *input*
This parameter is the device handle returned by EftOpen.

Returns

EftAbort returns the following word values:

0	No Error
1793	Invalid Handle
1798	Unrecoverable Error
1801	Device Busy
1810	No request (thread) was pending when Abort was issued.

Remarks

This function call disables the 4779 device. Before your application program issues an x4779OS2 function call, it must start another thread to support the EftAbort function call. Until the operator completes the 4779 operation, the thread for the 4779 function call is blocked. You can cancel the blocked thread with the EftAbort function call. The blocked thread is typically an EftRead that is awaiting a response from a 4779.

The EftAbort command aborts the 4779 by resetting the device when the last device command is not a keypad input related command. When the last device command is a keypad input related command, the 4779 is not reset.

Keypad input related commands :

X'03'	Read Keypad
X'09'	Generate PIN Block (X9.8)
X'0A'	Smart Card Password Verification (SAISS)
X'0B'	Generate PIN Block (3624)
X'0C'	Generate Offset (3624)
X'0F'	Verify PIN (3624)
X'12'	Generate PIN Block (X9.8) Parameter Version
X'13'	Generate Offset (3624) Parameter Version
X'14'	Verify PIN (3624) Parameter Version
X'21'	Generate Offset (3624) Comprehensive
X'22'	Verify PIN (3624) Comprehensive
X'30'	Smart Card Password Verification (MFC)

Example Code

This example invokes the EftAbort cpmmand.

```
#include <os2.h>
#include "4779OS2.H"    /  4779  DLL function prototypes  /

HFILE  EftHandle;      /  4779  handle  /
USHORT rc;            /  Return code  /

rc = EftAbort(EftHandle);
```

Interpreting the API Return Codes

This section describes in more detail the status codes returned by the 4779 OS/2 dynamic link library function calls described in Chapter 3, “OS/2 Device Driver and Support Code” on page 3-1. The return codes for the 4779 functions are in decimal form as follows:

0 **No Error**

Explanation: The function call completed successfully.

2 **OS/2 System Error - File Not Found**

Explanation: This code may be returned on a call to EftOpen or EftClose. It indicates that the OS/2 operating system cannot open the 4779 physical device driver (4779OS2.SYS). Refer to “Loading and Initializing the Physical Device Driver” on page 3-2 to verify that 4779OS2.SYS is properly installed. Then, retry the EftOpen or EftClose function.

5 **OS/2 System Error - Access Denied**

Explanation: This code may be returned on a call to EftOpen or EftClose. It indicates that OS/2 has denied you access to the 4779 physical device driver. Refer to “Loading and Initializing the Physical Device Driver” on page 3-2 to verify that 4779OS2.SYS is properly installed. Then, retry the EftOpen or EftClose function.

6 **OS/2 System Error - Invalid Handle**

Explanation: This code may be returned on a call to EftOpen or EftClose. It indicates that OS/2 has detected an invalid handle for the physical device driver. Refer to “Loading and Initializing the Physical Device Driver” on page 3-2 to verify that 4779OS2.SYS is properly installed. Then, retry the EftOpen or EftClose function.

21 **OS/2 System Error - Initialization Failed**

Explanation: This code is returned on a call to EftOpen. It indicates that the 4779 was not attached when the system loaded the physical device driver. Attach the 4779 Refer to “Loading and Initializing the Physical Device Driver” on page 3-2 to verify that 4779OS2.SYS is properly installed. Reload OS/2 on your workstation to properly load the 4779 physical device driver. Then, retry the EftOpen.

1793 **Invalid Handle**

Explanation: The EftHandle that was supplied on the function call does not match the EftHandle that was returned from the EftOpen function call.

1795 **Invalid BufferLength**

Explanation: The BufferLength parameter specified by the user is invalid. It is either greater than the maximum value or is equal to 0.

1798 Unrecoverable Error

Explanation: The 4779 DLL (x4779OS2.DLL) detected an unrecoverable error and ended the current function call.

1799 Operation Aborted

Explanation: The application program issued the EftAbort function call and ended the current I/O operation.

1801 Device PDD Not Open

Explanation: An operation was requested but the 4779 physical device driver is not open.

1802 Device Busy

Explanation: The application program issued a function call but the 4779 is not available. It is owned by a different handle.

1807 Sequence Error

Explanation: This error may occur as a response to either an EftRead or EftWrite function call. As a response to an EftRead call it means that the application program wanted to read data from the device but the 4779 device-support code is not expecting data from the device. As a response to an EftWrite call it means that the application program wanted to write data to the device but the 4779 device-support code is expecting data from the device.

1810 No Request Pending

Explanation: This error will occur when there is not a pending EftRead or EftWrite function call. There was no request pending that would require an abort. A request is pending when a EftWrite or EftRead has been issued and no response has been received. Care must be taken by the application to synchronize abort requests with pending responses.

Chapter 4. 4779 Device Resident Application Function Definitions

This chapter describes the non-security functions provided by the 4779 Resident Application Program. These functions, also referred to as *commands*, are available for use in programming the 4779 device to perform non-security functions. Commands associated with security are described in “4779 Security Function Calls.” Several of the commands are *comprehensive* in that they combine elemental commands into a high level function. The intent is to reduce the programming effort required to implement generally used transactions. The comprehensive commands include Manage Card Insertion, Generate Offset (3624) Comprehensive, and Verify PIN (3624) Comprehensive.

The following 4779 non-security commands are described in detail in the remainder of this chapter:

Figure 4-1. 4779 Non-Security Command Codes

Command	Description
X'00'	Read Device Information, see page 4-3
X'01'	Read Status, see page 4-3
X'02'	Write Display, see page 4-4
X'03'	Read Keypad, see page 4-5
X'04'	Media Arm, see page 4-6
X'05'	Eject Card, see page 4-7
X'06'	Magnetics Read, see page 4-7
X'07'	Magnetics Write, see page 4-9
X'08'	Smart Card Exchange, see page 4-10
X'09'	Generate PIN Block (ANSI X9.8), see page 4-10
X'12'	Generate PIN Block (ANSI X9.8) Parameter Version, see page 4-12
X'0A'	Smart Card Password Verification (SAISS), see page 4-13
X'30'	Smart Card Password Verification (MFC), see page 4-14
X'0B'	Generate PIN Block (IBM 3624), see page 4-15
X'0C'	Generate Offset (IBM 3624), see page 4-17
X'13'	Generate Offset (IBM 3624) Parameter Version, see page 4-19
X'21'	Generate Offset (IBM 3624) Comprehensive, see page 4-21
X'0F'	Verify PIN (IBM 3624), see page 4-24
X'14'	Verify PIN (IBM 3624) Parameter Version, see page 4-27
X'22'	Verify PIN (IBM 3624) Comprehensive, see page 4-29
X'20'	Manage Card Insertion, see page 4-32
X'11'	Execute Utility Functions, see page 4-34
X'0D'	Load Device Parameters, see page 4-35
X'10'	Load Verification Parameters, see page 4-37
X'0E'	Read Machine Information Data Structure, see page 4-40
X'F2'	Abort, see page 4-42
X'F3'	Reset Device, see page 4-42
X'F4'	Reset Security Function, see page 4-42
X'F5'	Query CRC of Device Memory, see page 4-43

The commands are formatted as indicated below:

Command - 1 Byte
Optional Parameters - n Bytes
Data - n bytes

The command responses are formatted as indicated below:

Expected Return Codes - 1 Byte
Data - n bytes

For a complete list and description of the 4779 Device Resident Application Program return codes, see Appendix B, “4779 Device Return Codes” on page B-1. For examples of how to invoke the 4779 Resident Application Program functions in a C Language PC host application, refer to the 4779 Sample Application Program which may be found on the 4779 Device-Support Code Diskette.

Read Device Information

This command is used to obtain basic information about the device. The format of the command is as follows:

Command - X'00'

The format of the response is as follows:

Expected Return Codes

- X'00' - No Error
- X'0A' - Security Error

Data -

- Serial Number - 8 Bytes ASCII string
- Main Processor Microcode Version Number - 14 byte ASCII string "v.vv, mm/dd/yy", where v.vv=version, mm=month, dd=day, yy=year
- Main Processor Application Identifier - 18 byte ASCII String
- Security Feature Microcode Version Number - 14 byte ASCII string "v.vv, mm/dd/yy", where v.vv=version, mm=month, dd=day, yy=year
- Security Feature Application Identifier - 16 byte ASCII String

Note: If a communications error occurs when the main processor tries to read device information from the security feature, the main processor information will still be returned, even though the response will contain an error code. Those fields which would have contained security feature data will be filled with question marks, to show that the information is unknown.

Read Status

This command is used to obtain the status of the device. The format of the command is as follows -

Command - X'01'

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'0E' - Feature Not Available

Data

- Status Byte 1

- Bit 7 - Motor timeout occurred
- Bit 6 - Chip Present
- Bit 5 - Track 3 Data Present
- Bit 4 - Track 2 Data Present
- Bit 3 - Track 1 Data Present
- Bit 2 - Card Detect 3 - Card Lock Status
- Bit 1 - Card Detect 2 - Card Present Status
- Bit 0 - Card Detect 1 - Card Present Status

– Status Bytes 2-n Chip Card Answer to Reset Data if Chip Present

The *Motor timeout occurred* bit is set whenever the device attempts to feed or eject a card, but times out before the card reaches its expected end point. This occurs if the card jams, or if something is wrong with the drive motor or the related electronic components.

Notes:

1. The card detect status bits are not operational if a timeout has occurred; if the timeout bit is on, all card detect bits will be off, regardless of the card position.
2. The Card Locked bit is turned OFF whenever the magnetic head is in motion, as an indication that the card is “unavailable.” It is turned back on when the head motion is complete.
3. Before issuing the MEDIA ARM command use the READ STATUS command to determine that a card has been entered into the device. The Card Detect 1 bit (Bit 0) and the Card Detect 2 bit (Bit 1) will be on(1) when the card is present at the entry to the device.

Write Display

This command is used to write up to 80 characters of data on the 4779 display (4 lines of 20 characters each). The format of the command is as follows.

Command - X'02'

x coordinate (Column) - 1 Byte (X'00-X13'),

y coordinate (row) - 1 Byte (X'00-X'03')

Attributes

– X'00' for normal mode
– All other values are reserved.

Data - 1-80 ASCII characters as described in Appendix G, “Character Font Table” on page G-1.

The format of the response is as follows.

Expected Return Codes

– X'00' - No Error
– X'02' - Data Length Error
– X'04' - Invalid Value
– X'0E' - Feature Not Available

Read Keypad

Using this command cleartext (unencrypted) data can be read from the keypad. Three methods of keystroke data collection are possible. Keystrokes may be read until the enter (#) key is pressed, until a specified number of keystrokes have been entered, or until a specified number of keystrokes have been entered or the enter (#) key is pressed. Invocation of this command will clear the display line (row 3) of existing information. Display information resulting from the use of this command will not be cleared by the command. The format of the command is as follows.

Command - X'03'

Parameter 1 - Keypad Entry Method

- X'00' - Key until Enter - With this method keystrokes will be read until the enter key (#) is pressed. The enter key will not be returned to the application as data.
- X'01' - Key for Specified Number or until Enter - With this option keystrokes will be read until the number specified as a parameter is reached, or until the enter (#) key is pressed. The enter key will not be returned to the application as data.
- X'02' - Key for Specified Number - With this option keystrokes will be read until the number specified as a parameter is reached, at which time the command will complete. If the enter (#) key is pressed during the entry process, this key will be returned as data to the application.

Parameter 2 - X'nn' - Number of keystrokes to read

- A value of X'00' is required if parameter 1 is X'00', key until enter. Parameter 1 values of X'01' or X'02' require the number of keystrokes to be read as parameter 2. The maximum number of keystrokes that can be read is 128.

Parameter 3 - Feedback Method

- X'00' - No Feedback - This option provides nothing visible or audible.
- X'01' - Beep - This option provides an audible indication of the keystroke.
- X'02' - Echo to Display - This option displays the key entered followed by the cursor.
- X'03' - Beep and Echo - This option displays the key entered followed by the cursor and an audible indication of the keystroke.
- X'04' - Asterisk - This option displays an asterisk in place of the echo character followed by the cursor.
- X'05' - Beep and Asterisk - This option displays an asterisk in place of the echo character followed by the cursor and an audible indication of the keystroke.

When keystrokes are echoed to the display, they are written on the last line of the display, starting in the first column. If the number of characters is greater than the display length, the line will scroll when the last character position is reached, so that the new characters continue to be visible. The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the keypad buffer and erase the display line.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error

– X'04' - Invalid Value
– X'0C' - Keypad Operation Aborted
– X'0E' - Feature Not Available

Data - The key strokes read from the keypad are returned as ASCII characters. The following list describes the characters that may be returned by this command.

Keycap	ASCII Code
0	30H
1	31H
2	32H
3	33H
4	34H
5	35H
6	36H
7	37H
8	38H
9	39H
F1	41H
F2	42H
F3	43H
F4	44H
#	0DH

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of the timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Media Arm

This command is used to arm the magnetic and smart card readers for read/write. Until armed the device can not be used to read or encode magnetic stripe cards or communicate with smart cards. The format of the command is

Command - X'04'

Parameter 1 b'xxxxxxxx'

– Bit 7 - reserved
– Bit 6 - Read/Write Chip
– Bit 5 - Reserved
– Bit 4 - Write Track 2
– Bit 3 - Reserved
– Bit 2 - Read Track 3
– Bit 1 - Read Track 2
– Bit 0 - Read Track 1

The format of the response is as follows -

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'0E' - Feature Not Available

Notes:

1. Only track 2 can be encoded.
2. The device can be armed for read *or* for write, but *not* for both at the same time.
3. Before issuing the Media Arm command use the Read Status command to determine that a card has been entered into the device. The Card Detect 1 bit(Bit 0) will be on(1) when the card is present at the entry to the device.

Eject Card

This command is used to eject a card from the device.

Command - X'05'

The format of the response is as follows -

Expected Return Codes

- X'00' - No Error

Magnetics Read

This command is used to obtain magnetic stripe data read from the card in the reader. Any combination of tracks 1, 2, or 3 can be read with a single request. This command requires that the Media Arm command has been successfully issued to arm the device prior to reading.

The data is returned in the form of a length byte followed by the data for the track. If more than one track is read, the length/data for each track is concatenated to that for the preceding track. The data is returned in ascending order of the track numbers; track 1, then track 2, and then track 3.

It is assumed that the tracks being read have been encoded in accordance with the ISO or ANSI standard for that track regarding character density and character set. For track 1, up to 64 characters may be encoded. Tracks 2 and 3 are limited to 16 characters (numeric and special characters). The 4779 device is designed to return the hexadecimal value of ASCII characters read from any of the three tracks. For instance, the ASCII character "A" is encoded on track 1 as a X'21' according to ISO or ANSI specifications. When reading track 1, the 4779 device adds X'20' to this value and returns a X'41', which is the hexadecimal equivalent of the ASCII character "A" (65 decimal). When reading tracks 2 and 3, the 4779 will add X'30' to the encoded value to return the hexadecimal value of the character.

In the description below, Lx, Ly, and Lz refer to the length bytes, while Tx, Ty, and Tz refer to the associated data from the magnetic stripe. Tx, Ty, and Tz are in the form of unterminated ASCII strings. For example, if the character read is X'2', the data returned by this command will be the character ASCII “2,” which has a hex value of X'32'.

The format of the command is as follows.

Command - X'06'

Parameter 1 -b'xxxxxxxx'

- Bit 7 - reserved
- Bit 6 - reserved
- Bit 5 - reserved
- Bit 4 - reserved
- Bit 3 - reserved
- Bit 2 - Track 3
- Bit 1 - Track 2
- Bit 0 - Track 1

The format of the response is as follows.

Expected Return Codes
- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'07' - Media Error

Lx - Length of Tx data, in bytes
Tx Data - Lx Bytes
Ly - Length of Ty data, in bytes
Ty Data - Ly Bytes
Lz - Length of Tz data, in bytes
Tz Data - Lz Bytes

Notes:

1. Ly, Ty, Lz, and Tz are only present if more than one track is to be read.

   Example: If Parameter 1 is X'03', the 4779 will return data for tracks 1 and 2, if available. As an example, assume track 1 contains the characters 0, 1, A, B, C, D and track 2 contains the digits 9, 8, 7 and 6. Each would be translated to its ASCII equivalent, so track 1 would become X'303141424344' (ASCII “01ABCD”), and track 2 would become X'39383736' (ASCII “9876”). The length of the track 1 data is 6 characters, and the length of track 2 is 4 characters. The data returned by the Magnetics Read command would then look like this:

```
        Track 1          Track 2


     6    3 3141424344    4    39383736


   Lengths                  Data
```

Figure 4-2. Example of Magnetics Read data

2. Reading must conform to ISO standards for bank cards with regard to the maximum number of characters read per track.

Magnetics Write

This command is used to encode data on track 2 of a magnetic stripe card in accordance with ANSI standard X4.16. As such only numeric characters are supported for encoding. Start-of-message (SOM) and end-of-message (EOM) characters are automatically added to the message being encoded by the supporting microcode. This command requires that the Media Arm command has been successfully issued to arm the device prior to encoding.

See “Magnetics Read” on page 4-7 for a description of the magnetic stripe data format used by this command. The data read from the card is in the same format as the data to be written to the card.

The format of the command is as follows.

Command - X'07'

Parameter 1 -b'xxxxxxxx'

- Bit 7 - Reserved
- Bit 6 - Reserved
- Bit 5 - Reserved
- Bit 4 - Reserved
- Bit 3 - Reserved
- Bit 2 - Reserved
- Bit 1 - Track 2
- Bit 0 - Reserved

Data

- L - Length of track 2 data, in bytes
- T Data - L Bytes

Notes:

1. The data T is in the form of unterminated ASCII string. For example, to encode a character of X'2', the data passed with this command must be the character ASCII “2,” which has a hex value of X'32'.
2. The maximum number of characters that may be encoded on track 2 is 37.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'07' - Media Error
- X'0B' - Timeout

Smart Card Exchange

This command is used to send a command to the smart card chip and receive a response. This command requires that the Media Arm command has been successfully issued to arm the smart card prior to using the Smart Card Exchange command.

The format of the command is as follows.

Command - X'08'

CLA - Class Byte

INS - Instruction Byte

P1 - Parameter 1 Byte

P2 - Parameter 2 Byte

LN - Data Length Byte (Send or receive depending on the smart card instruction)

N Data Bytes (Depending on the smart card instruction)

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'05' - Hardware Error
- X'06' - Sequence Error
- X'07' - Media Error
- X'0E' - Feature Not Available

N Data Bytes (depending on smart card instruction)

Smart Card Return Code (RC) - 2 Bytes

In order to understand this command, the user should be familiar with smart cards and their protocols. For general information on smart cards, refer to the 7816 standards described in “References” on page xii. For a description of the commands and data formats supported for smart cards, see the Interbank Smartcard document described in the same list. For other smart cards, consult the manual supplied by the smart card vendor.

Generate PIN Block (ANSI X9.8)

This command is used to format an encrypted PIN block for use by the application. It uses a PAN (Primary Account Number) read from a magnetic stripe card, a PIN as entered by the user and a specified encryption key to create the ANSI X9.8 formatted PIN block.

The device will perform the following steps.

Read the PIN once or twice as it is entered on the keypad depending upon the PIN entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Read the PAN from the magnetic stripe card data buffer.

Format the PIN block according to the ANSI X9.8 standard using the specified encryption key.

The PC application program should prompt the user to insert the magnetic stripe card and enter the PIN on the keypad once or twice depending upon the option selected. Prior to using this command, the application must successfully issue the Media Arm command to initialize the device to read the magnetic stripe. This command will read data from track two until the designated ANSI X9.8 separator character is reached indicating the conclusion of the PAN information. In accordance with ANSI X9.8 standards, the PAN read from the magnetic stripe will be modified by excluding the check digit (last digit) prior to it's use in formatting the encrypted PIN block.

The keypad operation will complete if the enter (#) key is pressed or the maximum number of digits is entered. The number of PIN digits for the ANSI X9.8 format may range from 4 (X'04') to 12 (X'0C').

The format of the command is as follows.

Command - X'09'

Key Indicator - X'xx' - This parameter specifies the number of the DES data key to be used.

PIN Entry Type - 1 Byte - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.

- Single PIN Entry - X'00'
- Dual PIN Entry - X'01'

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'07' - Card Error
- X'08' - Bad Key
- X'0A' - Security Error
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

Formatted PIN Block - 8 Bytes

Note: The ANSI X9.8 PIN block format is described in the 4777/4778 manual referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of the timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device

Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Generate PIN Block (ANSI X9.8) Parameter Version

This command is used to format an encrypted PIN block for use by the application without the use of a magnetic stripe card. It uses a PAN entered as a parameter, a PIN as entered by the user and a specified encryption key to create the ANSI X9.8 formatted PIN block.

The device will perform the following steps.

Read the PIN once or twice as it is entered on the keypad depending upon the PIN entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Format the PIN block according to the ANSI X9.8 standard using the specified encryption key.

The PC application program should prompt the user to enter the PIN once or twice on the keypad. This command will perform no modifications on the PAN entered as a parameter to this command.

The keypad operation will complete if the enter (#) key is pressed or the maximum number of digits is entered. The number of PIN digits for the ANSI X9.8 format may range from 4 (X'04') to 12 (X'0C').

The format of the command is as follows.

Command - X'12'

Key Indicator - X'xx' - This parameter specifies the number of the DES data key to be used.

PIN Entry Type - 1 Byte - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.

- Single PIN Entry - X'00'
- Dual PIN Entry - X'01'

PAN (12 Digit Account Number) - 6 bytes of PAN. For example the account number 11 22 33 44 55 66 would be specified as X'112233445566'.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'08' - Bad Key
- X'0A' - Security Error
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN

– X'0E' - Feature Not Available

Formatted PIN Block - 8 Bytes

Note: The ANSI X9.8 PIN block format is described in the 4777/4778 manual referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of the timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Smart Card Password Verification (SAISS)

This command is used in order to have the smart card verify a password in accordance with the South African Interbank Smartcard Standards. The number of password digits may range from 4(X'04') to 8(X'08').

The device will perform the following steps.

Read the password once or twice as it is entered on the keypad depending upon the password entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Send the password data to the smart card using the card's *Client Card Check Password* command.
Return a message to the PC indicating whether the password verification was successful.

The PC application program should prompt the user to enter a password once or twice on the keypad depending upon the entry type selected. This command will provide no messages to the display.

The format of the command is as follows.

Command - X'0A'

Password Number - X'00' to X'07'

Password Entry Type

– X'00' - Single Entry - With this option, the command requires only one entry of the password.
– X'01' - Dual Entry - With this option, two inputs of the password are required. The passwords are compared for validity before data is sent to the smart card.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'09' - Bad Password
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

Note: The format and use of this command are described in the Interbank Smartcard document referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of the timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Smart Card Password Verification (MFC)

This command is used in order to have the smart card verify a password in to support the IBM Multi Function Card. The number of password digits may range from 4(X'04') to 8(X'08').

In order to achieve compatibility between the MFC card and this command, the MFC card password must be initialized with the pad character 0X'30'(ASCII zero). For example, a password of '1234' would be defined to the MFC card as 0X31, 0X32, 0X33, 0X34, 0X30, 0X30, 0X30, 0X30. Refer to the MFC documentation for specific information related to card initialization.

The device will perform the following steps.

Read the password once or twice as it is entered on the keypad depending upon the password entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Send the password data to the smart card using the card's *Client Card Check Password* command.
Return a message to the PC indicating whether the password verification was successful.

The PC application program should prompt the user to enter a password once or twice on the keypad depending upon the entry type selected. This command will provide no messages to the display.

The format of the command is as follows.

Command - X'30'

Password Number - X'01' to X'02'

Password Entry Type

- X'00' - Single Entry - With this option, the command requires only one entry of the password.
- X'01' - Dual Entry - With this option, two inputs of the password are required. The passwords are compared for validity before data is sent to the smart card.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'09' - Bad Password
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

Note: The format and use of this command are described in the Interbank Smartcard document referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of the timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Generate PIN Block (IBM 3624)

This command is used to format an encrypted PIN block for use by the application. It uses a PIN as entered by the user and a specified encryption key to create the IBM 3624 formatted PIN block.

The device will perform the following steps.

Read the PIN once or twice as it is entered on the keypad depending upon the PIN entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Pad the entered PIN with the specified pad character.

Format the PIN block according to the IBM 3624 standard.

The PC application program should prompt the user to enter the PIN on the keypad.

The keypad operation will complete if the enter (#) key is pressed or the maximum number of digits is entered. The number of PIN digits for the 3624 format may range from 1 (X'01') to 16 (X'10').

The format of the command is as follows.

Command - X'0B'

Key Indicator - X'xx' - This parameter specifies the number of the DES data key to be used.

PIN Entry Type - 1 Byte - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.

- Single PIN Entry - X'00'
- Dual PIN Entry - X'01'

Pad Character - X'0x'

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'08' - Bad Key
- X'0A' - Security Error
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

Formatted PIN Block - 8 Bytes

Note: The IBM 3624 PIN block format is described in the 4777/4778 manual referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of the timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Generate Offset (IBM 3624)

This command is used to generate PIN offset data using the 3624 offset generation algorithm. This command will read validation data from the card based upon the location and length of this information. The validation data is encrypted under a specified key. It is used with a decimalization table and the entered PIN to generate an offset. The offset is returned to the application.

The specific execution of this command is governed by a table of parameters. If parameter values other than the default are to be used, then the command “Load Verification Parameters” on page 4-37 must be issued for those values prior to executing this command. The default verification parameter values are specified below.

This command will not modify validation data read from the magnetic stripe card. If check digit(last digit) removal is required, the validation data length and validation data offset parameters may be altered to achieve this. A description of these parameters is provided below.

The keypad operation will complete if the enter (#) key is pressed or the maximum number of digits (12) is entered.

The device will perform the following steps.

Use track data previously read from the magnetic stripe card.
Read the PIN once or twice as it is entered on the keypad depending upon the PIN entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Encrypts the validation data with the specified encryption key.
Calculates the offset data and returns the offset to the application.

The PC application program should perform the following to implement this command.

Issue the Media Arm command to initialize the magnetic stripe reader.
Prompt the user to insert the card that track may be read.
Issue the Magnetics Read command to read the required track.
Prompt the user to enter a PIN once or twice on the keypad.
Issue the Generate Offset (3624) command to calculate the offset.
Issue the Eject command to remove the card from the device.

The format of the command is as follows.

Command - X'0C'

The following parameter values are utilized by this command. If the specified default values require modification, use the "Load Verification Parameters" command to implement a change.

Key Indicator - This value specifies the DES data key number to be used to encrypt the validation data. The default is X'00'.
PIN Length - This value specifies the number of offset digits to be created by the execution of this command. This value must be equal to or less than the number of entered PIN digits. The length may range from 1 to 16. The default length is 4.

PIN Entry Type - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.
Decimalization Table - This parameter specifies a 16 byte array that is used to convert encrypted data from hexadecimal to decimal. The default is as follows.

Position	Table Value
0	00H
1	01H
2	02H
3	03H
4	04H
5	05H
6	06H
7	07H
8	08H
9	09H
A	00H
B	01H
C	02H
D	03H
E	04H
F	05H

Validation Data Track - This parameter specifies the track for the validation data. The default is track two.
Validation Data Length - This parameter specifies the length of the validation data. The default is 16 bytes.
Validation Data Offset - This parameter specifies the location of the validation data relative to the first encoded data position. The default is X'00.
Validation Data Pad Character - This value indicates the pad character to be used for encryption when the validation data is less than 16 bytes. It is specified as X'0x'. The default pad character is X'00'. To specify the character 'F' as the pad character, this parameter would be X'0F'.
Offset Data Pad Character - This value specifies the pad character to be used when the offset is created. The default for this parameter is the character X'00'. When this command is used, the pad character returned to the application is the byte specified by this parameter. For example, if X'FF' is specified, 'FF' will be returned as the pad character.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'07' - Card Error
- X'08' - Bad Key
- X'0A' - Security Error
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

The calculated offset data is returned in the form of a 16 byte unterminated ASCII string. Data exceeding the PIN length will be padded with the specified or default offset data pad character.

Note: The IBM 3624 PIN block format is described in the 4777/4778 manual referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of this timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Generate Offset (IBM 3624) Parameter Version

This command is used to generate PIN offset data using the 3624 offset generation algorithm without the use of a magnetic stripe card. This command will use the validation data entered as a parameter to this command. The validation data is encrypted under a specified key. It is used with a decimalization table and the entered PIN to generate an offset. The offset is returned to the application.

The specific execution of this command is governed by a table of parameters. If parameter values other than the default are to be used, then the command “Load Verification Parameters” on page 4-37 must be issued for those values prior to executing this command. The default verification parameter values are specified below.

This command will perform no modifications to the validation data entered as a parameter to this command.

The keypad operation will complete if the enter (#) key is pressed or the maximum number of digits (12) is entered.

The device will perform the following steps.

Read the PIN once or twice as it is entered on the keypad depending upon the PIN entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Encrypts the validation data with the specified encryption key.
Calculates the offset data and returns the offset to the application.

The PC application program should perform the following to implement this command.

Prompt the user to enter a PIN once or twice on the keypad.
Issue the Generate Offset (3624) Parameter Version command to calculate the offset.

The format of the command is as follows.

Command - X'13'

Validation Data - 8 Bytes - All eight bytes of the validation will be encrypted by this command.

The following parameter values are utilized by this command. If the specified default values require modification, use the "Load Verification Parameters" command to implement a change.

Key Indicator - This value specifies the DES data key number to be used to encrypt the validation data. The default is X'00'.
PIN Length - This value specifies the number of offset digits to be created by the execution of this command. This value must be equal to or less than the number of entered PIN digits. The length may range from 1 to 16. The default length is 4.
PIN Entry Type - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.
Decimalization Table - This parameter specifies a 16 byte array that is used to convert encrypted data from hexadecimal to decimal. The default is as follows.

Position	Table Value
0	00H
1	01H
2	02H
3	03H
4	04H
5	05H
6	06H
7	07H
8	08H
9	09H
A	00H
B	01H
C	02H
D	03H
E	04H
F	05H

Offset Data Pad Character - 1 Byte - This value specifies the pad character to be used when the offset is created. The default for this parameter is the character X'00'. When this command is used, the pad character returned to the application is the byte specified by this parameter. For example, if X'FF' is specified, 'FF' will be returned as the pad character.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'08' - Bad Key
- X'0A' - Security Error
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

The calculated offset data is returned in the form of a 16 byte unterminated ASCII string. Data exceeding the PIN length will be padded with the specified or default offset data pad character.

Note: The IBM 3624 PIN block format is described in the 4777/4778 manual referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of this timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Generate Offset (IBM 3624) Comprehensive

This command is an extension of the Generate Offset (3624) command in that it provides for not only calculation of the offset, but card insertion and encoding of the offset in one command. The user is initially prompted to enter their magnetic stripe card in the device. If it is inserted incorrectly, the user is first prompted to observe the card orientation and then is prompted to re-insert the card. If the card is not correctly inserted within the time allotted by the card insertion timeout parameter, the command will terminate returning an appropriate return code. When the card is correctly inserted in the device, the user is prompted to enter their PIN. If the entered PIN length does not equal the expected length, defined by the validation parameters, the user is notified and prompted to re-enter the PIN. The keypad operation may be terminated either by exceeding a timeout, depressing a predefined cancel key, or with a command from the PC application. The card validation data is read based upon the location and length of this information on the specified track. The validation data is encrypted under a specified key. It is used with a decimalization table and the entered PIN to generate an offset. The offset is encoded on the magnetic stripe card in the device at the location and track specified by the validation parameters. The length of the encoded offset may also be specified by the validation parameters.

The presence of a card correctly inserted in the device is determined by the existence of data encoded on any track of the magnetic stripe. Therefore, in the case of magnetic stripe cards, use of this command assumes that data is present on at least on track.

The specific execution of this command is governed by a table of parameters. If parameter values other than the default are to be used, then the command “Load Verification Parameters” on page 4-37 must be issued for those values prior to executing this command. The default verification parameter values are specified below.

This command will not modify validation data read from the magnetic stripe card. If check digit(last digit) removal is required, the validation data length and validation data offset parameters may be altered to achieve this. A description of these parameters is provided below.

The keypad operation will complete if the enter (#) key is pressed or the maximum number of digits (12) is entered.

The device will perform the following steps.

Prompt the user to enter their card in the device.
Insure that the card correctly positioned in the device and read track data from the magnetic stripe card.
Prompt the user to enter their PIN for either single or dual PIN entry.
Read the PIN once or twice as it is entered on the keypad depending upon the PIN entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Encrypts the validation data with the specified encryption key.
Calculates the offset data and encodes the offset on the magnetic stripe card in the device.

The PC application program should perform the following to implement this command.

Initialize the device parameters as required by the application. This may require the use of the Load Device Parameters command, the Load Verification Parameters command, or security key loading commands. This will be required if the default parameter values for these commands are not used.
Issue the Generate Offset (3624) Comprehensive command to calculate the offset and verify the completion status of the command. .

The format of the command is as follows.

Command - X'21'

The following messages may be displayed with this command.

Message 1
- Line 0 - Please
- Line 1 - insert your card

Message 2
- Line 0 - Please
- Line 1 - remove your card

Message 3
- Line 0 - Please re-insert
- Line 1 - your card

Message 4
- Line 0 - Please check card
- Line 1 - for correct
- Line 2 - orientation

Message 5
- Line 0 - Please
- Line 1 - enter your PIN

Message 6
- Line 0 - Please
- Line 1 - re-enter your PIN

– Line 2 - for verification

Message 7

– Line 1 - Card insertion error

The following parameter values are utilized by this command. If the specified default values require modification, use the "Load Verification Parameters" command to implement a change.

Key Indicator - This value specifies the DES data key number to be used to encrypt the validation data. The default is X'00'.

PIN Length - This value specifies the number of offset digits to be created by the execution of this command. This value must be equal to or less than the number of entered PIN digits. The length may range from 1 to 16. The default length is 4.

PIN Entry Type - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.

Decimalization Table - This parameter specifies a 16 byte array that is used to convert encrypted data from hexadecimal to decimal. The default is as follows.

Position	Table Value
0	00H
1	01H
2	02H
3	03H
4	04H
5	05H
6	06H
7	07H
8	08H
9	09H
A	00H
B	01H
C	02H
D	03H
E	04H
F	05H

Validation Data Track - This parameter specifies the track for the validation data. The default is track two.

Validation Data Length - This parameter specifies the length of the validation data. The default is 16 bytes.

Validation Data Offset - This parameter specifies the location of the validation data relative to the first encoded data position. The default is X'00.

Validation Data Pad Character - This value indicates the pad character to be used for encryption when the validation data is less than 16 bytes. It is specified as X'0x'. The default pad character is X'00'. To specify the character 'F' as the pad character, this parameter would be X'0F'.

Offset Data Track for Encoding - This value specifies the magnetic stripe track to which the offset data may be encoded. The default is track two.

Offset Data Pad Character - This value specifies the pad character to be used when the offset is created. The possible values are the numerics 0 - 9. When using this command the pad character is returned as an ASCII character since it is being encoded. For instance, X'01' is returned as '31'. The default for this parameter is the character X'30'.

Offset Data Offset for Encoding - This value specifies the start location of the encoded offset field from the first data position on the track. The first data

position is designated as zero for the purpose of defining this parameter. The default for this parameter is the first data position after the existing track data. This default is designated by X'FF'.
Offset Data Length for Encoding - This value specifies the length of the offset to be encoded. The default is 16 characters.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'08' - Bad Key
- X'07' - Card Error
- X'0A' - Security Error
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

The calculated offset data is encoded on the magnetic stripe as specified by the verification parameters that describe the length and location. Data exceeding the PIN length will be padded with the specified or default offset data pad character.

Notes:

1. The IBM 3624 PIN block format is described in the 4777/4778 manual referenced in “References” on page xii.
2. Encoding must conform to ISO standards for bank cards with regard to the number of characters encoded per track.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of this timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Verify PIN (IBM 3624)

This command is used to verify an entered PIN using the 3624 algorithm. This command will read validation and offset data from the card based upon the location and length of this data. The validation data is encrypted under the specified key. It is used with a decimalization table and offset data to calculate a PIN. This PIN is

compared to the entered PIN and the result of the comparison is return to the application.

The specific execution of this command is governed by a table of parameters. If parameter values other than the default are to be used, then the command “Load Verification Parameters” on page 4-37 must be issued for those values prior to executing this command. The default verification parameter values are specified below.

This command will not modify validation data read from the magnetic stripe card. If check digit(last digit) removal is required, the validation data length and validation data offset parameters may be altered to achieve this. A description of these parameters is provided below.

When entering the PIN, this function expects the enter (#) key will be pressed to indicate the end of the entered PIN if the number of PIN digits is less than the maximum of 12. In other words, the keypad operation will complete if the enter (#) key is pressed or the maximum number of digits is entered.

The device will perform the following steps.

Use track data previously read from the magnetic stripe card.
Read the PIN once or twice as it is entered on the keypad depending upon the PIN entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Encrypts the validation data with the specified encryption key.
Calculates the PIN based on track data.
Verifies the entered PIN and returns the result of the comparison.

The PC application program should perform the following to implement this command.

Issue the Media Arm command to ready the magnetic stripe reader.
Prompt the user to insert the card that track may be read.
Issue the Magnetics Read command to read the required track.
Prompt the user to enter a PIN once or twice on the keypad.
Issue the Verify PIN (3624) command.
Issue the Eject command to remove the card from the device.

The format of the command is as follows.

Command - X'0F'

The following parameter values are utilized by this command. If the specified default values require modification, use the "Load Verification Parameters" command to implement a change.

Key Indicator - This value specifies the DES data key number to be used to encrypt the validation data. The default is X'00'.
PIN Length - This value specifies the number of offset digits to be created by the execution of this command. This value must be equal to or less than the number of entered PIN digits. The length may range from 1 to 16. The default length is 4.
PIN Entry Type - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.
Decimalization Table - This parameter specifies a 16 byte array that is used to convert encrypted data from hexadecimal to decimal. The default is as follows.

Position	Table Value
0	00H
1	01H
2	02H
3	03H
4	04H
5	05H
6	06H
7	07H
8	08H
9	09H
A	00H
B	01H
C	02H
D	03H
E	04H
F	05H

Validation Data Track - This parameter specifies the track for the validation data. The default is track two.
Validation Data Length - This parameter specifies the length of the validation data. The default is 16 bytes.
Validation Data Offset - This parameter specifies the location of the validation data relative to the first encoded data position. The default is X'00.
Validation Data Pad Character - This value indicates the pad character to be used for encryption when the validation data is less than 16 bytes. It is specified as X'0x'. The default pad character is X'00'. To specify the character 'F' as the pad character, this parameter would be X'0F'.
Offset Data Track for Reading - This value specifies the track from which the offset data is to be read. The default is track two.
Offset Data Length for Reading - This value specifies the length of the offset to be read. The default is the PIN Length of 4. A change to the PIN length parameter will automatically be reflected in the offset data length for reading parameter. Use of this parameter will allow an offset length to be read that is different from the PIN length.
Offset Data Offset for Reading - This value specifies the start location of the offset data field from the first data position on the track. The first data position is designated as zero for the purpose of defining this parameter. The default for this parameter is X'00'.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'07' - Card Error
- X'08' - Bad Key
- X'0A' - Security Error
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

Note: The IBM 3624 PIN block format is described in the 4777/4778 manual referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of this timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Verify PIN (IBM 3624) Parameter Version

This command is used to verify an entered PIN using the 3624 algorithm without the use of a magnetic stripe card. This command will use validation and offset data entered as parameters to this command. The validation data is encrypted under the specified key. It is used with a decimalization table and offset data to calculate a PIN. This PIN is compared to the entered PIN and the result of the comparison is returned to the application.

The specific execution of this command is governed by a table of parameters. If parameter values other than the default are to be used, then the command “Load Verification Parameters” on page 4-37 must be issued for those values prior to executing this command. The default verification parameter values are specified below.

This command will perform no modifications on the validation data or offset data entered as parameters to this command.

When entering the PIN, this function expects the enter (#) key will be pressed to indicate the end of the entered PIN if the number of PIN digits is less than the maximum of 12. In other words, the keypad operation will complete if the enter (#) key is pressed or the maximum number of digits is entered.

The device will perform the following steps.

Read the PIN once or twice as it is entered on the keypad depending upon the PIN entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Encrypts the validation data with the specified encryption key.
Calculates the PIN based on the input parameters.
Verifies the entered PIN and returns the result of the comparison.

The PC application program should perform the following to implement this command.

Prompt the user to enter a PIN once or twice on the keypad.
Issue the Verify PIN (3624) Parameter Version command.

The format of the command is as follows.

Command - X'14'

Validation Data - 8 Bytes - All eight bytes of the validation will be encrypted by this command.

Offset Data - 8 Bytes - X'nn', where 'nn' is the number of bytes based on the variable PIN Length. This is followed by the pad character 'F'. The pad character used here functions only to maintain field length integrity. For example the offset 11 22 would be specified as X'1122FFFFFFFFFFFF'. If offset data is not used, enter zeroes in this field.

The following parameter values are utilized by this command. If the specified default values require modification, use the "Load Verification Parameters" command to implement a change.

Key Indicator - This value specifies the DES data key number to be used to encrypt the validation data. The default is X'00'.
PIN Length - This value specifies the number of offset digits to be created by the execution of this command. This value must be equal to or less than the number of entered PIN digits. The length may range from 1 to 16. The default length is 4.
PIN Entry Type - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.
Decimalization Table - This parameter specifies a 16 byte array that is used to convert encrypted data from hexadecimal to decimal. The default is as follows.

Position	Table Value
0	00H
1	01H
2	02H
3	03H
4	04H
5	05H
6	06H
7	07H
8	08H
9	09H
A	00H
B	01H
C	02H
D	03H
E	04H
F	05H

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'08' - Bad Key
- X'0A' - Security Error
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

Note: The IBM 3624 PIN block format is described in the 4777/4778 manual referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of this timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Verify PIN (IBM 3624) Comprehensive

This command is an extension of the Verify PIN (3624) command in that it provides for not only PIN verification but card insertion. The user is initially prompted to enter their magnetic stripe card in the device. If it is inserted incorrectly, the user is first prompted to observe the card orientation and then is prompted to re-insert the card. If the card is not correctly inserted within the time allotted by the card insertion timeout parameter, the command will terminate returning an appropriate return code. When the card is correctly inserted in the device, the user is prompted to enter their PIN. If the entered PIN length is less than the expected length, defined by the validation parameters, an error condition occurs and the command terminates. The keypad operation may be terminated either by exceeding a timeout, depressing a predefined cancel key, or with a command from the PC application. The card validation and offset data are read based upon the location and length of this information on the specified track. The validation data is encrypted under a specified key. It is used with a decimalization table and offset data to calculate a PIN. This PIN is compared to the entered PIN and the result of the comparison is returned to the application.

The presence of a card correctly inserted in the device is determined by the existence of data encoded on any track of the magnetic stripe. Therefore, in the case of magnetic stripe cards, use of this command assumes that data is present on at least on track.

The specific execution of this command is governed by a table of parameters. If parameter values other than the default are to be used, then the command “Load Verification Parameters” on page 4-37 must be issued for those values prior to executing this command. The default verification parameter values are specified below.

This command will not modify validation data read from the magnetic stripe card. If check digit(last digit) removal is required, the validation data length and validation data offset parameters may be altered to achieve this. A description of these parameters is provided below.

When entering the PIN, this function expects the enter (#) key will be pressed to indicate the end of the entered PIN if the number of PIN digits is less than the maximum of 12. In other words, the keypad operation will complete if the enter (#) key is pressed or the maximum number of digits is entered.

The device will perform the following steps.

Prompt the user to enter their card in the device.
Insure that the card correctly positioned in the device and read track data from the magnetic stripe card.
Prompt the user to enter their PIN for either single or dual PIN entry.
Read the PIN once or twice as it is entered on the keypad depending upon the PIN entry type. The command will display an asterisk on the bottom line of the LCD for each keystroke.
Encrypts the validation data with the specified encryption key.
Calculates the PIN based on track data.
Verifies the entered PIN and returns the result of the comparison.

The PC application program should perform the following to implement this command.

Initialize the device parameters as required by the application. This may require the use of the Load Device Parameters command, the Load Verification Parameters command, or security key loading commands. This is required if the default parameter values for these commands are not used.
Issue the Verify PIN (3624) Comprehensive command to verify the PIN and verify the completion status of the command.

The format of the command is as follows.

Command - X'22'

The following messages may be displayed with this command.

Message 1
- Line 0 - Please
- Line 1 - insert your card

Message 2
- Line 0 - Please
- Line 1 - remove your card

Message 3
- Line 0 - Please re-insert
- Line 1 - your card

Message 4
- Line 0 - Please check card
- Line 1 - for correct
- Line 2 - orientation

Message 5
- Line 0 - Please
- Line 1 - enter your PIN

Message 6
- Line 0 - Please
- Line 1 - re-enter your PIN
- Line 2 - for verification

Message 7
- Line 1 - Card insertion error

The following parameter values are utilized by this command. If the specified default values require modification, use the Load Verification Parameters command to implement a change.

Key Indicator - This value specifies the DES data key number to be used to encrypt the validation data. The default is X'00'.

PIN Length - This value specifies the number of offset digits to be created by the execution of this command. This value must be equal to or less than the number of entered PIN digits. The length may range from 1 to 16. The default length is 4.

PIN Entry Type - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.

Decimalization Table - This parameter specifies a 16 byte array that is used to convert encrypted data from hexadecimal to decimal. The default is as follows.

Position	Table Value
0	00H
1	01H
2	02H
3	03H
4	04H
5	05H
6	06H
7	07H
8	08H
9	09H
A	00H
B	01H
C	02H
D	03H
E	04H
F	05H

Validation Data Track - This parameter specifies the track for the validation data. The default is track two.

Validation Data Length - This parameter specifies the length of the validation data. The default is 16 bytes.

Validation Data Offset - This parameter specifies the location of the validation data relative to the first encoded data position. The default is X'00.

Validation Data Pad Character - This value indicates the pad character to be used for encryption when the validation data is less than 16 bytes. It is specified as X'0x'. The default pad character is X'00'. To specify the character 'F' as the pad character, this parameter would be X'0F'.

Offset Data Track for Reading - This value specifies the track from which the offset data is to be read. The default is track two.

Offset Data Length for Reading - This value specifies the length of the offset to be read. The default is the PIN Length of 4. A change to the PIN length parameter will automatically be reflected in the offset data length for reading parameter. Use of this parameter will allow an offset length to be read that is different from the PIN length.

Offset Data Offset for Reading - This value specifies the start location of the offset data field from the first data position on the track. The first data position is designated as zero for the purpose of defining this parameter. The default for this parameter is X'00'.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'07' - Card Error
- X'08' - Bad Key
- X'0A' - Security Error
- X'0B' - Timeout
- X'0C' - Keypad Operation Aborted
- X'0D' - Invalid PIN
- X'0E' - Feature Not Available

Note: The IBM 3624 PIN block format is described in the 4777/4778 manual referenced in “References” on page xii.

A timeout can be defined to limit the amount of time the device will wait for each keystroke. For a description of this timeout parameter and its use, see “Load Device Parameters” on page 4-35.

The erase (*) key may be used to delete entered keystrokes. This will remove keystrokes from the buffer and erase the display line.

The user has the ability to abort a keypad operation by pressing a predefined key for this purpose. Initializing this key value is done with the command “Load Device Parameters” on page 4-35. Load Device Parameters command. The default abort key is none. The host application will be notified of an abort event.

In the DOS environment, this function may also be terminated with the application issued abort command. Refer to “Abort” on page 4-42. To cancel this command from the OS/2 environment, use the EftAbort function described in chapter 3.

Manage Card Insertion

This command is used to manage the insertion of conventional credit cards and smart cards into the device. This function will issue all prompt messages and check all error conditions associated with the insertion of a card. Initially the function will check for the presence of a card in the device. If one is detected, it is ejected with a display prompt to remove the card. If the device is clear of a card, the user is prompted to enter their card. If it is inserted incorrectly, the user is first prompted to observe the card orientation and then is prompted to re-insert the card. If the card is not correctly inserted within the time allotted by the card insertion timeout parameter, the command will terminate returning an appropriate return code.

Successful completion of this command indicates that the card is correctly positioned in the device and is ready for subsequent processing. At this point the device is armed to read any of the three magnetic stripe tracks or armed to communicate with smart cards. If encoding is to be the next operation, a Media Arm command for that purpose must be issued.

The presence of a card correctly inserted in the device is determined by the existence of data encoded on any track of the magnetic stripe. Therefore, in the case of magnetic stripe cards, use of this command assumes that data is present on at least one track.

The device will perform the following steps.

Check for presence of card in device.
Prompt the user enter their card.
Check for correct orientation of card, eject if incorrect.
Prompt the user re-enter their card if required.
Return to the application with correct insertion or error.

The format of the command is as follows.

Command - X'20'

Parameter 1 - Card Type

- X'01' - Magnetic Stripe Card
- X'02' - Smart Card
- X'03' - Magnetic Stripe Card and Smart Card

Parameter 2 - Audible Prompt

- X'00' - No audible prompt
- X'01' - Audible prompt

The following messages may be displayed with this command.

Message 1
- Line 0 - Please
- Line 1 - insert your card

Message 2
- Line 0 - Please
- Line 1 - remove your card

Message 3
- Line 0 - Please re-insert
- Line 1 - your card

Message 4
- Line 0 - Please check card
- Line 1 - for correct
- Line 2 - orientation

Message 5
- Line 1 - Card insertion error

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'0B' - Timeout
- X'0E' - Feature Not Available

A timeout can be defined to limit the amount of time the device will wait for a card to be inserted. For a description of this timeout parameter and its use, see “Load Device Parameters” on page 4-35.

Execute Utility Functions

This command is used to execute functions that enhance programmability of the 4779 device. This command will execute one of the following depending upon the command parameter;

issue an audible prompt
define the cursor control mode
load a user defined character

The audible prompt produces a beep from the device the duration of which is defined by a two-byte hex value. The value is specified as the number of milliseconds the audible prompt will sound. The maximum value is 200 milliseconds (i.e. X'00C8').

The cursor control mode defines the presentation of the cursor on the display. The following four modes may be selected.

cursor off and no blink
cursor on and no blink
cursor off and blink
cursor on and blink

The cursor mode selected with this command will be in effect until the device is reset or redefined with this command.

The display character definition function defines the placement of dots in a 5x8 matrix to form a character. Up to eight separate characters may be defined and are distinguished by a character identification number. Eight bytes of data are used to define the dot placement per row of the matrix. The display character defined by this command will be in effect until the device is reset or redefined with this command.

The format of the command is as follows.

Command - X'11'

Parameter ID - one byte

- X'00' for audible prompt
- X'01' for cursor mode control
- X'02' for display character definition
- All other values are reserved for future use.

Parameter values. The length and meaning of this value are dependent on the parameter being loaded.

- For parameter ID X'00', the parameter value is a two-byte integer defining the beep duration in milliseconds. The range of accepted values is X'000A' (10 decimal) through X'00C8' (200 decimal). The default value is 200 milliseconds.
- For parameter ID X'01', the parameter value is a one-byte integer defining the cursor control mode.
  - X'00' - With this option, the cursor is not visible.
  - X'01' - With this option the cursor (character 5Fh) is displayed. Refer to characters as described in Appendix G, “Character Font Table” on page G-1.

- X'02' - With this option the cursor is not visible but the character FFh blinks. Refer to characters as described in Appendix G, “Character Font Table” on page G-1.
- X'03' - With this option the cursor (character 5Fh) is displayed and the character FFh blinks. Refer to characters as described in Appendix G, “Character Font Table” on page G-1.

– For parameter ID X'02', the nine bytes of data required for this function are specified as the character identification and definition data.
  - Character ID - 1 byte X'00' - X'07'
  - Definition Data - 8 bytes define the dot locations on a 5 x 8 matrix as follows.

    Each byte of data defines the dots activated for each row. The following example shows the definition of bytes to define an "up arrow".

Matrix Row	b7	b6	b5	b4	b3	b2	b1	b0
0	X	X	X	0	0	1	0	0
1	X	X	X	0	1	1	1	0
2	X	X	X	1	0	1	0	1
3	X	X	X	0	0	1	0	0
4	X	X	X	0	0	1	0	0
5	X	X	X	0	0	1	0	0
6	X	X	X	0	0	1	0	0
7	X	X	X	0	0	0	0	0

    where ***X*** is ignored, ***0*** is a dot that is to be off, and *1* is a dot that is to be on.

    The definition data for this example would be X'040E150404040400'

The format of the response is as follows.

Expected Return Codes

– X'00' - No Error
– X'02' - Data Length Error
– X'04' - Invalid Value

Load Device Parameters

This command is used to load operating parameters into the 4779. Four parameters may be loaded using this command, the keypad timeout value, the card insertion timeout value, the keypad abort key and the country code. Loading parameters initializes global variables that are in effect until the device is reset. Therefore, it is required to issue these commands only once when the PC application program is initialized.

The keypad timeout value is a two-byte hex value that defines the number of seconds the 4779 should wait for the user to press a key each time it is waiting for keyboard input. If the timeout period expires before the next key is pressed, the function will abort and return the timeout error code. The timeout is restarted each time the user presses a key. Keypad timeout value X'0000' has a special meaning. The value of zero tells the 4779 not to use a timeout when waiting for keystrokes; it will wait indefinitely. The maximum value allowed for the timeout is 600 (i.e. X'0258') seconds, or 10 minutes.

The card insertion timeout value is a two-byte hex value that defines the number of seconds the 4779 should wait for the user to insert or remove a card from the device. If the timeout period expires before insertion or removal, the function will abort and return the timeout error code. A timeout value X'0000' has a special meaning. The value of zero tells the 4779 not to use a timeout when waiting for a card event; it will wait indefinitely. The maximum value allowed for the timeout is 600 (i.e. X'0258') seconds, or 10 minutes.

The keypad abort key value is a one byte hex value that defines the key that the user may use to abort a keypad operation.

The country code is used to display application messages for the commands Generate Offset (IBM 3624) Comprehensive, Verify PIN (IBM 3624) Comprehensive, and Manage Card Insertion in the desired language. There are five country options, English, German, French, French Canadian and Spanish. When the 4779 device is powered on, the messages will default to English.

The format of the command is as follows.

Command - X'0D'

Parameter ID - one byte

- X'00' for the keypad timeout
- X'01' for the card insertion timeout
- X'02' for the keypad abort key definition
- X'03' for the country code
- All other values are reserved for future use.

Parameter values. The length and meaning of this value are dependent on the parameter being loaded.

- For parameter ID X'00', the parameter value is a two-byte integer defining the keypad timeout. The range of accepted values is X'0000' through X'0258' (600 decimal). X'0000' indicates that no timeout will be used. Values X'0001' through X'0258' are the number of seconds to wait for a keystroke before timing out. The default value in the 4779 is 60 (X'003C') seconds.
- For parameter ID X'01', the parameter value is a two-byte integer defining the card insertion or removal timeout. The range of accepted values is X'0001' through X'0258' (600 decimal). X'0000' indicates that no timeout will be used. Values X'0000' through X'0258' are the number of seconds to wait for a card event to take place. The default value is 60 (X'003C') seconds. This parameter applies only when using the commands Manage Card Insertion, Generate Offset (IBM 3624) Comprehensive, or Verify PIN (IBM 3624) Comprehensive.
- For parameter ID X'02', the parameter value is a one byte value defining the keypad key that is to be used to abort a keypad operation. The available keys and their associated values are listed in the table below. The default for this parameter is that no key is defined for the purpose of aborting a keypad operation. To define the default after an abort key has been established with this command, use the ASCII code of 00H.

Keycap	ASCII Code
0	30H
1	31H
2	32H

Keycap	ASCII Code
3	33H
4	34H
5	35H
6	36H
7	37H
8	38H
9	39H
F1	41H
F2	42H
F3	43H
F4	44H
Default	00H

– For parameter ID X'03', the parameter value is a one byte value that designates the particular country as indicated.
  - X'01' - English
  - X'02' - German
  - X'03' - French
  - X'04' - French Canadian
  - X'05' - Spanish

The format of the response is as follows.

Expected Return Codes

– X'00' - No Error
– X'02' - Data Length Error
– X'04' - Invalid Value
– X'0E' - Feature Not Available

Load Verification Parameters

This command is used to load operating parameters into the 4779 that will be used with the Generate Offset (IBM 3624), Generate Offset (IBM 3624) Parameter Version, Generate Offset (IBM 3624) Comprehensive, Verify PIN (IBM 3624), Verify PIN (IBM 3624) Parameter Version, and Verify PIN (IBM 3624) Comprehensive commands. Four parameters categories may be loaded with this command, general verification parameters, a decimalization table, validation parameters, and offset parameters. Loading parameters initializes global variables that are in effect until the device is reset. Therefore, it is required to issue these commands only once when the PC application program is initialized.

The general verification parameters include the key indicator, the PIN length and the PIN entry type. The decimalization table specifies a 16 byte array that is used to convert encrypted data from hexadecimal to decimal. The validation parameters include the track from which the validation data is to be read, the validation length, the location of this data and the pad character. The offset parameters include the track on which the offset is to be read and or encoded, the pad character, the offset length for reading or encoding, and the location of the offset field on the track for reading and encoding.

The format of the command is as follows.

Command - X'10'

Parameter ID - one byte

– X'00' for the general verification parameters
– X'01' for the decimalization table
– X'02' for the validation parameters
– X'03' for the offset parameters
– All other values are reserved for future use.

Parameter values. on the parameter being loaded.

– For parameter ID X'00', general verification parameters. The parameter values and their defaults are as follows.

  - Key Indicator - 1 Byte - This value specifies the DES data key number to be used in the encryption of the validation data. The default key is X'00'.

    X'00' - X'07'

  - PIN Length - 1 Byte - This value specifies the number of offset digits to be created when executing a generate offset command or the number of PIN digits to verify when executing a PIN verification command. This value must be equal to or less than the number of entered PIN digits. The length may range from 1 to 16 digits. The default length is 4.

    X'01' - X'10'

  - PIN Entry Type - 1 Byte - This value specifies if single or dual PIN entry is required. The default is single entry of the PIN.

    Single PIN Entry - X'00'
    Dual PIN Entry - X'01'

    To initialize the general verification parameters to the default values, the following command would be sent by the PC application.
    X'1000000400'

– For parameter ID X'01', decimalization table. The parameter value and default are as follows.

  - Decimalization Table - 16 Bytes - This parameter specifies an array of 16 values that are used to convert the encrypted validation data from hexadecimal to decimal. The default is as follows.

Position	Table Value
0	00H
1	01H
2	02H
3	03H
4	04H
5	05H
6	06H
7	07H
8	08H
9	09H
A	00H
B	01H
C	02H
D	03H
E	04H
F	05H

To initialize the decimalization table to the default values, the following command would be sent by the PC application.
X'10010001020304050607080900010203040 5'

– For parameter ID X'02', validation parameters. The parameter values and their defaults are as follows.

  - Validation Data Track - 1 Byte - This value specifies the magnetic stripe track which has the validation data. The default track is track two.

    Bit 7 - reserved
    Bit 6 - reserved
    Bit 5 - reserved
    Bit 4 - reserved
    Bit 3 - reserved
    Bit 2 - Track 3
    Bit 1 - Track 2
    Bit 0 - Track 1

  - Validation Data Length - 1 Byte - This value specifies the length of the validation data. The default length is 16 (X'10') bytes.

  - Validation Data Offset - 1 Byte - This value specifies the start location of the validation data field from the first data position on the track. The first data position is designated as zero for the purpose of defining this parameter. The default offset is X'00'.

  - Validation Data Pad Character - 1 Byte - This value indicates the pad character to be used for encryption when the validation data is less than 16 bytes. It is specified as X'0x'. The default pad character is X'00'. To specify the character 'F' as the pad character, this parameter would be X'0F'.

    To initialize the validation parameters to the default values, the following command would be sent by the PC application.
    X'100202100000'

– For parameter ID X'03', offset parameters. The parameter values and their defaults are as follows.

  - Offset Data Track for Reading - 1 Byte - This value specifies the magnetic stripe track from which the offset data is to be read. The default is track two.

    Bit 7 - reserved
    Bit 6 - reserved
    Bit 5 - reserved
    Bit 4 - reserved
    Bit 3 - reserved
    Bit 2 - Track 3
    Bit 1 - Track 2
    Bit 0 - Track 1

  - Offset Data Track for Encoding - 1 Byte - This value specifies the magnetic stripe track to which the offset data may be encoded. The default is track two.

    Bit 7 - reserved
    Bit 6 - reserved
    Bit 5 - reserved

Bit 4 - reserved
Bit 3 - reserved
Bit 2 - reserved
Bit 1 - Track 2
Bit 0 - reserved

- Offset Data Pad Character - 1 Byte - This value specifies the pad character to be used when the offset is created. The default for this parameter is the X'00'. When commands Generate Offset (IBM 3624) or Generate Offset (IBM 3624) Parameter Version are used, the pad character returned to the application is the byte specified by this parameter. For example, if X'FF' is specified, 'FF' will be returned as the pad character. When the command Generate Offset (IBM 3624) Comprehensive is used, the pad character is returned as an ASCII character since it is being encoded. For instance, X'01' is returned as '31'. In this case the pad character is limited the numerics values, 0-9.
- Offset Data Offset for Reading - 1 Byte - This value specifies the start location of the offset data field from the first data position on the track. The first data position is designated as zero for the purpose of defining this parameter. The default for this parameter is X'00'.
- Offset Data Offset for Encoding - 1 Byte - This value specifies the start location of the encoded offset field from the first data position on the track. The first data position is designated as zero for the purpose of defining this parameter. The default for this parameter is the first data position after the existing track data. This default is designated by X'FF'.
- Offset Data Length for Reading - This value specifies the length of the offset to be read. The default is the PIN length of 4 (X'04'). A change to the PIN length parameter will automatically be reflected in the offset data length for reading parameter. Use of this parameter will allow an offset length to be read that is different from the PIN length.
- Offset Data Length for Encoding - This value specifies the length of the offset to be encoded. The default is 16 (X'10') characters.

To initialize the offset parameters to the default values, the following command would be sent by the PC application. X'100302020000FF0410'

The format of the response is as follows.

– Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'0E' - Feature Not Available

Read Machine Information Data Structure

This command is used to obtain information describing the hardware and BIOS features of the 4779 device. A command parameter is used to obtain information related to a specific device category. Information may be accessed from the categories that include the general device, magnetic stripe, keypad, smart card reader, display, communications, version and model.

The machine information is separated into functional categories, where each category is given a separate tag. Please refer to Appendix F, “Machine Information Data Structure” on page F-1 for a detailed description of the machine information data structure. A brief description of each category is provided below.

The format of the command is as follows.

Command - X'0E'

Parameter value - 1 byte

– X'01' All device information

  This category will return all device information described by tags X'02' to X'09'.

– X'02' General device information

  This category describes general features of the 4779, which do not pertain to any particular component or subsystem.

– X'03' Magnetic stripe information

  The magnetic stripe information describes features involved in reading and writing magnetic stripe cards.

– X'04' Keypad information

  The keypad information indicates whether the 4779 has an integrated keypad, and if so it provides some information about the available keys.

– X'05' Smart card information

  This category indicates whether the device includes a smart card reader, and if so it provides information on the types of smart cards supported.

– X'06' Display information

  The display information indicates whether the device includes a display, and if so it provides information about the features of that display.

– X'07' Communication interface information

  This category contains information about the interface the 4779 can use to communicate with other devices, such as a PC. Note that many applications call for a stand-alone device, with no communications.

– X'08' BIOS version information

  This information identifies the version and build date of the resident BIOS. The data enclosed in this object consists of a 20 character string containing the version number and build date in the form "VER: vv.vv, mm/dd/yy".

– X'09' 4779 model type information

  This information identifies the model of 4779, model numbers may be 1 or 2.

– All other values are reserved for future use.

The format of the response is as follows.

Expected Return Codes

– X'00' - No Error
– X'02' - Data Length Error
– X'04' - Invalid Value

Data and length based upon the parameter value.

Abort

This command is used to abort keypad operations from the host application. This command is only applicable for use with the DOS operating system. To terminate keypad operations in the OS/2 environment, use the EFTAbort function described in chapter 3.

When issued, this command will abort the current process and pass a return code to the host indicating that the abort was successful. If an abort is received and no process is in progress, an out of sequence error will be returned. This command will perform an abort of keypad operation associated with the with Read Keypad, Smart Card Password Verification (SAISS), Smart Card Password Verification (MFC), Generate PIN Block (ANSI X9.8), Generate PIN Block (ANSI X9.8) Parameter Version, Generate PIN Block (IBM 3624), Generate Offset (IBM 3624), Generate Offset (IBM 3624) Parameter Version, Generate Offset (IBM 3624) Comprehensive, Verify PIN (IBM 3624), Verify PIN (IBM 3624) Parameter Version and Verify PIN (IBM 3624) Comprehensive commands.

The format of the command is as follows.

Command - X'F2'

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'06' - Sequence Error
- X'0E' - Feature Not Available

Reset Device

This command is used reinitialize the device application from the PC. There is no response to this command. Attempting to read a response after this command is issued is not recommended.

The format of the command is as follows.

Command - X'F3'

The format of the response is as follows.

This command returns no response.

Reset Security Function

This command is used to reinitialize the security function from the PC application for 4779 devices with the Enhanced Security Feature. If a security feature command returns a sequence error (X'06'), executing this command eliminates the need to reset the device. To determine whether a 4779 device has the Enhanced Security Feature, the "Read Machine Information Data Structure" command can be issued.

The format of the command is as follows.

Command - X'F4'

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'0E' - Feature Not Available

Query CRC of Device Memory

This command is used to obtain a CRC of the loaded device memory. This would be done to verify the integrity of the memory by a comparison with a previously obtained CRC. The CRC evaluation starts at the memory address specified by the first parameter and terminates when the number of bytes specified in the second parameter have been evaluated.

The format of the command is as follows.

Command - X'F5'

Beginning Address - X'xxxx' - two bytes - This parameter specifies the beginning memory address at which the CRC evalution will start.

Evaluation Length - X'xxxx' - two bytes - This parameter specifies the length of memory included in the CRC evaluation. The length value of X'xx00' is invalid.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value

CRC Value - 2 Bytes

Chapter 5. 4779 Security Functions

Base Security Function versus Enhanced Security Function

Models of the 4779 that contain the base security function support DES key loading and importation, MAC generation and verification, key verification, and random number generation. The keys are stored in non-volatile memory within the device. These keys are protected by intrusion circuitry. Furthermore, there is neither a public BIOS interface nor an interface provided by the IBM device application (see Chapter 4, “4779 Device Resident Application Function Definitions”) available to retrieve these keys. However, since the keys are stored in addressable non-volatile memory, a special option has been provided to prevent a rogue device application from being downloaded to perform a "memory dump" of all addressable memory upstream to the host for a hacker program to find the keys. This special option, termed *Key Reset Between Application Loads*, is a byte of data contained within the command format for all security functions that pertain to the loading of DES keys. When this byte is X'01', the key is erased by BIOS whenever a new application is being downloaded. If a rogue device application that may perform addressable memory dumps is not a concern, the *Key Reset Between Application Loads* byte should be set to X'00' which informs BIOS ***not*** to erase the key whenever a new application is being downloaded.

Models of the 4779 that contain the enhanced security function support all the functions of the base security function and in addition, support loading of an RSA private key, one time only, and importation of DES keys under an RSA public key. Furthermore, the keys are stored in ***non***-addressable non-volatile memory. Thus, a rogue device application downloaded into the 4779 cannot obtain the keys from a memory dump of addressable non-volatile memory. For this reason, the *Key Reset Between Application Loads* option byte is not supported in models containing the enhanced security function. If this option byte is present in the command format for functions that pertain to the loading of DES keys, the enhanced security function will ignore the byte.

The device application can determine whether a given 4779 contains base security or enhanced security by querying the Machine Information Data Structure, Structure Tag X'02' described in Appendix F, “Machine Information Data Structure” on page F-1.

Security Function Interface Buffer

The size of the security function interface buffer for models of the 4779 containing either base security or enhanced security is limited to 155 bytes maximum. Only the Generate Mac and Verify Mac commands (X'8D' and X'8E' respectively) can potentially use this many bytes per command.

Clearing Keys

Clearing the stored keys for models with either base security or enhanced security can be achieved by loading a key value set to all zeroes or some other value. However, once a key is loaded, the key will be adjusted to have odd parity where the least significant bit of each byte of the key is the parity bit. Thus, loading a key of all zeroes will be deemed good and usable by the device. This may be acceptable if the intent is to erase a previously loaded key. The onus of key

management resides with the application in the PC system unit that uses the 4779 device driver to communicate to the 4779 device.

Key Terminology

A 4779 with base security can store a total of 16 keys whereas a 4779 with enhanced security can store a total of 17 keys as defined below:

8 of the keys are referred to as *Key Encrypting Keys*, or *KEK*s, which are 16 bytes in length. These keys can be loaded in clear form (see “Load Cleartext Double Length DES Key Part” on page 5-5) or encrypted form. *KEK*s that are loaded in clear form are often referred to as *Master Keys*. Typically there is only one Master Key. The remaining seven KEKs may be loaded in clear form but are typically imported as keys which are either encrypted:

- under the RSA Public Key (see “Import Double Length DES Key under RSA Public Key” on page 5-6). This option is only supported by models of the 4779 containing enhanced security.
- under another KEK, or Master Key (see “Import Double Length DES Key under a DES Key Encrypting Key” on page 5-7).
- or by the IBM CCA compatible KEK with Control Vector (see “Import Double Length DES Key with Control Vector” on page 5-8).

KEKs imported in encrypted form are decrypted or *recovered* and stored in clear form in a KEK table at the index specified by the *Key Number* command parameter.

8 of the keys are referred to as *Data Keys* which are 8 bytes in length. These keys are imported triple encrypted under a previously loaded KEK. (See “Import Single Length DES Data Key” on page 5-9). Once imported, these keys are decrypted and stored in clear form in a Data Key table at the index specified by the *Key Number* command parameter.

A 4779 that contains the enhanced security function can store 1 extra key called the RSA Private Key. This key is not supported by models of the 4779 that contain the base security function. The RSA Private Key can only be loaded once. (See “Load Cleartext RSA Private Key” on page 5-4 for details).

4779 Security Function Calls

The 4779 security function definitions are a set of function calls that allow the programmer access to the security functionality of the 4779 device. The security functions may be called by a 4779 PC host application or by the 4779 device resident application program. The remaining sections of this chapter describe the 4779 security functions listed below:

Figure 5-1. 4779 Security Function Command Codes

Command	Description
X'80'	"Read Serial Number" on page 5-3
X'81'	"Generate Random Number" on page 5-4
X'82'	"Load Cleartext RSA Private Key" on page 5-4
X'83'	"Load Cleartext Double Length DES Key Part" on page 5-5
X'84'	"Import Double Length DES Key under RSA Public Key" on page 5-6
X'85'	"Import Double Length DES Key under a DES Key Encrypting Key" on page 5-7
X'86'	"Import Double Length DES Key with Control Vector" on page 5-8
X'87'	"Import Single Length DES Data Key" on page 5-9
X'8D'	"Generate MAC" on page 5-10
X'8E'	"Verify MAC" on page 5-14
X'94'	"Compute Verification Pattern" on page 5-18
X'95'	"Verify Key" on page 5-19

The 4779 security function calls use the same return codes as the 4779 device resident application function calls. For a complete list and description of these return codes, see Appendix B, "4779 Device Return Codes" on page B-1.

For examples of how to invoke the 4779 security functions in a C Language PC host application, refer to the source files for the 4779 Sample Application Program which may be found on the 4779 Device Support Code Diskette.

Read Serial Number

This command is used to obtain the serial number stored in the security area non-volatile memory. The format of the command is as follows -

Command - X'80'

The format of the response is as follows -

Expected Return Codes

- X'00' - No Error
- X'0A' - Security Error

Data -

- Serial Number - 8 Bytes of ASCII characters

Generate Random Number

This command is used to obtain an 8 byte random number generated by the security function.

The random number can be in one of two forms.

64 fully random bits.

A parity-adjusted DES key, in which the least significant bit of each byte has been given a value that will result in odd parity for the byte.

The format of the command is as follows.

Command - X'81'

Random number form

- X'00' - return a fully random 64 bit number.
- X'01' - return a value that has been parity-adjusted as a DES key.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'0A' - Security Error

Data -

- Random Number - 8 Bytes

Load Cleartext RSA Private Key

Note: This command is supported by the Enhanced Security Feature only. An attempt to use this command in a non-enhanced security function machine will result in a return code of "unimplemented command" (X'03').

This command is used to load an RSA Private Key into the 4779. This key is used to securely load DES operational and Key Encrypting Keys. The key information passed to the 4779 is a subset of the information normally available in the IBM Common Cryptographic Architecture defined Cryptographic Facility Public Key Record (CFPKR). The CFPKR layout is described in the Transaction Security System Programming Reference: Volume I, Access Controls and DES Cryptography, cited in “References” on page xii.

For the 4779, the modulus is of length 32 words (512 bits). The exponent is also of length 32 words (512 bits). Any other length specified for either the modulus or exponent will cause a data length error (X'02') return code. The 4779 device microcode allows the RSA Private Key to be loaded once.

The format of the command is as follows.

Command - X'82'

Length of Modulus in Words - X'20'

Length of Exponent in words - X'20'

The Value of the Modulus - X'40' bytes in length where the most significant bit of the modulus is located in the left-most position of the first byte.

The Value of the Exponent - X'40' bytes in length where the most significant bit of the exponent is located in the left-most position of the first byte.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'03' - Unimplemented Command (if device does not contain the enhanced security function).
- X'06' - Sequence Error
- X'08' - Bad Key
- X'0A' - Security Error

Load Cleartext Double Length DES Key Part

This command is used to load a double length DES Key Encrypting Key to the 4779 in clear form. In order to achieve multiple levels of control, the key is loaded in parts which are combined using the XOR function.

When the last key part is loaded, it is adjusted to have correct DES key parity.

The format of the command for models containing the enhanced security function is as follows:

Command - X'83'

Key Number X'00' to X'07'

Part Indicator

- X'00' - First
- X'01' - Middle
- X'02' - Last

Key part - 16 Bytes

The format of the command for models containing the base security function is as follows:

Command - X'83'

Key Number X'00' to X'07'

Part Indicator

- X'00' - First
- X'01' - Middle
- X'02' - Last

Key part - 16 Bytes

Key Reset Between Application Loads byte - required by models with the base security function only. The enhanced security function will ignore this byte if present. Possible values include:

- X'00' - Do not clear the key during the next device application download.
- X'01' - Clear the key during the next device application download.

If this byte is not present, the base security function will assume a value of X'01' and clear the key during the next device application download.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'06' - Sequence Error
- X'0A' - Security Error

Import Double Length DES Key under RSA Public Key

Note: This command is supported by the enhanced security function only. An attempt to use this command in a non-enhanced security function machine will result in a return code of "unimplemented command" (X'03').

This command is used to import a double length DES Key Encrypting Key to the 4779 device, encrypted under the RSA Public Key corresponding to the RSA Private Key loaded in the device. The DES KEK is decrypted using the RSA Private Key. The resulting decrypted DES KEK is stored in the KEK table. The key is adjusted to have correct DES key parity before it is stored in the device's KEK table.

The format of the command is as follows.

Command - X'84'

Key Number - X'00' to X'07'

Reserved - X'00'

Encrypted key - 64 bytes, as shown in Figure 5-2 below before RSA encryption. The most significant byte resides in the left-most position in the figure.

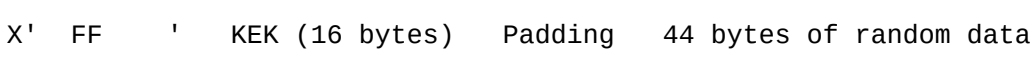


Figure 5-2. Format of key encrypting key buffer before RSA encryption

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'03' - Unimplemented Command (if device does not contain the enhanced security function)
- X'04' - Invalid Value
- X'0A' - Security Error

Import Double Length DES Key under a DES Key Encrypting Key

This command is used to import a double length DES Key Encrypting Key to the 4779 device, encrypted under a double length DES key which is already resident in the device. The key is passed to the security function with each half triple-encrypted under a second Key Encrypting Key. The triple-encryption process is described in Appendix D, “Triple Encryption Algorithm Without Control Vector” on page D-1. Inputs to this command are the encrypted key (2 halves), the KEK table index where the recovered key is to be stored, and the key ID of the KEK that the new key is encrypted under. The key is adjusted to have correct DES key parity before it is stored in the device's KEK table.

The format of the command for models containing the enhanced security function is as follows:

Command - X'85'

Key Encrypting Key number X'00' to X'07' - (used to recover input encrypted key)

Key Number X'00' to X'07' - (index of key table corresponding to where the recovered key is to be stored)

Encrypted key - 16 Bytes

The format of the command for models containing the base security function is as follows:

Command - X'85'

Key Encrypting Key number X'00' to X'07' - (used to recover input encrypted key)

Key Number X'00' to X'07' - (index of key table corresponding to where the recovered key is to be stored)

Encrypted key - 16 Bytes

Key Reset Between Application Loads byte - required by models with the base security function only. The enhanced security function will ignore this byte if present. Possible values include:

- X'00' - Do not clear the key during the next device application download.
- X'01' - Clear the key during the next device application download.

If this byte is not present, the base security function will assume a value of X'01' and clear the key during the next device application download.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'0A' - Security Error

Import Double Length DES Key with Control Vector

This command is used to import a double length DES Key previously encrypted by the supplied IBM CCA compatible Key Encrypting Key with Control Vector (CV). The CV is not checked by the device but is used in the key recovery process.

The KEK is passed to this command with each half triple encrypted under a second KEK using a control vector. This form of triple-encryption is written as e*KEK1.CV(KEK2), where KEK2 is the key being imported and KEK1 is the key it is encrypted under. Each half of the new key is independently triple-encrypted using the algorithm described in Appendix C, “Triple Encryption Algorithm With Control Vector” on page C-1.

Inputs to the command are the encrypted key (2 halves), the KEK table index where the recovered key is to be stored, the key ID of the KEK that the new key is encrypted under, and the control vector. The key is adjusted to have correct DES key parity before it is stored in the device's KEK table.

The format of the command on models containing the enhanced security function is as follows:

Command - X'86'

Key Encrypting Key number X'00 to X'07' - (used to recover input encrypted key)

Key Number X'00' to X'07' - (index of key table corresponding to where the recovered key is to be stored)

Encrypted key - 16 Bytes

Control Vector - 16 Bytes

The format of the command on models containing the base security function is as follows:

Command - X'86'

Key Encrypting Key number X'00 to X'07' - (used to recover input encrypted key)

Key Number X'00' to X'07' - (index of key table corresponding to where the recovered key is to be stored)

Encrypted key - 16 Bytes

Control Vector - 16 Bytes

Key Reset Between Application Loads byte - required by models with the base security function only. The enhanced security function will ignore this byte if present. Possible values include:

- X'00' - Do not clear the key during the next device application download.
- X'01' - Clear the key during the next device application download.

If this byte is not present, the base security function will assume a value of X'01' and clear the key during the next device application download.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'0A' - Security Error

Import Single Length DES Data Key

This command is used to import a single length DES Data Key. The key is passed triple-encrypted under a Key Encrypting Key. Inputs to the command are the encrypted key, the key ID where the recovered key is to be stored in the table, and the key ID of the KEK that it is encrypted under. The key is adjusted to have correct DES key parity before it is stored in the device's Data Key table.

The format of the command on models containing the enhanced security function is as follows:

Command - X'87'

Key Encrypting Key number X'00 to X'07' - (used to recover input encrypted key)

Key Number X'00' to X'07' - (index of key table corresponding to where the recovered key is to be stored)

Encrypted key - 8 Bytes

The format of the command on models containing the base security function is as follows:

Command - X'87'

Key Encrypting Key number X'00 to X'07' - (used to recover input encrypted key)

Key Number X'00' to X'07' - (index of key table corresponding to where the recovered key is to be stored)

Encrypted key - 8 Bytes

Key Reset Between Application Loads byte - required by models with the base security function only. The enhanced security function will ignore this byte if present. Possible values include:

- X'00' - Do not clear the key during the next device application download.
- X'01' - Clear the key during the next device application download.

If this byte is not present, the base security function will assume a value of X'01' and clear the key during the next device application download.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'0A' - Security Error

Generate MAC

This command is used to compute a Message Authentication Code (MAC) on a string of message text. The length of the message text must be a multiple of eight bytes; any padding that is required must be done prior to sending the *Generate MAC* request.

The MAC computation employs the DES algorithm, using a data key stored in the 4779 internal Data Key table. The key ID number is passed as one of the command parameters.

The command is either processes as an only part, or processed in three parts which must be issued in order. The three parts of the latter option are:

First part The first message contains setup information to be used in MAC processing. The message contains two elements:

The key ID number, referencing one of the data keys stored in the device to be used to calculate the MAC.
The ICV, the DES Initial Chaining Vector.

These two values are stored in the device for use during the MAC calculation.

There is no response data, other than the return code.

Middle part(s) These messages contain data used as input to the MAC calculation. Each message must contain data that is an even multiple of eight bytes in length. Any number of eight byte blocks can be included in a message, up to the limit imposed by the size of the security function interface buffer which is currently a maximum of nineteen 8-byte blocks.

The middle part is not required, if the message is short enough to fit entirely in the message for the last part.

There is no response data, other than the return code.

Last part The last message in the sequence carries MAC input message text, just like the middle messages. The only difference is that the MAC result is returned when the last part has been processed.

Alternately, the command can be processed in a single, or only part:

Only part An "only" part message interface is provided for those messages that are short enough to fit entirely in one part only. These messages contains setup information to be used in MAC processing just like the first part as well as data that is in an even multiple of eight bytes in length. The message contains the following three elements:

The key ID number, referencing one of the data keys stored in the device to be used to calculate the MAC.

The ICV, the DES Initial Chaining Vector (8 bytes).

Any number of eight byte blocks up to the limit imposed by the size of the security function interface buffer. The maximum buffer size of 19 eight-byte blocks can be utilized as follows:

– One 8-byte block for the ICV

– Between 1 and 18 eight-byte blocks for data

The MAC result is returned when the only part has been processed.

Each message contains a one-byte field called the *Sequence indicator*, which identifies the message as a first, middle, last, or only part.

The format of the command is as follows.

Command - X'8D'

Sequence indicator

– X'00' for the first part
– X'01' for the second part(s)
– X'02' for the last part
– X'03' for an only part

Key Number - use depends on the sequence indicator

For first part: ID of data key to use, X'00' to X'07'

For middle or last part: Unused - set to X'00'.

For only part: ID of data key to use, X'00' to X'07'

Data - use depends on the sequence indicator

For first part: This field contains the eight byte ICV (Initial Chaining Vector).

For middle or last part: This field contains between 1 to 19 eight-byte blocks of input message text.

For only part: This field contains the eight byte ICV (Initial Chaining Vector) immediately followed by between 1 to 18 eight-byte blocks of input message text.

The format of the response is as follows.

Expected Return Codes

– X'00' - No Error
– X'02' - Data Length Error
– X'04' - Invalid Value
– X'06' - Sequence Error
– X'08' - Bad Key
– X'0A' - Security Error

Data - use depends on the sequence indicator

For first or middle part: Unused - no data is returned.

For last part: The data field contains the eight byte MAC calculated on the input message text.

For only part: The data field contains the eight byte MAC calculated on the input message text.

Sequence Errors: Sequence error return codes will be generated for the following scenarios:

Middle part before First part
Last part before First part

First part after Middle part
First part after First part
Only part after First part
Only part after Middle part

In the event that a PC host application is terminated while the 4779 is processing a first part or middle part, then subsequently restarted, the PC and 4779 will lose part-sequence synchronization. A newly started PC application that sends a first part when the 4779 expects a middle or last part from the previously terminated application will receive a sequence error from the device. If this occurs, the application should send a Reset Security Function (X'F4') command to the 4779 to force a resynchronization.

Example: Here is an example of the *Generate MAC* message sequence. It shows three request/response message pairs for a first, middle, and last part. Alternately, an only part can be used which is shown following the last part example.

The data used in this example is as follows:

Figure 5-3. MAC example: input values

Item	Value	Description
Key	X' 2468ACE13579BDF'	The DES key used to compute the MAC. In this example, the key is in data key register number 0.
ICV	X' 123456789ABCDEF'	The initial chaining vector
Message text	X'1234123412341234' X'56789ABCDEF 1234' X'FEDCBA987654321 ' X'FFEEDDCCBBAA9988' X'77665544332211 '	The input message text on which the MAC is to be calculated.

```
First message  (First part)
                                                  Command ID

                                                 Sequence indicator

                                                Key ID

                                               ICV


   8D              123456789ABCDEF


                                                  Return code
```

```
Second message  (Middle part)
                                                Input message text

    8D    1        12341234123412345678 9ABCDEF 1234FEDCBA987654321



                                                  Return code

Third message  (Last part)

    8D    2        FFEEDDCCBBAA998877665544332211

         2FEFE FFE  9BB7B

                                                MAC result
                                                 Return code

Only message  (Only part)
                                                 Command ID

                                                 Sequence indicator

                                                 Key ID

                                                ICV

    8D    3         123456789ABCDEF

                  1234123412341234
                  56789ABCDEF 1234          Input message text
                  FEDCBA987654321
                  FFEEDDCCBBAA9988
                  77665544332211


         2FEFE FFE  9BB7B

                                                MAC result
```

Verify MAC

This command is used to verify a Message Authentication Code (MAC) for a string of message text. The length of the message text must be a multiple of eight bytes; any padding that is required must be done prior to sending the *Verify MAC* request.

The MAC computation employs the DES algorithm, using a data key stored in the 4779 internal Data Key table. The key ID number is passed as one of the command parameters.

The command is either processed in three parts which must be issued in order, or as an only part. In the former option, the three parts are:

First part The first message contains setup information to be used in MAC processing. The message contains two elements:

The key ID number, referencing one of the data keys stored in the device.
The ICV, the DES Initial Chaining Vector.

These two values are stored in the device for use during the MAC calculation.

There is no response data, other than the return code.

Middle part(s) These messages contain data used as input to the MAC calculation. Each message must contain data that is an even multiple of eight bytes in length. Any number of eight byte blocks can be included in a message, up to the limit imposed by the size of the security function interface buffer which is currently a maximum of 19 8-byte blocks.

There is no response data, other than the return code.

Last part The last message in the sequence contains the MAC value which is compared with the MAC calculated on the message text. The MAC may be any length between 4 and 8 bytes; the value will be compared with an equal length of the computed MAC, starting at the most significant byte.

The response is a one-byte boolean value, indicating whether the MAC provided in the command matches the MAC calculated over the supplied message text.

`X'01'` if the MAC values match (MAC verifies).

`X'00'` if the MAC values are different (MAC does not verify).

Alternately, the command can be processed in a single, or only, part:

Only part An "only" part message interface is provided for those messages that are short enough to fit entirely in one part only. These messages contains setup information to be used in MAC processing just like the first part as well as data that is in an even multiple of eight bytes in length and a MAC value between 4 and 8 bytes for comparison. The message contains the following elements:

The key ID number, referencing one of the data keys stored in the device to be used to calculate the MAC.

The ICV, the DES Initial Chaining Vector (8 bytes).

Any number of eight byte blocks up to the limit imposed by the size of the security function interface buffer. The maximum buffer size of 19 eight-byte blocks can be utilized as follows:

- One 8-byte block for the ICV
- Between 1 and 17 eight-byte blocks for data
- Between 4 and 8 bytes for the MAC value which is compared with the MAC calculated on the message text. The value will be compared with an equal length of the computed MAC, starting at the most significant byte.

  The response is a one-byte boolean value, indicating whether the MAC provided in the command matches the MAC calculated over the supplied message text.

  - X'01' if the MAC values match (MAC verifies).
  - X'00' if the MAC values are different (MAC does not verify).

Each message contains a one-byte field called the *Sequence indicator*, which identifies the message as a first, middle, last, or only part.

The format of the command is as follows.

Command - X'8E'

Sequence indicator

- X'00' for the first part
- X'01' for the second part(s)
- X'02' for the last part
- X'03' for an only part

Key Number - use depends on the sequence indicator

For first part: X'00' to X'07'

For middle or last part: Unused - set to X'00'.

For only part: X'00' to X'07'

Data - use depends on the sequence indicator

For first part: This field contains the eight byte ICV (Initial Chaining Vector).

For middle part: This field contains between 1 to 18 eight-byte blocks of input message text.

For last part: This field contains the 4-8 byte MAC value to compare with the calculated MAC.

For only part: This field contains:

- The eight byte ICV (Initial Chaining Vector).
- Between 1 to 17 eight-byte blocks of input message text.
- The 4 to 8 byte MAC value to compare with the calculated MAC.

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'06' - Sequence Error
- X'08' - Bad Key
- X'0A' - Security Error

Data - use depends on the sequence indicator

For first or middle part: Unused - no data is returned.

For last part: This field contains the one-byte MAC verification result described above - X'01' for a successful verification, and X'00' if verification fails.

For only part: This field contains the one-byte MAC verification result described above - X'01' for a successful verification, and X'00' if verification fails.

Sequence Errors: Sequence error return codes will be generated for the following scenarios:

Middle part before First part
Last part before First part
First part after Middle part
First part after First part
Only part after First part
Only part after Middle part

In the event that a PC host application is terminated while the 4779 is processing a first part or middle part, then subsequently restarted, the PC and 4779 will lose part-sequence synchronization. A newly started PC application that sends a first part when the 4779 expects a middle or last part from the previously terminated application will receive a sequence error from the device. If this occurs, the application should send a Reset Security Function (X'F4') command to the 4779 to force a resynchronization.

Example: Below is an example of the request and response messages for the *Verify MAC* command. The data and key are the same as that described for the *Generate MAC* command on page 5-12 above.

First message (First part)

Command ID

Sequence indicator

Key ID

ICV

```
8E              123456789ABCDEF
```

Return code

Second message (Middle part)

Input message text

```
8E    1         12341234123412345678 9ABCDEF 1234FEDCBA987654321
```

Return code

Third message (Middle part)

Input message text

```
8E    1         FFEEDDCCBBAA998877665544332211
```

Return code

```
Fourth message  (Last part)
                                                  Input MAC


    8E    2        2FEFE FFE  9BB7B


          1
                                                    MAC verification result


Only message  (Only part)
                                                     Command ID

                                                    Sequence indicator

                                                   Key ID

                                                  ICV


    8E    3         123456789ABCDEF

                  1234123412341234
                  56789ABCDEF 1234
                  FEDCBA987654321                  Input message text
                  FFEEDDCCBBAA9988
                  77665544332211

                  2FEFE FFE  9BB7B                 Input MAC


          1
                                                    MAC verification result
```

Compute Verification Pattern

This command is used to compute a key verification pattern on a data key or key encrypting key stored in the 4779 device. The verification pattern is computed according to the CCA algorithm, which is shown in Appendix E, “CCA Verification Pattern Algorithm” on page E-1. The command returns two eight-byte values that are the result of the computation:

An eight-byte random number
An eight-byte verification pattern.

These two values are used as inputs if the integrity of the key is later tested with the *Verify Key* command.

Note that all DES data keys and key encrypting keys are adjusted for correct DES key parity before they are stored in the device's Data Key or KEK tables,

respectively. Thus, if a verification pattern is calculated on the key elsewhere, care must be taken to ensure the key has correct parity at that time. Since parity is optional in the DES specification, keys may or may not have correct parity when handled in other environments.

The format of the command is as follows.

Command - X'94'

Key Type Indicator

- X'00' for a Data Key
- X'01' for a Key Encrypting Key

Key Number - X'00' to X'07'

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'08' - Bad Key
- X'0A' - Security Error

Random Number (eight bytes)

Verification Pattern (eight bytes)

Verify Key

This command is used to verify that a key stored in the device has the expected value, without compromising the key in any way, or allowing its use to encipher data. The command uses two eight-byte values which are the result of the *Compute Verification Pattern* command.

An eight-byte random number
An eight-byte verification pattern.

The command uses the CCA verification pattern algorithm, so a key and its verification pattern can be generated on another device, such as the 4755 Cryptograhpic Adapter. This algorithm is shown in Appendix E, “CCA Verification Pattern Algorithm” on page E-1.

Output of the command is a single one-byte boolean value, indicating whether the key matched the verification pattern.

Result = X'01' if the key verifies against the provided Verification Pattern.
Result = X'00' if the key does not verify.

Note that all DES data keys and key encrypting keys are adjusted for correct DES key parity before they are stored in the device's Data Key table or KEK table, respectively. Thus, if a verification pattern is calculated on the key elsewhere, care must be taken to ensure the key has correct parity at that time. Since parity is optional in the DES specification, keys may or may not have correct parity when handled in other environments.

The format of the command is as follows.

Command - X'95'

Key Type Indicator

- X'00' for a Data Key
- X'01' for a Key Encrypting Key

Key Number - X'00' to X'07'

Random Number (eight bytes)

Verification Pattern (eight bytes)

The format of the response is as follows.

Expected Return Codes

- X'00' - No Error
- X'02' - Data Length Error
- X'04' - Invalid Value
- X'08' - Bad Key
- X'0A' - Security Error

Verification Result

- X'01' if the verification pattern matches
- X'00' if the verification pattern does not match

Chapter 6. 4779 Support Programs

The 4779 Support Programs permit you to perform any of the following actions:

- Load a cleartext RSA private key into the 4779 device
- Load a cleartext double-length DES key part into the 4779 device
- Load 4779 device serial number
- Read 4779 device information
- Read 4779 device status
- Eject a magnetic or smart card from the 4779 device
- Generate a CRC from an Input HEX File
- Query the CRC of the Device Memory

The 4779 Support Program components are listed in the table below.

Figure 6-1. 4779 Support Program Component List

File Name	Description
4779SUP.EXE	4779 Support Program for DOS
4779SUP2.EXE	4779 Support Program for OS/2
4779SUP.PNL	4779 Support Program Panel Library for DOS and OS/2.

Invoking the 4779 Support Program for DOS

Before you invoke the 4779 Support Program for DOS, copy files *4779sup.exe* and *4779sup.pnl* from the 4779 DOS and OS/2 Device Driver Diskette to a directory on your PC workstation. In order to execute the 4779 Support Program for DOS, the 4779 DOS driver must be installed and loaded in your workstation. To invoke the 4779 Support Program, enter *4779sup* on the DOS command line with the appropriate path specification.

Invoking the 4779 Support Program for OS/2

Before you invoke the 4779 Support Program for OS/2, copy files *4779sup2.exe* and *4779sup.pnl* from the 4779 DOS and OS/2 Device Driver Diskette to a directory on your PC workstation. In order to execute the 4779 Support Program for OS/2, the 4779 OS/2 physical device driver (4779OS2.SYS) and OS/2 DLL (x4779OS2.DLL) must be installed and loaded in your workstation. To invoke the 4779 Support Program, enter *4779sup2* on the OS/2 command line with the appropriate path specification.

Using the 4779 Support Program

The 4779 Support Programs for DOS and OS/2 are menu driven programs. The initial menu (or, panel) of the 4779 Support Program, illustrated in Figure 6-2, prompts you for the action you wish to perform. Once you select an action and press the enter key, a popup panel will appear that either requests needed input parameters for the action that you have selected or displays the requested output results. Contextual help via the F1 function key is available for all input parameters.

```
 Help
--------------------------------------------------------------------------------
                   4779 Support Program


 Select one of the following.  Then press Enter.

  1.  Load Cleartext RSA Private Key
  2.  Load Cleartext Double Length DES Key Part
  3.  Load Device Serial Number
  4.  Read Device Information
  5.  Read Device Status
  6.  Eject Card
  7.  Generate a CRC from an Input HEX File
  8.  Query the CRC of the Device Memory









 (C) Copyright IBM 1996.  All rights reserved.
F1=Help   F3=Exit   F1 =Actions
```

Figure 6-2. 4779 Support Program Initial Panel

Appendix A. 4779 DOS and OS/2 Device Driver Diskette Components

The 4779 DOS and OS/2 Device Drivers Diskette includes the following components:

Figure A-1. 4779 Device-Support Code Components

Component	Description
4779DOS.SYS	4779 DOS Device Driver
4779OS2.SYS	4779 OS/2 Physical Device Driver
x4779OS2.DLL	4779 OS/2 Dynamic Link Library (DLL)
4779OS2.H	4779 OS/2 DLL Function Prototypes for IBM C/2
4779OS2V.H	4779 OS/2 DLL Function Prototypes for IBM VisualAge C++
FIO001.MSG	4717/18/77/78/79 OS/2 Support-Code Installation Msg File
FIO001H.MSG	4717/18/77/78/79 OS/2 Support-Code Installation Help File
4779INST.EXE	4779 OS/2 Installation Program
4779INST.MSG	4779 OS/2 Installation Program Message File
4779SUP.EXE	4779 Support Program for DOS
4779SUP2.EXE	4779 Support Program for OS/2
4779SUP.PNL	4779 Support Program Panel Library for DOS and OS/2
4779APD.EXE	4779 Device Resident Application Download Program for DOS
4779APD2.EXE	4779 Device Resident Application Download Program for OS/2
4779SZIP.EXE	Self-extracting image of the 4779 Sample Application Program Sample & Executable Files
MAGCALLS.DLL	4777 OS/2 Dynamic Link Library (DLL)
MAGDLL.H	4777 OS/2 DLL Function Prototypes for IBM C/2
4777DD.SYS	4777 MVDM Device Support Code
4777VDD.SYS	4777 Virtual Device Driver
4777SAP.EXE	4777 MVDM Support Application

Appendix B. 4779 Device Return Codes

Every response message from the device to the host PC includes a one byte return code. The device uses a relatively small number of generic return codes, rather than a large number of codes with very narrow meanings. The hexadecimal return codes and their descriptions are described below:

X'00' **No Error**

Explanation: The command completed successfully.

User Response: This does *not* mean that all operations had a “True” or “Success” result; it only means that processing completed along one of the normal paths. For example, the *Verify Key* command may return *True* or *False*, depending on whether the key verified against the supplied pattern. This is independent of the return code, which would be X'00' in either case.

X'01' **Unknown command**

Explanation: The device resident application program does not support the command code specified.

User Response: Check the command code specified. If the wrong command code was issued, correct the command code and retry the command. Otherwise, it is possible that the wrong device application is loaded in the device. If this is the case, load the device resident application that supports the command and retry.

X'02' **Data length error**

Explanation: The length of the command is incorrect.

User Response: Specify the correct command length, referring to Chapter 4, “4779 Device Resident Application Function Definitions” on page 4-1 as necessary, and retry the command.

X'03' **Unimplemented command**

Explanation: The specified command is not implemented in this version of the device resident application.

User Response: Modify your application so that it does not invoke the specified command on devices with this level of device resident application or attach a device that has the correct level of device resident application.

X'04' **Invalid value**

Explanation: One or more parameters in the received command message has an invalid value. Common examples include:

A DES key number is greater than the number of the last key in the table.
An option value is not among the set of available options for the command.

User Response: Specify the correct parameter values, referring to Chapter 4, “4779 Device Resident Application Function Definitions” on page 4-1 as necessary, and retry the command.

X'05' Hardware error

Explanation: There is an apparent malfunction in the 4779 device or smart card.

User Response: Inspect device and/or smart card for hardware problems. Correct as needed. Then, retry command.

X'06' Sequence error

Explanation: Operations that must be performed in a specific sequence were attempted out of sequence.

User Response: Refer to Chapter 4, “4779 Device Resident Application Function Definitions” on page 4-1 to correct command sequence. Then, retry command.

X'07' Card error

Explanation: This return code indicates that there was an error attempting to read or write the magnetic stripe card or smart card. This error may occur if the card is inserted improperly or is damaged.

User Response: Eject card. Then retry command, making sure card is properly inserted. If failure persists, assume card is damaged and follow your institution's procedures for processing damaged cards.

X'08' Bad key

Explanation: A DES or RSA key is corrupt or invalid. This usually indicates that the DES or RSA key you are attempting to use has not been loaded. It may also mean that the most significant bit of the modulus of the RSA private key was not set to '1'.

User Response: Refer to Chapter 4, “4779 Device Resident Application Function Definitions” on page 4-1 and “4779 Security Function Calls” on page 5-2 for the correct procedure for loading and using DES and RSA keys. Properly load the keys you need and retry the command.

X'09' Bad password

Explanation: This return code indicates that the smart card password specified is incorrect.

User Response: Retry the command with the correct password.

X'0A' Security error

Explanation: This is a generic error code, used to report fatal errors that should not occur under normal circumstances.

User Response: Reset the security processor. Then, retry the current command sequence.

X'0B' **Timeout**

Explanation: A timeout occurred before an operation was complete. This happens if the user fails to enter keystrokes before the keyboard timeout period expires.

User Response: Refer to the description of the Load Device Parameters command in Chapter 4, “4779 Device Resident Application Function Definitions” on page 4-1 to set the device's timeout parameters as you desire. Then, retry the command.

X'0C' **Abort**

Explanation: Keypad operation aborted by user.

User Response: Verify that the device keypad abort key has been set up as you desire, referring to the description of the Load Device Parameters command in Chapter 4, “4779 Device Resident Application Function Definitions” on page 4-1. Then, retry the command.

X'0D' **Invalid PIN or password**

Explanation: This return code indicates that the entered PIN or password was either entered incorrectly or that the entered PIN or password does not match the expected value.

User Response: Follow your institution's procedures for processing invalid PIN's.

X'0E' **Feature not available**

Explanation: This return code indicates that the requested function is not supported in this model of the device. For example, the application issued the Magnetics Write command, but the device is a 4779-1 model which does not support encoding (writing to) a magnetics card.

User Response: Attach a device which supports the requested function and then retry the command. Or, do not issue this command for the current device.

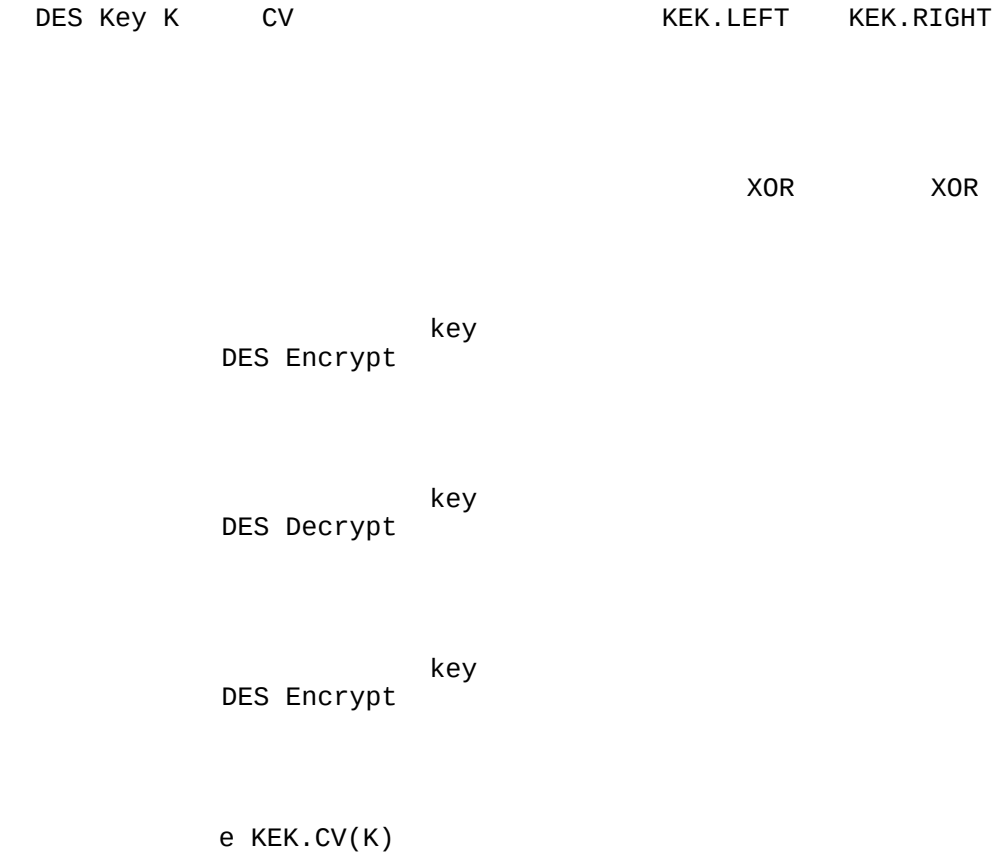
X'86' **Host Application Sequence Error**

Explanation: A DOS Read followed a DOS Write that failed. This is an Application sequence error.

User Response: Verify that the device is attached and powered on. Then retry the command.

Appendix C. Triple Encryption Algorithm With Control Vector

The diagram below shows the algorithm used for triple encryption with a control vector:

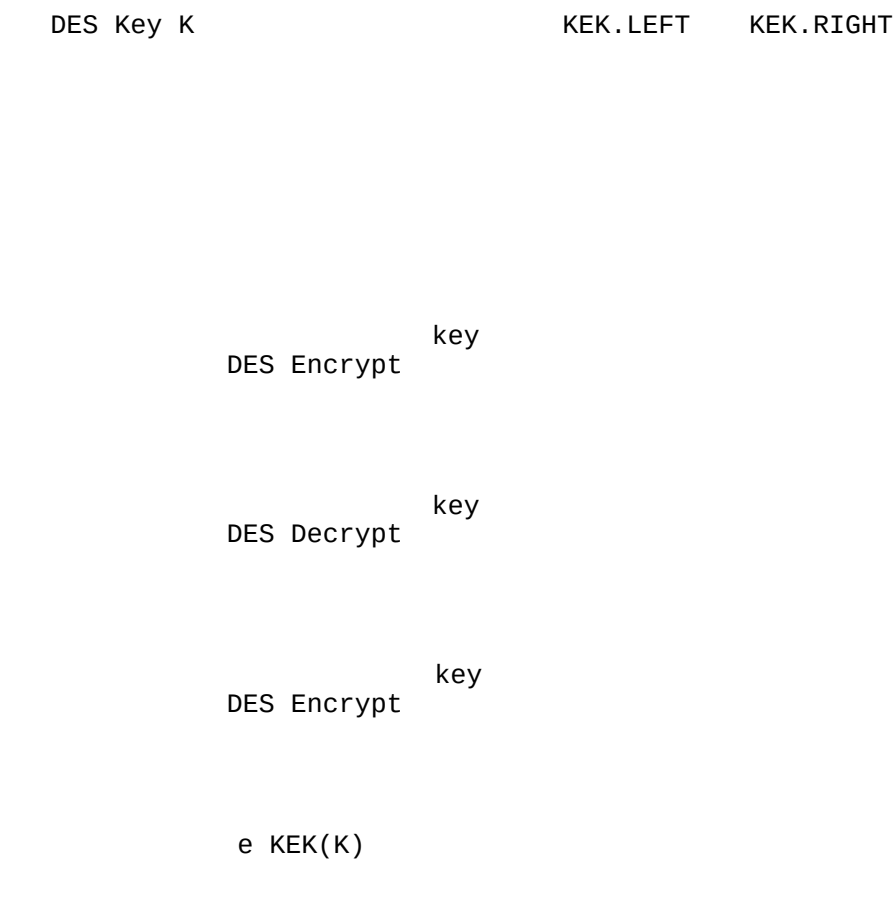


Notes: 1. Input is a DES data key K, with an associated Control Vector CV.
2. Key K is enciphered under a Key Encrypting Key KEK, which has left half KEK.LEFT and right half KEK.RIGHT.
3. The notation e KEK.CV(K) denotes the triple-encryption of key K under key encrypting key KEK, using control vector CV.

Figure C-1. Triple-encryption of a DES key with a Control Vector

Appendix D. Triple Encryption Algorithm Without Control Vector

The diagram below shows the algorithm used for triple encryption without a control vector:

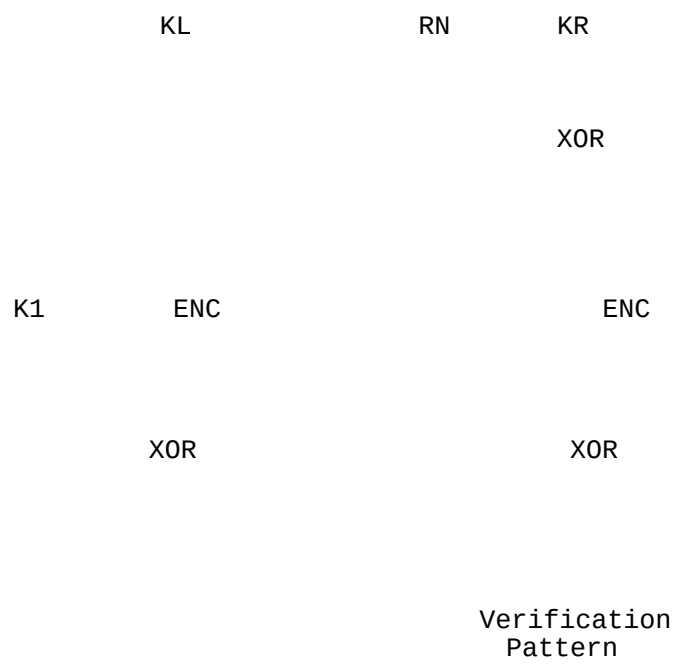


Notes: 1. Input is a DES data key K.
2. Key K is enciphered under a Key Encrypting Key KEK, which has left half KEK.LEFT and right half KEK.RIGHT.
3. The notation e KEK(K) denotes the triple-encryption of key K under key encrypting key KEK.

Figure D-1. Triple-encryption of a DES key

Appendix E. CCA Verification Pattern Algorithm

The diagram below shows the algorithm used to compute key verification patterns. This algorithm is a part of CCA, and is also described in the Transaction Security System Programming Reference: Volume I, Access Controls and DES Cryptography, cited in “References” on page xii.



Notes:
1. K1 is a constant DES key, with value X'4545454545454545'.
2. ENC = DES ECB encryption.
3. KL and KR are the left and right halves of a 16 byte key K.
   KL consists of the first 8 byte key part of a key load command.
   KR consists of the last 8 byte key part of a key load command.
4. KR is treated as zero when computing a verification pattern for a Data Key.
5. RN is a random number.
6. Caution: Keys that have been loaded into the device with no parity will be adjusted to have odd parity by the device. These parity bits are included in the computation of the verification pattern. The least significant bit of each byte of a key is the parity bit.

Figure E-1. Algorithm for computing a key part verification pattern

Appendix F. Machine Information Data Structure

Introduction

This appendix describes the Machine Information Data Structure (MIDS), which contains information describing the hardware and BIOS features of the 4779 device. This object can be accessed by the internal application program in order to determine the features of the machine in which it is running. This is essential, in order to permit the design of application programs that can run on more than one version of the 4779.

It is likely that future versions of the 4779 will include features we do not foresee today. Thus, it is imperative that the MIDS design be fully extensible, so that new elements can be added while still maintaining full compatibility with every earlier version of the structure. In order to accomplish this, the object uses concepts from ASN.1 and is based on TLV (Tag-Length-Value) objects. Each object begins with a two-byte tag identifying its purpose. The tag is followed by a two-byte length field, indicating how many bytes of data are associated with the structure. This number of data bytes then follows the tag and length fields. Successive TLV structures are concatenated to one another.

TLV Objects can be either *simple* object, or *constructed* objects. The difference is that simple objects contain only their associated data, while constructed objects contain other TLV objects. For MIDS, the entire MIDS structure is a constructed object, while each category of device characteristics is a separate simple object within the constructed object. Examples below will make this clearer.

The tag-based structure allows us to parse the information without requiring any of the objects to be fixed in length. This has two very important benefits with regard to future expansion.

Any object can be extended to meet future requirements by adding new information at the end of the object. Older applications that do not know about this extension will never see it, but they will still be able to parse the entire set of objects because they will use the integrated length fields.

Entirely new objects can be added at any time. Applications will parse the data to find objects of interest, and will ignore all others. Single objects may be retrieved based on the **tag** value.

Categories and Tags

The machine information is separated into functional categories, where each category is given a separate tag. The categories are as follows.

Figure F-1. MIDS tags

Construct Tag	Object Tag	Category
X'80'	X'01'	This is the tag for the constructed MIDS object, which contains all the other objects in its data field.
X'00'	X'02'	General device information which does not fall into the other categories.
X'00'	X'03'	Magnetic stripe read and write information.
X'00'	X'04'	Keypad information
X'00'	X'05'	Smart card reader information.
X'00'	X'06'	Display information.
X'00'	X'07'	Communications interface information.
X'00'	X'08'	Information about the version of BIOS resident in the device.
X'00'	X'09'	Information about the model type of 4779.

Notice that the construct tag X'80' has the high order bit set to 1, while all the simple objects carry tags with this bit set to 0. The high order bit is used to distinguish simple and constructed tags, allowing the parsing software to determine whether to expect the data field to contain raw data or a series of TLV objects.

Data Associated with the Tags

The Constructed Object

All the simple objects described below are assembled within a single MIDS constructed object with tag X'8001'. The simple tags can appear in any order within the constructed object. The MIDS object will be a single object, which contains the simple objects in the following form:

Figure F-2. Constructed Object Form

Offset	TAG	Length	Object1	Object(n-1)	Object(n)
00-01	X'8001'	Quantity of bytes to follow	Simple Object	Simple Object	Simple Object

The length of the constructed object is the total length of all the encapsulated simple objects.

Example Constructed Object

The following figure contains an example of a complete MIDS data structure matching the characteristics of the 4779-2 as of July, 1996. The figure is annotated to identify each of the components.

Figure F-3. Constructed Object Example

Tag	Length	Value	Description
X'8001'	X'0043'		MIDS constructed object
X'0002'	X'0001'	X'02'	General device information
X'0003'	X'0002'	X'8782'	Magnetic stripe information
X'0004'	X'0004'	X'80' X'0A' X'00' X'04'	Keypad information
X'0005'	X'0002'	X'83' X'01'	Smart card reader information
X'0006'	X'0004'	X'80' X'04' X'14' X'0D'	Display information
X'0007'	X'0001'	X'81'	Communications information
X'0008'	X'0014'	X'5645523A2031322E33342C2030372F33312F3935'	BIOS information
X'0009'	X'0001'	X'02'	General device information

Note: The BIOS information uses a contrived example of a possible build date and version number.

Simple Objects

The following sections describe the information carried in the data field for each of the simple objects.

X'0002' - General Device Information

This object describes general features of the 4779, which do not pertain to any particular component or subsystem. The information consists of a single one-byte field with two bit significant values:

General Device Information

- Bit 7 - Reserved, Value is 0 until later defined.
- Bit 6 - Reserved, Value is 0 until later defined.
- Bit 5 - Reserved, Value is 0 until later defined.
- Bit 4 - Reserved, Value is 0 until later defined.
- Bit 3 - Reserved, Value is 0 until later defined.
- Bit 2 - Reserved, Value is 0 until later defined.
- Bit 1 - Value is 0 for a device with the base security feature, and 1 for the enhanced security feature device.

– Bit 0 - Value is 0 for a motorized card reader, and 1 for a hand operated reader.

Figure F-4. General Device Information

Offset	Element
00-01	Tag X'0002'
02-03	Length X'0001'
04	Flags

Example

If the device has the enhanced security feature and a motorized reader, the general device information object would look like this:

```
X'  2'  X'  1'  X' 2'
```

X'0003' - Magnetic Stripe Information

The magnetic stripe information describes features involved in reading and writing magnetic stripe cards. This information consists of two one-byte fields:

Three bits indicate which tracks can be read in this device. The most significant bit indicates presence of a reader.

Tracks that can be read.

– Bit 7 - Reader Present, Value is 1 when Magnetic Stripe Reader is present.
– Bit 6 - Reserved, Value is 0 until later defined.
– Bit 5 - Reserved, Value is 0 until later defined.
– Bit 4 - Reserved, Value is 0 until later defined.
– Bit 3 - Reserved, Value is 0 until later defined.
– Bit 2 - Track 3
– Bit 1 - Track 2
– Bit 0 - Track 1

For example, `X'83'` indicates that the device can read tracks 1 and 2, but not track 3.

Three bits indicate which tracks can be encoded with this device. The most significant bit indicates presence of a encoder.

Tracks that can be encoded.

– Bit 7 - Encoder Present, Value is 1 when Magnetic Stripe Encoder is present.
– Bit 6 - Reserved, Value is 0 until later defined.
– Bit 5 - Reserved, Value is 0 until later defined.
– Bit 4 - Reserved, Value is 0 until later defined.
– Bit 3 - Reserved, Value is 0 until later defined.
– Bit 2 - Track 3
– Bit 1 - Track 2
– Bit 0 - Track 1

For example, `X'82'` indicates that the device can encode track 2, but not tracks 1 or 3.

Figure F-5. Magnetic Stripe Information	
Offset	**Element**
00-01	Tag X'0003'
02-03	Length X'0002'
04	Read Tracks
05	Encode Tracks

Example

If the device can read tracks 1, 2 and 3, and can encode track 2, the magnetics information object would look like this:

```
X'   3'   X'   2'   X'8782'
```

X'0004' - Keypad Information

The keypad information indicates whether the 4779 has an integrated keypad, and if so it provides some information about the available keys. The information is presented in four bytes, as follows:

Byte 0: Presence of keypad

- Bit 7 - Keypad Present, Value is 1 when Keypad is present
- Bit 6 - Reserved, Value is 0 until later defined.
- Bit 5 - Reserved, Value is 0 until later defined.
- Bit 4 - Reserved, Value is 0 until later defined.
- Bit 3 - Reserved, Value is 0 until later defined.
- Bit 2 - Reserved, Value is 0 until later defined.
- Bit 1 - Reserved, Value is 0 until later defined.
- Bit 0 - Reserved, Value is 0 until later defined.

Value is 0 if there is no keypad, or X'80' if a keypad is available.

Byte 1: Number of numeric keys

- Value is 0 if there are no numeric keys, and will be 10 (X'0A') if keys 0-9 are available.

Byte 2: Number of alphabetic keys

- Value is 0 if there are no alphabetic keys on the keypad. Otherwise, it denotes the number of alphabetic keys available.

Byte 3: Number of function keys

- Value is 0 if there are no function keys on the keypad. Otherwise, it denotes the number of function keys available.

Figure F-6 (Page 1 of 2). Keypad Information	
Offset	**Element**
00-01	Tag X'0004'
02-03	Length X'0004'
04	Presence
05	Numeric
06	Alpha

Figure F-6 (Page 2 of 2). Keypad Information	
Offset	**Element**
07	Function

Example

If the device has a keypad with numeric keys 0-9 as well as four function keys, the keypad information object would look like this:

```
X'   4'   X'   4'   X'8 '   X' A'   X'   '   X' 4'
```

.

X'0005' - Smart Card Reader Information

This object indicates whether the device includes a smart card reader, and if so it provides information on the types of smart cards supported. The information is presented in two bytes, as follows:

Byte 0: Smart card protocols

Three bits in this byte are used to indicate support for three possible smart card communications protocols. The most significant bit indicates presence of a smart card reader.

- Bit 7 - Reader Present, Value is 1 when Smart Card Reader is present.
- Bit 6 - Reserved, Value is 0 until later defined.
- Bit 5 - Reserved, Value is 0 until later defined.
- Bit 4 - Reserved, Value is 0 until later defined.
- Bit 3 - Reserved, Value is 0 until later defined.
- Bit 2 - 1 if the reader supports IBM T=14 protocol.
- Bit 1 - 1 if the reader supports ISO T=1 protocol.
- Bit 0 - 1 if the reader supports ISO T=0 protocol.

Byte 1: Read head positions

Two bytes indicate whether the smart card read head is positioned for the permanent ISO standard read position, the transitional position, or both.

- Bit 1 - Reader supports transitional contact position.
- Bit 0 - Reader supports ISO standard contact position.

Figure F-7. Smart Card Reader Information	
Offset	**Element**
00-01	Tag X'0005'
02-03	Length X'0002'
04	Protocols
05	Positions

Example

If the device has a smart card reader that supports ISO T=0 and T=1 in only the ISO standard contact position, the object would look like this:

```
X'   5'   X'   2'   X'83'   X' 1'
```

X'0006' - Display Information

The display information indicates whether the device includes a display, and if so it provides information about the features of that display. The information is presented in four bytes, as follows:

Byte 0: Presence of Display

- Bit 7 - Display Present, Value is 1 when Display is present
- Bit 6 - Reserved, Value is 0 until later defined.
- Bit 5 - Reserved, Value is 0 until later defined.
- Bit 4 - Reserved, Value is 0 until later defined.
- Bit 3 - Reserved, Value is 0 until later defined.
- Bit 2 - Reserved, Value is 0 until later defined.
- Bit 1 - Reserved, Value is 0 until later defined.
- Bit 0 - Reserved, Value is 0 until later defined.

- Value is 0 if there is no display, or X'80' if one is available.

Byte 1: Number of rows

Byte 2: Number of columns

Byte 3: Attributes

Four bits in this byte are used to indicate support for possible display attributes. Note that other attributes can be added to this byte as needed, and additional bytes can be added if there is a need to describe more than eight attributes.

- Bit 7 - Reserved, Value is 0 until later defined.
- Bit 6 - Reserved, Value is 0 until later defined.
- Bit 5 - Reserved, Value is 0 until later defined.
- Bit 4 - Reserved, Value is 0 until later defined.
- Bit 3 - 1 if the display supports reverse video characters.
- Bit 2 - 1 if the display supports blinking characters.
- Bit 1 - 1 if the display supports underlined characters.
- Bit 0 - 1 if the display supports programmable characters.

Figure F-8. Display Information

Offset	Element
00-01	Tag X'0006'
02-03	Length X'0004'
04	Presence
05	Rows
06	Columns
07	Attributes

Example

If the device has a 20 character by 4 row display that supports programmable characters, the object would look like this:

```
X'   6'   X'   4'   X'8 '   X' 4'   X'14'   X' 1'
```

X'0007' - Communications Interface Information

This object contains information about the interface the 4779 can use to communicate with other devices, such as a PC. Note that many applications call for a standalone device, with no communications.

The communications information consists of one byte field:

Byte 0: Presence of communications interface

- Bit 7 - Value is 1 if the unit has the ability to communicate with a Host
- Bit 6 - Reserved, Value is 0 until later defined.
- Bit 5 - Reserved, Value is 0 until later defined.
- Bit 4 - Reserved, Value is 0 until later defined.
- Bit 3 - Reserved, Value is 0 until later defined.
- Bit 2 - Reserved, Value is 0 until later defined.
- Bit 1 - Reserved, Value is 0 until later defined.
- Bit 0 - RS232 communications, Value is 1 when RS232, otherwise 0

This byte contains a code, which is mapped to each of the possible communications interfaces possible in the 4779. Today, there is only one interface supported:

Figure F-9. Communications interface types

Code	Interface type
X'81'	RS232 interface present and shareable with serial 4777.

Figure F-10. Communication Interface Information

Offset	Element
00-01	Tag X'0007'
02-03	Length X'0001'
04	Type

Example

If the device provides a 4778/4718 compatible serial interface, the object would look like this:

```
X'   7'   X'   1'   X'81'
```

X'0008' - BIOS Information

This information identifies the version and build date of the resident BIOS. The data enclosed in this object consists of a 20 character string containing the version number and build date in the form "VER: vv.vv, mm/dd/yy".

Figure F-11. Communication Interface Information

Offset	Element
00-01	Tag X'0008'
02-03	Length X'0014'
4-23	VER: vv.vv, mm/dd/yy

Example

If the BIOS code is version 12.34, created on the date 07/31/95, then the version information would be the ASCII string "VER: 12.34, 07/31/95", which is hex 5645523A2 31322E33342C2 3 372F33312F3935. Thus, the BIOS information object would look like this:

```
X'   8'   X'  14'   X'5645523A2 31322E33342C2 3 372F33312F3935'
```

X'0009' - 4779 Model

This information identifies the model of 4779.

Figure F-12. 4779 Model Number

Offset	Element
00-01	Tag X'0009'
02-03	Length X'0001'
04	Model Number

Example

If the 4779 model is Model 2 then two(2) is the model number object.

```
X'   9'   X'   1'   X' 2'
```

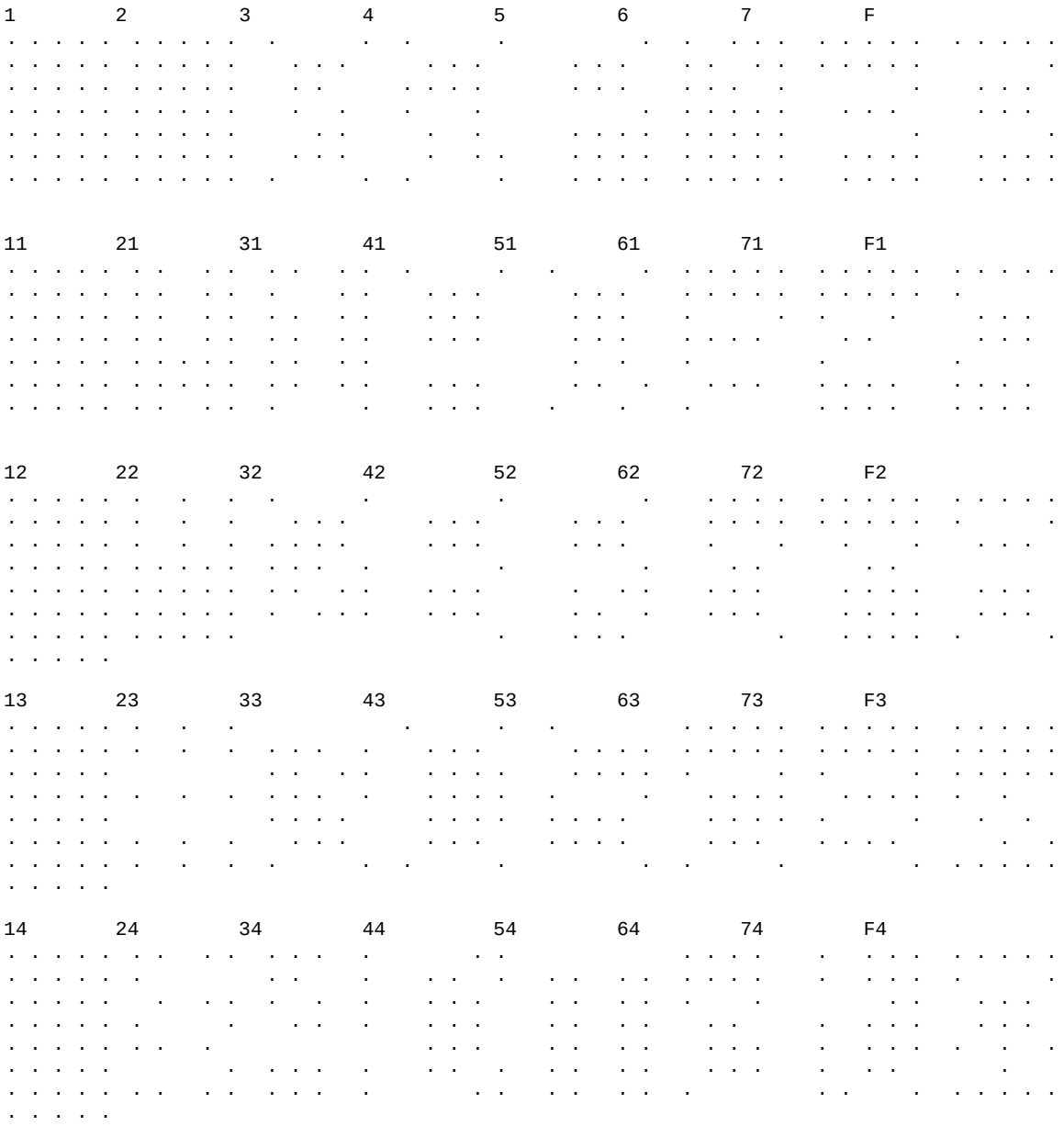
Application Access

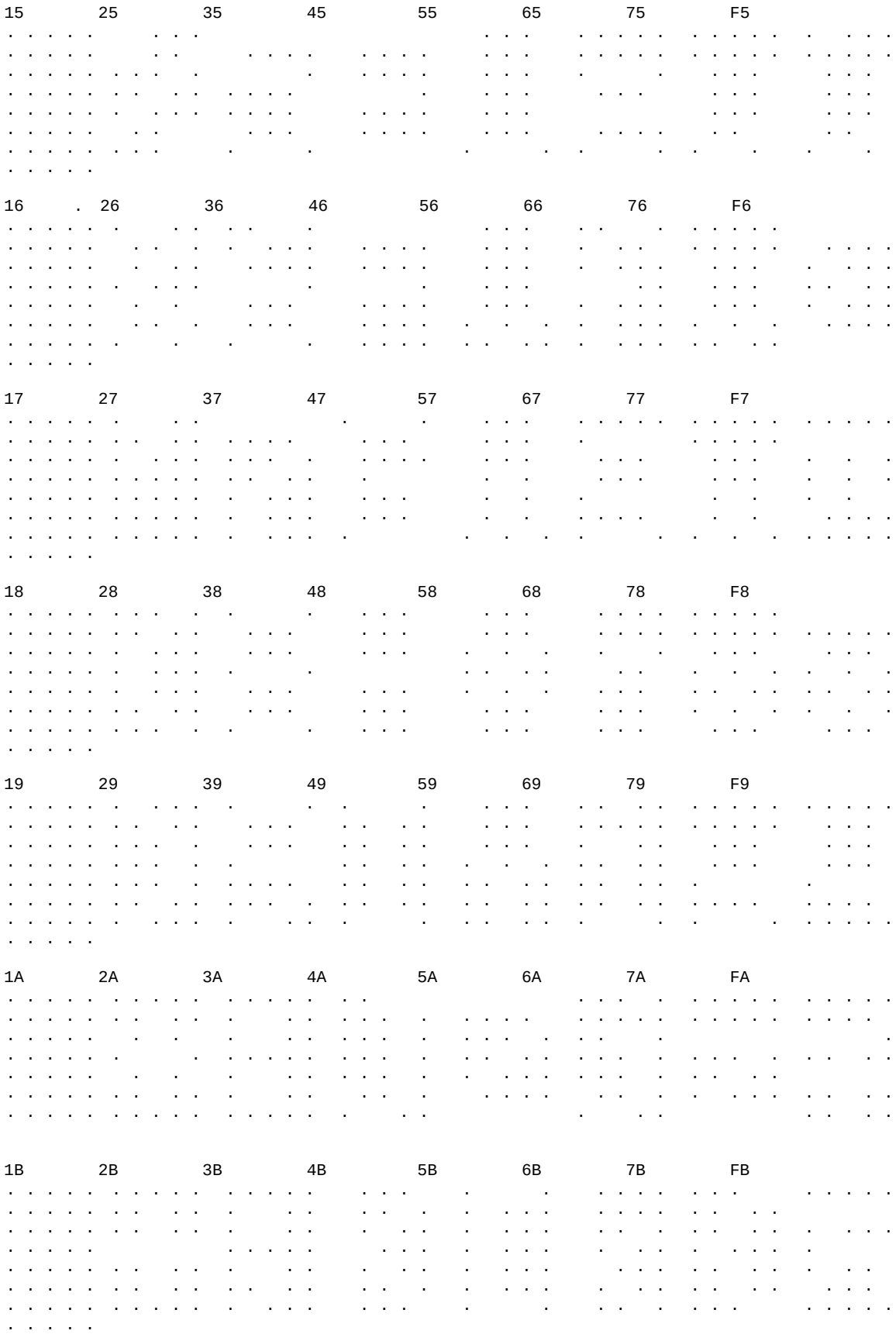
Applications may request the entire constructed object or some Simple object. Requesting the constructed object using Tag X'8001' retrieves all the simple objects information in TLV form, see Figure F-3 on page F-3 for constructed object form. Simple object Tags X'0002' to X'0009' retrieves only the information pertinent to the tag.

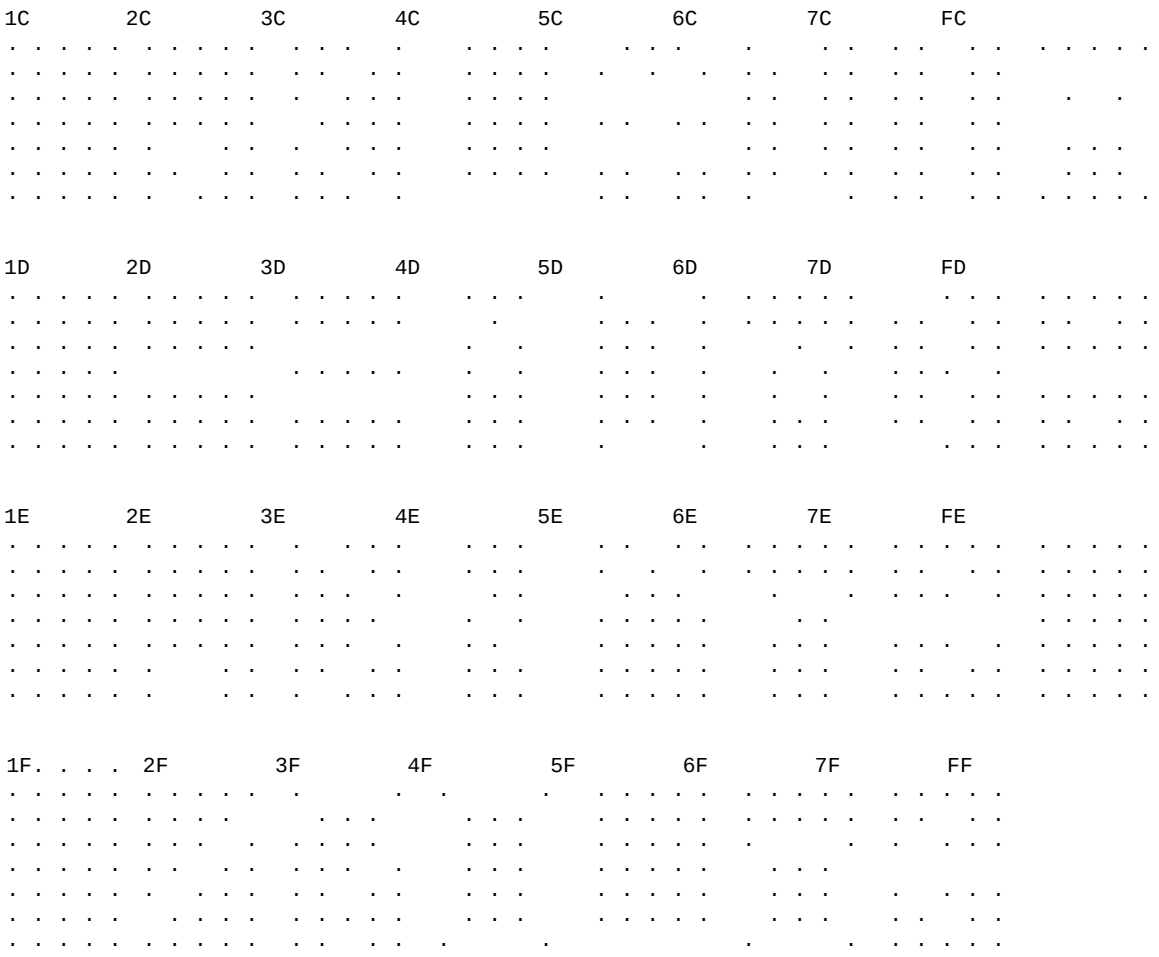
Appendix G. Character Font Table

Character Font Table

The following character font table will be available in the LCD. Character positions 00 through 07 are programmable RAM characters. Character positions 00 and 01 are preprogrammed with the characters 'up arrow' and 'down arrow' respectively. Positions 02 to 07 are blank. Refer to the Execute Utility Functions command found in chapter 4 for a description of how to program character positions 00 to 07. Character 20 is a blank. Other character positions which are not in the character set are shown as blanks but programming these character codes may produce unsupported character patterns.




15
25
35
45
55
65
75
F5
16
26
36
46
56
66
76
F6
17
27
37
47
57
67
77
F7
18
28
38
48
58
68
78
F8
19
29
39
49
59
69
79
F9
1A
2A
3A
4A
5A
6A
7A
FA
1B
2B
3B
4B
5B
6B
7B
FB


1C 2C 3C 4C 5C 6C 7C FC
1D 2D 3D 4D 5D 6D 7D FD
1E 2E 3E 4E 5E 6E 7E FE
1F 2F 3F 4F 5F 6F 7F FF

Index

Numerics

A

C

D

E

F

G

I

K

Communicating Your Comments to IBM

4779 Hybrid Smart Card Device
Programming Guide

Publication No. SA34-2360-01

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

4779 Hybrid Smart Card Device
Programming Guide

Publication No. SA34-2360-01

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
SA34-2360-01

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

Cut or Fold
Along Line

IBM

Part Number: 82H4793

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.