Intermec
Technologies Corporation

*PEN*KEY® 6100 Computer*

PROGRAMMER'S REFERENCE GUIDE

P/N 977-054-001
Revision B
December 2000

We welcome your comments concerning this publication. Although every effort has been made to keep it free of errors, some may occur. When reporting a specific problem, please describe it briefly and include the book title and part number, as well as the paragraph or figure number and the page number.

Send your comments to:
Intermec Technologies Corporation
Publications Department
550 Second Street SE
Cedar Rapids, IA 52401

Acknowledgments

ActiveX, Microsoft, MS, and *MS-DOS, Windows,* and *Windows NT* are registered trademarks and *MSDN, Visual Basic, Visual C++,* and *Windows for Pen* are trademarks of Microsoft Corporation.

Borland, *dBase*, and *Turbo Pascal* are registered trademarks and *Borland C* and *C++ for Windows* are trademarks of Borland International, Inc.

CIC is a registered trademark, and other CIC product names, service names, slogans or logos referenced are trademarks or registered trademarks of Communication Intelligence Corporation.

Handwriter is a registered trademark of Communication Intelligence Corporation.

Intel is a registered trademark of Intel Corporation.

PenRight! and *PenRight! Pro* are trademarks of PenRight Corporation.

SanDisk is a trademark of SanDisk Corporation.

SystemSoft is a registered trademark and *CardSoft* and *CardID* are trademarks of SystemSoft Corporation.

XMS is a registered trademark of Lotus, Intel, Microsoft, and AST Research.

CONTENTS

Preface

SECTION 1

Getting Started

SECTION 2

Supporting DOS Applications

SECTION 3

Supporting Windows Applications

SECTION 4

Power Management

SECTION 5

Communications and Device Support

SECTION 6

Conversions and Interfaces

SECTION 7

Reference, System Information

SECTION 8

Reference, Open Systems Publications

APPENDIX A

Sample Configuration Files

APPENDIX B

Common PEN*KEY 6000 Series Information

FIGURES

TABLES

INDEX

Preface

Introduction

This section is intended to help you to navigate through the book, as well as other useful topics to assist you in your development of your 6100 Computer.

Topic Summary

Topic	Page
Scope	1
Related Publications	2
Look at What has Changed (from previous editions)	2
The Structure of the Book this is an overview of how this publication is structured. If you are unfamiliar with this Programmer's Reference Guide, this section could be of assistance in getting around.	3
Locating Applications in this Book Lists several ways to locate information for any application described in this book. There is a table showing all of the applications and where you can find the descriptions of these applications.	5
Customer Support This paragraph provides the telephone numbers you may need if you discover a problem or need assistance in developing applications for your 6100 Computer. There is also a detailed list of the steps needed for access to the Intermec BBS.	6

The *Contents* (just previous to this section) and the individual *Topic Summaries* (in each section in this publication) could be useful in assisting you in your search for information in this book. For example, the *Topic Summary* in the *Supporting DOS Applications* section would be helpful in locating the details of a particular DOS application, because it provides the name of each DOS application and the page number where it is described.

Scope

The material presented in this publication pertains to the PEN*KEY® 6100 Hand-Held Computer. A few paragraphs are common to the PEN*KEY 6000 Series Computers, in general. These differences are obvious by their use in the text.

Flash Version 1.16

This publication has been updated for the drivers in flash version 1.16, but can be useful for units containing later flash versions. Refer to your Tool Kit for the latest information relating to the applications included in your version of flash.

Level of Expertise Needed

The majority of this material is intended for experienced application programmers.

For the New User

If you need further instruction before delving into the more detailed sections of this publication, then consider some of the available resources. The *Reference, Open Systems Publications* section lists several publications that could be useful for additional research.

Also, Intermec Technologies Corporation provides training and support for purchasers of our products. Refer to the *Customer Support* paragraph, on page 6, in this section.

Refer to *Appendix B, Common PEN*KEY 6000 Series Information*, for information that may be of interest to new users. For example, the *Memory Overview* and *The Windows Environment* paragraphs contain valuable information for new users.

Related Publications

PEN*KEY Model 6100 Hand-Held Computer User's Guide; P/N: 961-028-085.

Look at What has Changed (from previous editions)

Changes have taken place, in an effort to make this publication more usable and to provide easier access to information. Obviously, some of you are already familiar with finding information in previous editions; this may cause you some grief. However, please take time to become familiar with the new organization. These changes are based on input from readers, and for the most part were implemented to make the task of locating information easier for the reader. Unfortunately, we made these changes in several steps. We apologize for that; however, as readers tell us what works and what does not, we try to improve the reading environment.

The following examples may help you to understand where (and why) some of these changes have been implemented.

Recent Changes

System Information

The system reference information was originally located in several places in the book; and some information was hidden in sections whose title did not reveal what topics it contained. Therefore, the *Reference* section was split into two sections, to provide more visibility to the material contained therein. The new section titles became: *Reference, System Information* and *Reference, Open Systems Publications*.

With edition 2.0, all of the system information was moved into the *Reference, System Information* section. The title for the section (from which the system information was extracted) became: *Communications and Device Support*.

Interrupt Index

With edition 1.4, the alphabetical index for the interrupts is removed from the end of the *Conversions and Interrupts* section, and relocated to the end of the *Index* section, primarily because we changed to another desktop publishing program, which causes extra maintenance for an index to exist somewhere other than the Index section. The title of that index is now: *Interrupt Index*, and is located at the very end of the book so you can easily find it.

With edition 1.4, the "Interrupt Reference Table" was relocated to the "Index" section, under "Interrupt Index". With edition 2.0, it was then reorganized so you can find interrupts (alphabetically) by topic. The original order was by verb, which I discovered was totally useless.

Reprogramming Flash Memory

This material is moved to the *Getting Started* section, since it is similar to the types of information contained in that section.

Earlier Changes

Organization in General

When the *6100 Programmer's Reference Guide* was first developed from the original *PEN*KEY Programmer's Reference Guide*, a review by some representatives of various Intermec departments agreed the organization needed to be changed. Comments from readers (of the original manual) seemed to confirm that information was hard to find.

Sample Configuration Files

This information is removed from the *Getting Started* section and relocated to *Appendix A*, because it constitutes information to which statements in several different paragraphs need to refer. It also simplified the task of updating this information with each new edition.

Common 6000 Series Information

This information is collected from several places in the book and relocated to *Appendix B*. This is necessary to preserve it from extinction. New users still need some of this information. And placing it in the back of the book prevents experienced PEN*KEY users from tripping over it.

The Structure of the Book

The following shows how this publication is structured. The topics listed below consist of the actual names of the sections in this publication:

- ***Getting Started –*** contains information to help you to become familiar with the 6100 Computer. It describes the 6100 open system environment, the Tool Kit, tips for getting started, how to reprogram flash memory, how to setup for development from the PC, and some development environments and resources.
- ***Supporting DOS Applications –*** describes some DOS applications supported for the 6100 Computer, as well as printing and power management for DOS.
- ***Supporting Windows Applications –*** describes the minimal Windows installation, the NORAND Windows shell, several Windows applications that are supported for the 6100 Computer, printing and power management for Windows.

- ***Power Management –*** describes the Advanced Power Management (APM) supported on the 6100 Computer as it relates to the APM BIOS, power states, device power control, and firmware error codes.
- ***Communications and Device Support –*** describes the following:
  - **Communications Support:** INTERLNK and INTERSVR, NORAND Utilities, serial, and LAN communications.
  - **Device Support:** an overview of the 6100 memory, including upper memory management utility, keyboard definitions, and the 6100 display.
- ***Conversions and Interfaces –*** the first part of this section is devoted to porting 4000 applications to the 6100 Computer. The second part includes a detailed *Interrupt Cross-Reference* and a list of interrupts, organized by interrupt number. There is a white tab at the beginning of the second part:
  - **Conversions:** this is a set of paragraphs applicable to converting 4000 Series applications for use on the 6100 Computer.
  - **Cross-Reference by Interrupt Numbers:** provides a handy means of locating interrupts, by the interrupt number.
  - **Interrupt Definitions:** a comprehensive list of interrupts supported for the 6100 Computer, including the definitions for using them.
- ***Reference, System Information –*** includes system hardware information such as: messages, errors, IRQ and I/O maps, system variables, ROM DOS 5 operating system, and boot process.
- ***Reference, Open Systems Publications –*** includes a list of publications which are referenced from other places within this document, or that may be useful for developing applications for the 6100 Computer.
- ***Appendix A, Sample Configuration Files –*** provides examples of configuration files for your 6100 Computer.
- ***Appendix B, Common PEN*KEY 6000 Series Information –*** contains information that is common to all (or most) of the PEN*KEY 6000 Series Computers. This information could be useful to new users:
  - **Memory Overview:** is an overview of the memory system and some noteworthy design aspects of the PEN*KEY 6000 Series system.
  - **The Windows Environment:** an overview of the way in which Windows, BIOS, the applications, the drivers, and hardware all work together. It also includes an approach for determining the memory requirements for the Windows-based PEN*KEY 6000 Series systems.
- ***Index –*** a fairly comprehensive list of indexes to topics and items of interest for the entire publication:
  - **General Index:** includes all of the indexes within this publication, except for the *Files Index* (below) and the *Interrupt Index* (below).
  - **Files Index:** a fairly comprehensive index of the names of applications and other files mentioned within this publication.
  - **Interrupt Index:** an alphabetical index of all interrupts. These indexes are alphabetized by nouns, as well as most verbs.

    However, no indexing has been done for verbs such as: Read, Get, and Set, which include so many interrupts they get in the way when searching for other interrupts.

Locating Applications in this Book

There are several ways to locate the information for a desired application:

- The *Contents* (previous to this section) contains entries to all of the applications documented in this publication.
- The *Files Index* lists most of the occurrences of each file name included in this publication, including the application program names. All file names are listed under the topic, "Files:", in the Index.
- If you know whether a particular application is a DOS or Windows application, then you may find it either in the *Supporting DOS Applications* section or in the *Supporting Windows Applications*.
- The table, below, is another method of locating an application. It is provided to assist you in locating the applications supported for the 6100 Computer. The names of the applications are listed in the "Application" column. Next, the "Description" column is the actual paragraph title where the application is described. And finally, the "Section" column lists the section where the application description is located.

Application	Description	Section
4000API.EXE	4000 Series Programming Interfaces	Conversions and Interfaces
61MOUSE.COM	DOS Pen Driver	Supporting DOS Applications
61PODSCN.EXE	DOS Scanner	Supporting DOS Applications
61SCAN.DRV	Integrated Scanner	Supporting Windows Applications
BOOTDRV.COM	Other NORAND Software	Getting Started
CALIB.EXE	Other NORAND Software	Getting Started
6100DISP.DRV	Other NORAND Software	Getting Started
CRC32.EXE	Tool Kit for the 6100 Computer	Getting Started
DELAY.EXE	Other NORAND Software	Getting Started
ELANCFG.EXE	ELAN Configuration Driver	Supporting DOS Applications
ELANAPM.EXE	Power Management BIOS	Power Management
ELANAPM.EXE	Power Management BIOS Interfaces	Conversions and Interfaces
ELANUMP.SYS	Upper Memory Provider	Communications and Device Support
FONTSEL.EXE	4000 Screen Emulation for the 6200	Conversions and Interfaces
IRDAPDRV.SYS	DOS IrDA Printing	Supporting DOS Applications
ISRAMDRV.COM	Other NORAND Software	Getting Started
MININET.EXE	6000 Series LAN Communications	Communications and Device Support
MMBFLAG.COM	Other NORAND Software	Getting Started
NORDOSPM.EXE	DOS Power Management Driver	Supporting DOS Applications
NORIRDA.DRV	IrDA Printing for Windows	Supporting Windows Applications
NORMOD.SYS	Docks and Modems	Communications and Device Support
NORNPCP.DRV	NPCP Printing for Windows	Supporting Windows Applications
NORSHELL.EXE	NORAND Shell for Windows	Supporting Windows Applications
NORWINPM.DRV	Windows Power Management Driver	Supporting Windows Applications

Application	Description	Section
PC4800.SYS	DOS NPCP Printing	Supporting DOS Applications
PENALIGN.EXE	Other NORAND Software	Getting Started
PENWIN.DLL	Pen for Windows	Supporting Windows Applications
PRDRV.SYS	DOS IrDA Printing	Supporting DOS Applications
PSROM0C.EXE	NORAND Utilities	Communications and Device Support
RAMDFMT.EXE	The RAM Drive	Getting Started
UCLKPEN.DRV	Windows Pen Driver	Supporting Windows Applications
VPOWERD.386	Windows Power Management Driver	Supporting Windows Applications
VROTATE.EXE	4000 Screen Emulation for the 6200	Conversions and Interfaces

Refer to the paragraphs and sections listed above for details of these applications. Also, refer to the *"Development Resources"* paragraph, in the *"Getting Started"* section, for details of applications listed as *"Other NORAND Software"* in the "*Description*" column of this table.

Customer Support

Factory Service

If your unit is faulty, you can ship it to the nearest authorized Service Center for factory-quality service.

Customer Support Center

The Intermec Customer Support Center (technical support) telephone number is 800-755-5505 (U.S.A. or Canada) or 425-356-1799. The facsimile number is 425-356-1688. Email is *support@intermec.com*.

If you email or fax a problem or question include the following information in your message: your name, your company name and address, phone number and email to respond to, and problem description or question (the more specific, the better). If the equipment was purchased through a Value-Added Reseller please include that information.

Web Site

The Customer Support File Libraries, including Hot Tips and Product Awareness Bulletins, are available via the Intermec Product Support page at this URL: *http://norbbs.norand.com/index.htm*. New users can sign up for a new account on this page.

PDF versions of Intermec manuals can be found at this URL: *http://corp.intermec.com/manuals/english.htm*.

Bulletin Board Service

The Customer Support Bulletin Board (BBS), maintained by Intermec Technologies Corporation, provides software and documentation:

- **Phone number:** 319-369-3515 (14.4 Kbps modem)
  319-369-3516 (28.8 Kbps modem)
- **Protocol:** Full duplex, ANSI or ANSI-BBS; 300 to 28,800 bps; v.32bis; 8 bits, no parity, 1 stop bit. *For high-speed modems, disable XON/XOFF and enable RTS/CTS.*

This is the same location available via the web site. If your web access uses high-speed phone lines, the web interface provides a faster response.

PREFACE

Section 1

Getting Started

Introduction

This section contains information to help you become familiar with your 6100 Hand-Held Computer (HHC), so you can start building applications.

If you are not familiar with the structure of the PEN*KEY® 6000 Series Programmer's Reference Guides, be sure to turn back to the introduction section and read *Structure of the Book* and *Locating Information*.

Topic Summary

The following list may be of assistance in locating other topics of interest:

List of Figures and Tables

An Open System Environment

The 6100 Operating System consists of standard MS-DOS, version 5.0, and Windows, version 3.1.

One of the major benefits of this open-system approach is that you can acquire development equipment and software from many different vendors, including Intermec Technologies Corporation. This provides you with wide latitude in selecting the equipment and software tools that are best suited to your particular development needs. The challenge lies in finding pieces that work well together, especially when you are working in the areas of communications and interfaces.

If you have DOS and/or Windows programming experience, you will quickly feel comfortable with the PEN*KEY 6100 platform.

Introduction to the 6100 Computer

From the viewpoint of an application, the 6100 Computer is like other PCs (386 33 MHz), with some exceptions, as described below.

How the Hardware is Different

The 6100 Computer is a ruggedized, ergonomic, battery-powered, touch-based computer, with input from finger touch, stylus, keyboard, or scanner. It has integrated communications and various combinations of external and/or internal peripherals. It is designed for a mobile environment.

Processor

The 6100 Computer contains an AMD ELAN 386 33 MHz processor.

Display

The 6100 Computer features a backlit, LCD, touch-sensitive display. The normal mode is landscape, but most applications use a rotated screen (portrait mode). The physical display is the size of a 1/4 VGA (240 x 320 pixels). However, it uses a CGA controller.

User Input

The keyboard consists of a built-in numeric keypad with control keys. Much of the user input is done using touch, pen, or bar-code scanning.

Power Management

Advanced Power Management (APM) is critical. It contains one main battery and one backup battery. This combination of software and batteries provides a smart battery pack.

Beginning with flash version 1.11, the following features have been added to the APM BIOS:

- Suspend/resume coordination (Suspend/resume key mapped to I/O key if IO2SUS.COM is loaded):
  - To work with NORAND Card and Socket services
  - For radio integration
- Added OEM calls to match the portable APM specification
- Updated fuel gauging to support all pack types
- Updated idling functionality for lower run-time power

A new power management system (with version 1.15) has improved the capability of resuming or waking the terminal; and it is highly recommended for all 6100 Computers, and especially if you have a PC Card modem.

Batteries

▼CAUTION: **This terminal has ONE primary power source, the MAIN Battery. The only function of the Backup Battery is to maintain power to RAM while changing Main Batteries. At all times during operation and while being stored (overnight, long periods of time, and even while in suspend), the unit must have its Main Battery installed.**

The Main Battery is used, even in suspend mode, to maintain RAM, RTC, and CMOS settings. When the terminal is off charge, the Main Battery also supplies any needed charge to the Backup Battery.

The Backup Battery is for emergencies only. The Backup Battery should not be relied upon, for extended periods, to support the system with no Main Battery.

A unit with functioning Main and Backup Batteries can be left off charge overnight or for a long weekend. This presumes that it has a fully charged and functioning Main and Backup Battery.

▶NOTE: *A discharged Backup Battery takes 14 hours to recharge. A discharged Main Battery takes 2 hours to recharge.*

System Memory

System memory is provided via DRAM and Flash memory modules.

- Flash memory provides storage for executable and system files.
- For flash versions prior to 1.11, flash memory can be expanded up to eight megabytes. It is expected that on a future release the flash will be available in larger amounts.
- For flash version 1.11 and later, a 1 megabyte flash was used. Currently, only a 1 megabyte configuration is supported.

▶NOTE: *If you upgrade a system that had a flash version prior to 1.11, you will also need to upgrade the flash. Refer to the* ***Reprogramming Flash Memory*** *instructions on page 1-7.*

- DRAM can be expanded up to 16 megabytes.
- An MS-DOS 5 RAM disk may be created out of DRAM for fast, temporary storage of data and program files.

Input, Output, and Storage Devices More Varied

PC Card

Two PC Card type II slots, or one type III slots are available. These slots are for non-volatile SRAM, flash or hard disk data storage, radio or land modems, or other devices. System resources, which are available to the applications, may also store on the flash drive.

▶ **NOTE:** *Throughout this publication, cards that conform to the PC Card interface standard (the new standard), or the PCMCIA interface standard (the old standard), are commonly referred to as: "PC Card", or "PC Card xxxxx" (where xxxxx consists of " modem", "drive", or other device type). This is done because PC Card is rapidly becoming the accepted industry term for a storage medium that conforms to one of these standards.*

Surface and Pin Connections

These connections support battery charge, printers, scanners, and other peripherals.

Infrared Printing

IrDA printing is also available.

Serial Data Communications

Communications takes place through standard UART ports.

Keep the System Environment in Mind

The 6100 system environment should be kept in mind, while writing applications for it. The 6100 Computer has no external keyboard port, uses a flash memory file system, and has a built-in VGA viewing screen.

Applications programmers need to remember the constraints listed previously, and the flash file system is usually kept write-protected, to prevent overwriting.

The Hardware Ports

For information relating to the communication ports, addresses, IRQs, and devices supported for these ports, refer to the *Reference, System Information* section of this publication.

Tool Kit for the 6100 Computer

The file complement in the Tool Kit for the 6100 Computer differs from one release to the next. For an accurate list of Tool Kit files, refer to the RELNOTES.TXT file, included in the Tool Kit distribution package.

The Tool Kit contains DOS and Windows resources for configuration, power management, communications, and peripherals. Tool Kits are available, as follows:

Part Number (P/N)	Description
215–598–001	6100 DOS (only) Tool Kit, WITH manual
215–597–001	6100 DOS (only) Tool Kit, WITHOUT manual
215–600–001	6100 DOS and Windows Tool Kit, WITH manual
215–599–001	6100 DOS and Windows Tool Kit, WITHOUT manual

The files and documentation in the Tool Kit are available to purchasers of the 6100 Computer, who hold the following licenses:

- NORAND Operating System (P/N: 999-001-014)
- MS-DOS and Microsoft Windows (P/N: 999-001-041)
- MS-DOS (P/N: 999-001-042)

The DOS Tool Kit requires a DOS license only.

The 6100 Computer, as shipped from the factory, has preloaded flash memory with the files required for booting and telecommunicating.

At a minimum, DOS 5.0 is preloaded In flash. To create the environment required by your application software, select any other system files needed from the Tool Kit and copy them to the 6100 Computer.

The Tool Kit does not include application software. You may obtain such software from Intermec, from third-party suppliers, or you can design your own.

The files necessary to run Windows and the application software are often loaded onto the RAM drive. The instructions included in the Tool Kit assume that you will install all files on the RAM drive. Other methods of arranging files are possible. And in some cases, this requires substitution of the appropriate drive letter, in place of D:, for the RAM drive.

File Integrity Verification Utility: CRC32.EXE

The Tool Kit contains a utility, CRC32.EXE, for verifying the integrity of the files provided on the Tool Kit diskettes. CRC32.EXE calculates a 32-bit CRC value for a file. This value may then be compared with the factory CRC value for the file. Factory CRCs are found in the file CRC.ALL , also in the Tool Kit.

Usage is as follows:

```
CRC32 [@][filename | pathname] [/s]
```

where:

filename is the name of the file on which the CRC is calculated. One or more files or directories can be processed at one time.

pathname is the location of the file to be processed. Wildcard processing is not allowed (in the pathname specified after the "@" symbol, nor any of the pathnames within the argument file). A valid argument file has the same format as an argument for the IPLFMT.EXE program.

/s indicates all subdirectories should be searched for matching file names.

@ is (optionally) included in front of the filename to indicate it is an argument file, which contains names of files to be checked.

Refer to the RELNOTES.TXT file in the Tool Kit for the CRC values for each of the modules used on the 6100 Computer.

Tips for Getting Started

As applications are developed for the 6100 Computer (or port existing applications), keep in mind the following basic considerations, ideas, and suggestions:

1. ROM DOS 5 is the operating system, with Windows 3.1.
2. Understanding how the system configuration files are used in the 6100 environment provides easier application development. These files include DOS configuration files: CONFIG.SYS and AUTOEXEC.BAT; and Windows configuration files: SYSTEM.INI and WIN.INI.
3. Become familiar with the tools and techniques for power management. Monitoring the state of battery power can be an especially critical function.
4. When developing for pen input, remember that interpreting handwriting is still a developing field; accuracy has not yet reached 100 percent. Furthermore, interpreting and storing the results puts an additional load on the processor. Excessive use of handwriting recognition software can slow down an otherwise speedy application. Instead, try to use buttons, item lists, and pull-down menus for common tasks.
5. When designing a pen-centric interface, use the area provided by the VGA screen. Make buttons, pull-down menus, text entry fields, etc. large enough for easy, accurate use in a mobile or high-pressure environment.
6. Make your applications *drive-independent*. Do not hard-code drive designations. Utilize the many available PC Card storage solutions, remembering that the devices can be moved about as required by different configurations. The PC Card slots can accept nonvolatile SRAM, flash, hard disk devices, RF devices, radio or land modems, and other devices.
7. Keep in mind the general system design of the pen-oriented 6100 Computer: flash memory file system, built-in CGA screen and keyboard, etc.
8. There are some files listed in this publication that could be useful for your configuration or application. If you need any of these files, first look in the Tool Kit for them. If not found there, try one of the Product Forums on the Intermec BBS.
9. Finally, for development purposes you may consider using certain external devices. Keep in mind whether these items are available (or practical) for the application to use in the field.

System Configuration

Minimum Development Configuration

Note that the following paragraphs apply to the standard configuration for the 6100 Computer; but keep in mind that it can be configured to meet your specific needs. Additional RAM can be obtained to bring total RAM to 2, 4, 8, or 16 MB.

The following configuration items are needed for a minimum development, except for items 4 and 5, which are only useful if you do not have a docking station.

1. PEN*KEY 6100 Computer, with two megabytes of system RAM for DOS (or 4 MB of system RAM for Windows) and 1 MB of flash RAM.
2. The DOS Tool Kit or the Windows Tool Kit for the 6100 Computer.
3. Main batteries: at least one.
4. One 2- or 4-MB SRAM card (see Note 1). You need at least one 2-MB SRAM card, which is useful when you need to update the flash software. For larger storage, SanDisk flash memory cards are available (see Note 2).

▶ **NOTE 1:** *SunDisk has been changed to SanDisk.*

▶ **NOTE 2:** *The 6100 Computer supports SRAM PC Cards natively — meaning, no extra device drivers to load. However, SanDisk cards are not supported through the 6100 BIOS and thus, need Card and Socket services to be loaded before they can be used.*

5. PC Card reader/writer (for example, DataBook card reader-writer or notebook computer that can read/write PC cards).
6. Cable for connecting 6100 Dock to a PC (NULL modem cable).
7. If printing is needed, cable for connecting dock to serial or 4815 Printer.
8. Single dock or wall charger.

Sample Configuration Files

The configuration files, CONFIG.SYS and AUTOEXEC.BAT, are required for an appropriate environment for the 6100 Computer.

Some examples of configuration files, used to configure the 6100 Computer for DOS or Windows, are detailed in *Appendix A, Sample Configuration Files*.

Reprogramming Flash Memory

▶ **NOTE:** *Always keep the 6100 Computer on charge while performing any setup, reprogramming, or reflashing.*

Updating to Flash Version 1.16 or Later

If you have a flash version older than 1.16, it is highly recommended that you obtain an upgrade to the latest version of flash.

A self-extracting executable archive file, found in the Tool Kit contains the entire flash load. Run this executable from a temporary directory on a desktop PC. After running the executable, a file is produced that explains how to reprogram the flash (which is essentially the same information presented here).

The self-extracting archive also contains 61FL1000.BIN, the 6100 Initial Flash Load (IFL) card image, and PROG.BAT, an automatic reflashing batch file. These two files create an IFL card, which in turn installs the flash image on 6100 Computer.

▶ **NOTE:** *If you are updating from an older flash to version, and you have an 8 MB system, you must format the RAM drive to a size of zero. Otherwise the system may lockup. The only way to recover from this, once it occurs, is to remove the DRAM board.*

DOS Configurations

You **must** change the following:

- All system files should be accessed from drive D:
- for IRDAPDRV, remove the –b??? switch and add the –x switch
- for ELANCFG.EXE, remove the /C /R switches.
- ELANCSSS.EXE is needed, no matter what kind of card you have

You now have the option to change the following:

- You can replace CardSoft drivers with NORCS drivers
- ATABIOS.SYS is needed for SanDisk support
- NORATA.SYS is needed for SanDisk support
- NORMOD.SYS is needed for modem card support
- ELANCFG.EXE, you can set the /L switch to a value greater than one.

Windows Configurations

You **should** change the following:

- Power driver to NORWINPM.DRV
- EMM to exclude range C000 to C3FF (if using NORCS drivers)
- Pen driver from UCLKPEN.DLL to UCLKPEN.DRV

You need to remove the following:

- SystemSoft drivers (if you are using NORCS drivers)
- MMSYSTEM.DLL
- all .WAV files (optional)

Prerequisites for IFL Card Creation

The following items are required for creating an IFL card:

- Disk file 61BCxxxx.EXE contains the new flash archive. The first two characters of the filename indicate the model number for the hardware configuration of the 6100 Computer. The next two characters indicate the upgrade configuration. For example, BC means Boot Card (IFL) version. The last four characters indicate the version of the flash. For example, 0116 would indicate flash version 1.16.
- A 2-megabyte (or larger) SRAM card with battery installed.
- A PC Card reader/writer adapter (drive) for a standard PC, together with the associated software, or a PC that can access PC Cards.
- A standard PC for use with the PC Card reader/writer or a PC that can access PC Cards. If the card drive is an external unit, you can use a laptop or notebook computer.
- The PROG.BAT utility and the 61FL1000.BIN.

Creating a Master Mode Boot (IFL) Card

Use the PROG.BAT utility, included in the release package, to copy the IFL card image to the SRAM card by entering the following at the DOS command prompt:

```
PROG 61FL1000.BIN d:
```

where “d” is the drive letter of the PC card drive. This command copies the contents of 61FL1000.BIN to the SRAM card in the format needed to perform a master mode boot. After successfully creating the IFL card, write-protect it.

NOTE: *Do not use the DOS COPY or XCOPY command to create the IFL card. Master mode booting requires certain files at specific locations on the SRAM card. DD.EXE (which is part of the PROG.BAT process) is designed to place every byte in a specific location on the media; these DOS commands are not.*

NOTE: *The PROG.BAT file performs a write operation followed by a verify operation. It is possible for the write operation to succeed and the verify to fail. This is particularly true when using a DataBook TMD–500–03. If you are using such a card drive, this problem can be corrected by downloading updated drivers from the DataBook BBS.*

Updating PEN*KEY 6100 Flash

Preferred Approach

To boot the IFL card, place it in drive B (this is the drive closest to the display), then press the Reset button.

Follow the prompts on the display to update the PEN*KEY 6100 Flash. To verify that you have updated the Flash, look for "PEN*KEY 6100 FLASH 61FL1000" and the current version number on the top line of the display when starting from the ROM drive D. See the release notes for information on changes in this version of Flash.

Master Mode Boot Approach

It is possible to boot an IFL card even if the flash in your PEN*KEY has never been programmed or has been corrupted. Use this procedure if the preferred approach above does not work.

To perform a master mode boot, insert the IFL card in drive B. Then, press and hold the Suspend/Resume key(I/O), while holding down the reset button. The screen displays "Master Mode Boot" for a few seconds, while performing the memory test. (If you do not see "Master Mode Boot", try to reset again.)

Follow the prompts on the display to update the PEN*KEY 6100 Flash. To verify that you have updated the Flash, look for "PEN*KEY 6100 FLASH 61FL1000" and the current version number on the top line of the display when starting from the ROM drive D. See the release notes for information on changes in this version of Flash.

Creating a Custom Flash

If necessary to create a custom flash for an application, a flash customization utility is available. Use the following procedure to create the custom flash:

1. Obtain the flash customization utility from the BBS 6100 forum. If you do not have access to this, contact your field technical support person to forward it on to you. This is a .zip file that has the file structure of the flash plus the BIOS files that need to be built into the image.
2. Extract this file with PKUNZIP using the –d option to a subdirectory on your hard drive, such as C:\6100\CUSTOM\. A subdirectory, FLASH, has been created. This is where you can modify files, add files and subdirectories, and delete files.

▶ **NOTE:** *Be aware! DO NOT attempt to add files beyond the 1 MB limit for disk space on the 6100 Computer. This does not check disk space for you.*

For recommendation on usage of necessary power drivers to be loaded in the boot files, refer to the Power Management section, of this publication. It would be a good idea to modify the first line of the CONFIG.SYS with some comment about the version of your custom flash for easy reference.

3. Once the files in the FLASH subdirectory are the way you want them, run the RELEASE.BAT file in the root directory. This creates the file, 61FL1000.BIN. This is the image of the master mode boot program.
4. Using the included PROG.BAT batch file, program a 2M or larger SRAM card with the file, 61FL1000.BIN. This is located in the BOOTCARD subdirectory in the flash on your 6100 Computer. For example,

```
C:\NORAND\6100\FLASH\BOOTCARD>PROG ..\61FL1000.BIN E:
```

5. Now, place the 6100 Computer on EXTERNAL CHARGE. (Or verify that the battery is fully charged) ***This is a very important***, because you cannot afford to lose power during this procedure.
6. Place the 2M SRAM card in the front slot (B:, the slot closest to the display) and press the reset button.
7. The 6100 Computer will boot from the card, and ask you to verify that you want to reflash the terminal. Once you press '3', the unit will reflash. When it is done, remove the card, and press the reset button again.
8. Verify that your new flash has been loaded.

While RELEASE.BAT program is running, the file, 61FL1000.BIN, is created by IMAGE.EXE. Use the information below to configure your custom flash.

Imager for DOS, Version 2.2
Creates or modifies a disk image file.
`IMAGE imgfile [switches] [filelist]`
`imgfile` and the pathname of the image to create or modify.

When creating a new file, specify the format and size using the switches below:

/BOOTSEC=pathname

Specifies the pathname of a binary file used to define the logical boot sector. If this switch is not specified, a DOS 5.0 compatible boot sector with English error messages is supplied.

/DATE=mm/dd/yy[yy]

When this switch is used, all files and subdirectories have their datestamp set to the supplied value. This switch does not affect the corresponding timestamp. If this switch is not supplied, the datestamp for all files and subdirectories is copied from the source and the datestamp for the volume label is set to today's date.

/FATS=[1|2]

Normally DOS allocates two file allocation tables (FAT) as does this program. To conserve space in the image file, use one FAT instead of two.

/FILES=n

Specify how many files can be stored in the root directory.

/FORMAT=[360|512|720|1024|1440|2048|2880|4096[K[B]]]

Defines which disk format to use within the image. The default format is 720K.

/LABEL=["]volume name["]

Defines the disk volume label. If not supplied, the image has no label.

/ORG=n[[K[B]]

This switch changes which cluster is to next store data. All format parameters specified after an ORG switch are ignored.

/SPC=n

Specify how many sectors per cluster to use.

/TIME=hh:mm:ss

When this switch is used, all files and subdirectories have their timestamp set to the supplied value. The corresponding datestamp is not affected. If this switch is not supplied, the timestamp for all files and subdirectories is copied from the source and the timestamp for the volume label is set to the current time.

/VOLSER=xxxx[–]xxxx

Defines the volume serial number to be placed in the logical boot sector of the disk image. If this switch is not supplied a pseudo-random volume serial number is generated.

@pathname

Since the DOS command line is limited to about 128 characters, this switch provided so that switches may be specified in a file.

pathname (with wild card characters)

Any parameter other than those specified above is treated as a pathname of files that should be added to the image. When a subdirectory name is supplied, the entire subdirectory tree is added to the image. All format parameters specified after a pathname are ignored. Note that the only way to create a subdirectory is to add a subdirectory that has children subdirectories.

(a final comment)

DOS environment variables can be used either on the command line or within an argument filename simply by enclosing the environment variable in percent signs as in .BAT files. For example %TEMP% evaluates to the current value of the TEMP environment variable.

Key Files Used in the Flash Upgrade Process

The following files are necessary for reflashing a 6100 Computer:

- FLASH.EXE Writes data to flash
- FLASH.BIN Flash image that goes into flash

Development Environments

The following development environments are supported on the 6100 Computer:

Microsoft DOS 5.0

- PenPal (DOS), by PenPal Associates, Inc.
- PenRight! Pro (DOS), by PenRight Corporation

Microsoft Windows 3.1

- Microsoft Pen Extensions, for Windows
- Microsoft Visual Basic, for Windows
- Borland Delphi, for Windows
- Microsoft Visual C++ or Borland C++, for Windows

DOS

The 6100 Computer can run any mouse-aware DOS program using only DOS. The Tool Kit provides a DOS mouse driver (61MOUSE.COM), and is used with any mouse-aware application, meaning the program runs in landscape mode.

Screen rotation can be handled several different ways. Two examples are:

- Using VROTATE.EXE and FONTSEL.EXE
- For Borland graphics, using the BGI driver (for more information on this, refer to the *DOS Pen Calibration* paragraph, in the *Supporting DOS Applications* section, and the *BGI Support* paragraph, in *Appendix A, Sample Configuration Files*)

PenDOS Handwriter Recognition System

CIC's PenDOS *Handwriter Recognition System* is a DOS application. It is not included in the Tool Kit. A license for CIC's PenDOS (specific to the PEN*KEY 6000 Series computers) can be obtained through Intermec. To order PenDOS, contact a Intermec sales representative.

PenPal (DOS)

Use PenPal to quickly draw screens and develop small, simple applications and prototypes. PenPal provides an easy-to-use interface for both experienced and relatively inexperienced programmers. It uses a simple proprietary language for defining the functionality of the application. PenPal was developed specifically for mobile systems. Because of this, it is of a manageable size for a 6100 Computer. PenPal handles dBase, ASCII, and text files.

NOTE: *PenPal Associates no longer provides enhancement support for PenPal in the environment in which PEN*KEY 6000 Series Computers operate, except for bug fixes. This means if you continue to use PenPal with the display configuration in which your 6100 Computer was shipped, it continues to serve you well; but, if you want to use PenPal with a different display configuration, PenPal Associates will not necessarily provide you with a version that works with your configuration.*

For an example of setting up PenPal (DOS), refer to the *Setups for Third Party Applications* paragraph, in *Appendix A, Sample Configuration Files*.

PenPal runs with CIC's Handwriter for PenDOS, or with no recognizer. With no recognizer, signature capture is available and character recognition is disabled.

PenPal runtime licenses (specific to the PEN*KEY 6000 Series computers) can be obtained through Intermec. To order PenPal runtime licenses, contact a Intermec sales representative. PenPal development packages can be ordered from the manufacturer, PenPal Associates.

For an example setup for Handwriting Recognition System, refer to paragraph, *Handwriting Recognition Setup*, in *Appendix A, Sample Configuration Files*.

PenRight! Pro (DOS)

PenRight! is a development environment that also includes screen-drawing capability, as well as C code. These tools make it possible for an experienced C programmer to add extra functionality to an application. PenRight! Pro was also developed specifically for mobile systems, and creates programs of a manageable size for a 6100 Computer. PenRight!, which is slightly harder to use than PenPal, has dBase support, as well as ASCII text files. PenRight Pro! development kits can be obtained through the manufacturer, PenRight!.

For an example of setting up PenRight! (DOS), refer to the *Setups for Third Party Applications* paragraph, in *Appendix A, Sample Configuration Files*.

Windows

The 6100 Computer can run any mouse-aware Windows application. Note that the Windows configuration provided by Intermec is not a full-featured Windows. You may find that files you need have not been included. Add the files if needed.

Borland's C compiler includes a utility called TDUMP. You can run TDUMP on an executable file to display the files or libraries called by the executable.

Handwriting Recognition

If handwriting recognition is required, there are two options available. One option is provided by a product from Communication Intelligence Corp. (CIC); the other option is provided by a product from Synaptics. For descriptions, refer to the *CIC Handwriter Recognition System for Windows* paragraph and the *Synaptics Handwriter Recognition HR-1200* paragraph, on page 1-13.

If signature capture (bit maps) is required, but handwriting recognition is not required, this can be done with regular Windows. Simply trap the mouse-move and mouse-pressed events and manually draw the ink. The ink then can be saved as bit maps and compressed, if necessary. Microsoft Visual Basic Professional Edition has an example of "catching ink" in this way. The extra files for enabling the Pen Extensions are provided in the Tool Kit.

Standard and Enhanced modes for Windows are discussed the in *Memory Overview* topic, in *Appendix B, Common PEN*KEY 6000 Series Information*. Contact your Intermec sales representative to order software and manuals. Any of the following handwriter recognition packages can be ordered:

Manufacturer: Communication Intelligence Corp. (CIC)
Software: Handwriting Recognition System for Windows

Manufacturer: Synaptics
Software: Handwriting Recognition HR–1200

CIC Handwriter Recognition System for Windows

This system is a full-featured recognizer which includes a "trainer" that provides a means to train the recognizer to better recognize your handwriting.

▶ **NOTE:** *This product only works in the ENHANCED mode.*

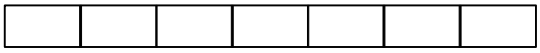
This software also requires numerous files to be installed, about 720K bytes of disk space. Once loaded, it recognizes handwriting in any text field. In addition, it recognizes standard Pen Extensions for Windows 1.0 gestures.

Synaptics Handwriter Recognition HR-1200

This system is a more limited recognizer. It requires only one file to be installed, which is about 200K bytes in size.

It can run in standard or enhanced mode Windows. It only recognizes handwriting in boxed or hashed edits, which are Pen Windows controls.

Boxed edit (bEdit) looks similar to the following, and each letter goes in a box:



Hashed edit (hEdit) looks similar to the following, and each letter goes in a slot:

A Pen Extensions for Windows program, PENPAL.EXE, (or Pen Palette) makes use of the hashed edit, and can enable applications that are not written for handwriting recognition. This program allows a user to write in the boxes, then send the characters to the application with the current focus. This file and the Pen Extensions for Windows files can be found on the fourth diskette of the Tool Kit.

Pen Extensions for Windows

Pen Extensions for Windows consists of standard Windows with extra files, some changes to SYSTEM.INI, and a new file called PENWIN.INI. Pen Extensions for Windows provides character recognition and an easier interface for capturing ink. A handwriting capture engine must be purchased separately, either through Intermec or other sources.

Keyboard Options

Even though the onscreen keyboard, SKB.EXE, from *Microsoft Windows for Pen Computing* is available in the Tool Kit, it is too wide for the PEN*KEY 6100 portrait-size screen. If you need an onscreen keyboard, you could write your own keyboard application or consider ordering another keyboard application that runs in portrait mode.

The *Configurable PEN Popup Keyboard* from Intermec is one that is suitable for portrait screens, and was created to provide a universal means of enabling key-intensive operations for any of the NORAND pen computers (such as the 6100 Computer). It runs on any Windows 3.1-compatible system that supports Pen Windows. It is totally configurable and uses a tabbed-notebook metaphor, where each notebook tab represents an independent, configurable sub-keyboard. A file (.INI) configures the keyboard. The keyboard kit contains default .INI files for several of the PEN*KEY 6000 Series Computers, including the 6100 Computer. The keyboard automatically creates the proper number of tabs based upon the contents of the .INI file. The legends that appear on the tabs are user-defined. The tabbed notebook metaphor permits a large, complex keyboard to be subdivided in any way desired, which minimizes the screen space required when the keyboard is active. All Windows key functions are supported except for multiple shifts (e.g., Ctrl+Alt+F1). For each keyboard you can specify the Windows font, basic key cell size, key layout, any combination of double-wide and double-high keys, the legend displayed on the key for normal, shifted, Alt-shifted, and Ctrl-shifted operation, and the key sequence that is sent to Windows. This means that the keyboard can be configured to send macros of up to 255 characters each, each of which can be sent by pressing a single key.

The User's Guide and software comes in one package using the following part number: P/N 215-601-001. See your Intermec sales representative for information on licenses and pricing.

Microsoft Visual Basic for Windows

Microsoft Visual Basic (VB), for Windows, is an easy-to-use tool that lends itself to quick creation of prototype screens for customer demos and reviews. It includes screen drawing plus an easy method for inserting code to control the application. A few warnings, however, are in order. If frequently used forms are not preloaded, the screen drawing may be slow. Also, the data-aware controls are very nice, but supporting a database with these controls consumes a lot of memory. VB can handle a large number of databases, including dBase, Access, Paradox, and Foxpro. It can also handle ASCII and text files.

When you encounter a Visual Basic program, you should be aware of some common requirements.

- All Visual Basic applications require VBRUNxxx.DLL, where *xxx* corresponds to the version of Visual Basic used to develop the application (e.g., VBRUN300.DLL is required for applications developed under VB V3.00).
- Files with the VBX extension are Visual Basic custom controls. These files support common features (such as command buttons, list boxes, pen edit boxes, data-aware controls, etc.) that are frequently seen in Windows apps. They are typically distributed with the application that requires them.
- Programs written in Visual Basic V4.0 or higher must be compiled in 16-bit mode; PEN*KEY 6100 Windows does not support 32-bit mode programs.

NOTE: *Be aware that the 6100 Computer is a 386-based system. Performance with Visual Basic may be an issue, especially with the latest Visual Basic offerings.*

Borland Delphi for Windows

Delphi is a Borland product that allows quick application development with Pascal support.

Microsoft Visual C++ or Borland C++ for Windows

These are tools for experienced C Windows programmers. The foundation class libraries or the object windows library can be used; just make sure to keep an eye out for space limitations. Windows programs, in general, grow to a large size quickly and require a lot of extra DLLs or VBXs.

Other Environments

Obviously, any development environment, that runs on the aforementioned operating systems, may potentially be used to develop software for the 6100 Computer. However, not all of the development environments will necessarily work as well, nor are they recommended. This is mainly, because of size and speed limitations of the 6100 Computer.

Some Screen Considerations

Some of the standard Windows dialog boxes are too large. For example, the file open dialog does not fit. Running in landscape mode requires a different display driver, VGAP.DRV. This is the standard display driver for Windows and is not customized to the smaller screen of the 6100 Computer. You need to make changes in the SYSTEM.INI file for correctly aligning the pen. Look in the `[Pen Driver]` section of SYSTEM.INI. Refer to the 6100 forum on the BBS for more details and a sample configuration.

Some Keyboard Considerations

Even though the onscreen keyboard from Pen Windows (SKB.EXE) is available in the Tool Kit, it is too wide for the portrait-size screen on the 6100 Computers. Since it is impossible to use both landscape and portrait modes within the same application in Windows, it is not advisable to run SKB.EXE in landscape mode. If you need an onscreen keyboard that runs in portrait mode, you may either write your own keyboard application or consider ordering another keyboard application.

You might want to consider the Configurable Pen Popup Keyboard available from Intermec, for an extra charge. The part number for this product is P/N 215-601-001. Documentation is included.

Some System Guidelines

There are other usable development environments. Here are some guidelines that can help to determine whether a particular environment is viable for a 6100 application.

1. How much space is required?

   Many environments assume that the target machine is a desktop or laptop with virtually unlimited hard drive space. This is certainly not the case with a hand-held computer; memory is still at a premium. For example, Power Builder is a very popular environment for development. However, just to get an application started, Power Builder uses more than three megabytes of DLLs in addition to Windows just to run the Hello World program. Power Builder applications tend to be large and slow; at this time, it is not recommended.

2. (Windows) How are the dialogs being stored?

   This is a question you must ask if you want to use a tool that provides one environment for both DOS and Windows applications. If the dialog information is stored as separate files, you will find it very hard to get adequate response time for drawing the screens. Also, some of these tools insulate you from the Windows APIs; consequently, it becomes very difficult to use any of the APIs that are provided for you.

3. The 6100 Computer uses a 33 MHz 386 processor. Test the performance of your application on a comparable computer.

▶ **NOTE:** *A typical development machine (60 100 MHz 486) hides some speed issues that is evident on a slower 386 machine.*

Setup for PC Development

▶ **NOTE:** *Always keep the 6100 Computer on charge while performing any setup.*

The following steps outline a general approach for equipment setup:

1. Connect power to the single dock.
2. Connect a NULL modem cable between the dock and a serial port on the desktop or portable PC.
3. If the PC Card slots contain cards, remove the cards.
4. Install the main battery.
5. Insert the 6100 Computer into the dock.
6. As the 6100 Computer starts the boot process, you will hear one of the following beep signals:
   - one beep: boot files are moved into shadow RAM from an SRAM card in drive B: (master mode boot cycle)
   - two beeps: shadow RAM receives files through a download process through a communications port.
   - three beeps: shadow RAM receives files from flash memory.
7. Verify that the 6100 Computer has booted. It should show a box with “NORAND UTILITIES” at the top.

   If this is not the case, then press the reset switch on the 6100 Computer. The reset switch is located under the top lid, to the left of the PC Cards, near the front eject button (slot 1). For a diagram showing the location of the reset button and PC Card drives, refer to Figure 1-1, page 1-17.

8. After the initial double beep, hold the [I/O] key down until the following ROM DOS 5 “Start from” menu comes up.

```
ROM DOS 5

Start from:
1) Memory card 1 =A:
2) Memory card 2 =B:
3) RamDrive      =C:
4) RomDrive      =D:
```

9. Press the “4” key to choose option 4. The 6100 Computer should boot to the Norand Utilities screen.
10. Press YES at the Norand Utilities copyright screen to open the main menu.

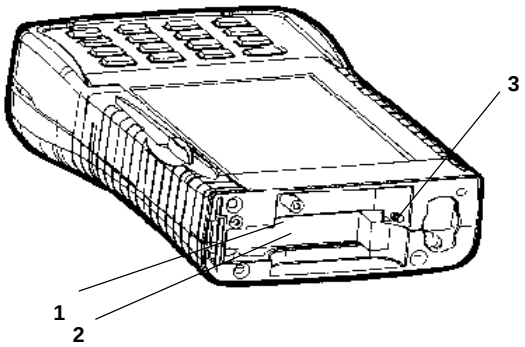
    The Norand Utilities program supports a variety of communication options for loading application programs. Each option is explained later.

    Press option 2 to select a communication option from a drop down menu, then press option 1 from the main menu to initiate a communication session. For getting started, the most common option is to use INTERSVR.

    For a complete description of navigating the screens in the Norand Utilities program, refer to the *PEN*KEY Model 6100 User’s Guide*.
11. Prepare the CONFIG.SYS and AUTOEXEC.BAT files for initial program load. For an example CONFIG.SYS file or an example AUTOEXEC.BAT file, refer to *Appendix A, Sample Configuration Files*. At a minimum, the CONFIG.SYS file needs the following lines:

```
device=d:\elanapm.exe
device=d:\nordospm.exe
```

    The content of these configuration files depends on the type of application being loaded. Examples for DOS and Windows can also be found on the Tool Kit diskettes.
12. Using one of the communication options, described later in this section, transfer the application program files to the 6100 Computer. Following a successful communication session, the Norand Utilities program executes the application program files.



1. Drive B: (upper) PC Card slot
2. Drive A: (lower PC Card slot
3. Reset switch

Figure 1-1
Location of Reset Button and PC Card Drives

Development Resources

This paragraph covers the following topics:

PC Card Support

NORAND Card and Socket Services: ELANCSSS.EXE

Support for the AMD ELAN PC card (accessory card) controller is provided via the NORAND Card and Socket services product, ELANCSSS.EXE. This product is configured, integrated, and tested in the 6100 Computer, and supports ATA and COM cards on the 6100 Computer. It is used in conjunction with PC card device drivers that use card services.

For a listing of the full PC Card socket and card services support, refer to the RELNOTES.TXT file in the Tool Kit.

ELANCSSS is used if you need to access ATA or modem cards.

If you are using ATA cards, you must also load ATABIOS.SYS, then NORATA.SYS, in that exact order.

For details of ATABIOS.SYS device driver, refer to the *ATA BIOS* paragraph, in the *Conversions and Interfaces* section.

If you are using a modem (landline or wireless), then you need to load NORMOD.SYS.

For details of NORMOD.SYS device driver, refer to the *Docks and Modems* paragraph, in the *Communications and Device Support* section.

EXAMPLE: The following is an example for ELANCSSS.EXE usage:

```
device=d:\norcs\elancsss.exe
```

EXAMPLE: The following are examples for SanDisk usage:

```
device=d:\norcs\atabios.sys /r
device=d:\norcs\norata.sys
```

EXAMPLE: The following is an example for PC Card modem usage:

```
devicehigh=d:\norcs\normod.sys /c4 /s /d 16b 21 20
```

where:
`/c4` = install as COM4
`/s` = save and restore UART values for COM4 on SUSPEND/RESUME
`/d m n t` = delay t ticks (where `m` = man code and n = man info)
`16b 21` = refers to the Erickson PIA radio

NORAND Card and Socket services can replace CardSoft completely.

Under the FORMAT RAM DRIVE option, the drive letter of the RAM drive is determined by its media type. SRAM cards are recognized as drives A and B. ATA cards are recognized as drives E and F. If you select the option to format an SRAM card, when an ATA card is actually in the slot, you will get an Error code 4 (invalid media).

NORAND PC Card Files

The following list contains a brief description of each of the files in the integrated and tested distribution:

Table 1-1
NORAND Card and Socket Files

File	Description
ATABIOS.SYS	A PC card device driver. It identifies and configures cards with a device ID of 0Dh and a function code of 4 (FIXED_DISK_CARD) and takes over the INT 13 interface for these cards. It is used with NORATA.SYS (see the *ATA BIOS* paragraph, in the *Conversions and Interfaces* section, for details.
ELANCSSS.EXE	The NORAND Card and Socket services product. It is smaller, faster, less costly, and better power coordinated than SystemSoft. The SystemSoft services are also included with the 6100 Computer.
NORATA.SYS	A DOS block device driver for removable hard disk media, for SanDisk support.
NORMOD.SYS	Loaded when you want to use a modem. This includes landline and wireless modems.

SystemSoft Card and Socket Services

CardSoft files (by SystemSoft) were used in early releases, and can continue to be used. The SystemSoft files are available in the D:\CS directory on the 6100 Computer. They are not loaded, by default. The following is a list of those files and their descriptions:

Table 1-2
NORAND Card and Socket Files

File	Description
ATADRV.EXE /s:2	A block device driver that supports ATA type II flash or ATA type III hard disk PC Cards. ATA drives are allocated the drive letters of E: and F: when this driver loads. The /s:2 option installs this ATA driver as a slave to the memory technology driver (MTDDRV.EXE) that follows, thus preventing the creation of new drive letters as additional memory technology drivers are installed. With this approach, all memory technology can be referenced as E: and F: for PC Card slots 0 and 1.
ATAINIT.EXE	is a low-level format for SanDisk cards. It places a hard disk partition table onto a blank ATA drive. This utility initializes the ATA drive as FDISK does with a standard hard drive. ***Note: use with CardSoft drivers only***
CARDID.EXE	Client device driver for the configuration of the PC Card slot/adapter and card on card insertion detection. Also frees allocated system resources upon detection of card removal.
CARDINFO.EXE	helps diagnose problems with cards by verifying if the card is recognized. It is a DOS utility that scans the PC Card slots on the system and lists information about the cards in the slots. It also lists any warnings or error messages that may have occurred during the configuration of the cards by the CardSoft drivers. ***Note: use with CardSoft drivers only.***
CS.EXE	Card services driver to manage the configuration of the system PC Card resources.
NORCSAPM.EXE	Enabling TSR for communication between power management and card services software for the correct processing of suspend/resume requests.
CSALLOC.EXE	Resolution and assignment of system memory, I/O port, and IRQ resources for card services.
MTDDRV.EXE	Memory technology driver that supports PC Card SRAM cards through the MTSRAM.EXE driver and allows for the sharing of letters between the different types of memory cards (SRAM and ATA).
MTSRAM.EXE	Memory technology driver for PC Card SRAM cards.
SSELAN.EXE	Standard Socket Services driver interface to the ELAN host controller chip.

Table 1-3
Initialization Files

File	Description
CARDID.INI	Default settings for card identification
CSALLOC.INI	Used by CardSoft to determine which system resources may be used by PC Cards

The following card libraries are for the support of modems and ATA drives:

Table 1-4
Card Libraries

File	Description
GENATA.CLB	Generic ATA drive library
SUNDISK5.CLB	SanDisk SDPL5 5-meg ATA drive library
GENMODEM.CLB	Generic modem library

These drivers provide audio feedback when a card is inserted or removed from the card slot. When the 6100 Computer recognizes that a card is inserted, it sounds like "boo-BEEP." When a card is removed, it sounds like "BEE-boop."

Before the Norand Utilities program is executed, each accessory card slot (E: and F:) is checked. If a card is present and contains an AUTOEXEC.BAT file, that drive is made the current drive and AUTOEXEC.BAT is processed immediately. No Norand Utilities program screens are seen.

SanDisk Support

To support SanDisk cards loaded with STACKER, the CONFIG.SYS file in flash loads STACKER.DRV (if it is present). In this case, the AUTOEXEC.BAT file on the stacked volume of the card is processed.

Note that the Norand Utilities program does NOT process a CONFIG.SYS file on the card. If the card is an SRAM card, it is possible to change the default boot drive to be the card rather than the flash in order to process the CONFIG.SYS file on the card.

If an application requires a CONFIG.SYS file different from the one in the flash, the AUTOEXEC.BAT file on the card must copy the new CONFIG.SYS file to the RAM drive and reboot from the RAM drive. Any drivers specified in the new CONFIG.SYS that are not in the flash must also be copied to the RAM drive.

To access a SanDisk card loaded with Stacker from a laptop computer that supports PC Cards, you must do one of two things:

- Install the Stacker device driver on the laptop. This is done by running SINSTALL from the SanDisk drive. SINSTALL is on the unstacked portion of the SanDisk drive; or
- Run Stacker Anywhere, which is a TSR shell that accesses the stacked portions of the SanDisk card without needing to modify the CONFIG.SYS file. Start Stacker Anywhere by typing "STACKER" and exit by typing "EXIT."

▶ **NOTE:** *The version Stacker that is shipped on the SanDisks is incompatible with Windows 95. If your development environment is Windows 95 and you need to leave Stacker on the SanDisks, then when you copy files over to the SanDisk, you need to boot a previous version of DOS in order to access the card.*

When loading Stacker and INTERLNK, be sure that Stacker is loaded before INTERLNK in the CONFIG.SYS file. Also, in order to write files to the SanDisk card using INTERLNK, both the 6100 Computer and the development PC need to have Stacker loaded. If a SanDisk card becomes corrupted by INTERLNK, it can be restored by reformatting it using FORMAT.COM from DOS 5 (also supplied in the Programmer's Tool Kit).

▶ **NOTE:** *All data will be lost when the card is formatted.*
Also note that you will need to boot a previous version of DOS in order to run INTERLNK, if you have a Windows 95 development environment.

Communications Via INTERLNK and INTERSVR

INTERLNK is a device driver that interconnects a PEN*KEY 6000 Series Computer and a PC, via serial ports. INTERSVR is the INTERLNK server, a communication option in the Norand Utilities program. These two resources are provided with ROM DOS 5 and are shipped with the 6100 Tool Kit. The cable that connects the PC to the 6100 Computer is a standard null modem cable. A TTY TCOM cable also works. And, you need a dock for the 6100 Computer.

INTERLNK causes the 6100 Computer drives to appear as virtual drives on the PC, with drive letters that are immediately beyond the highest drive letter currently used on the PC. Typing "INTERLNK" from the PC command line displays the designations of the redirected drives. For more details relating to INTERLNK and INTERSVR topics, refer to the DOS online help text.

The following instructions assume that when you installed the 6100 flash, all files from the archive were placed in C:\PENKEY\FLASH on your host PC. If you have files in a different location, adjust these instructions accordingly.

Read all instructions below before proceeding.

INTERLNK

INTERLNK, which is part of MS-DOS supplied with your PC operating system, can be installed on a PC, using the following statement in the CONFIG.SYS file on your host PC:

```
device=c:\dos\interlnk.exe /drives:4
```

The above statement assumes that MS-DOS is located in the C:\DOS directory on your host PC. The "/DRIVES:4" parameter allows mapping of four drives from your 6100 Computer. This statement should be inserted at the end of the CONFIG.SYS file (after any other statement that creates a drive letter).

After rebooting your host PC, you can copy the application files to the 6100 Computer, with INTERLNK.

▶ **NOTE:** *If you are running Windows 95 on your 6100 Computer, you may want to consider an alternative method of running INTERLNK/INTERSVR, such as booting to an earlier version of DOS.*

INTERSVR

Begin with the 6100 Computer at the NORAND UTILITIES LOAD PROGRAMS/ DATA menu. It may be necessary to force the unit to start from the ROM Drive (D:) by pressing the reset button, then pressing [I/O] while the start-up messages are being displayed. This brings up a menu that allows selection of the start-up drive. Select [4] to start from the ROM Drive (D:).

1. Press Enter at the main menu to advance to the LOAD PROGRAMS/DATA menu. Select Option 9 "ADVANCED UTILITIES".
2. Select Option 3 "FORMAT RAM DRIVE".
3. Create a 2 MB (or larger) RAM Drive on 6100 Computer. After completion, return to the LOAD PROGRAMS/DATA menu and select option 2, COMM. From the COMM menu select option 6 INTERSVR. Now select option 1, BEGIN COMM SESSION.
4. Establish the INTERSRV connection to your desktop/laptop PC, using a PEN*KEY single dock and cable, IrDA Dongal or direct serial cable.
5. Copy all files in C:\PENKEY\FLASH to the RAM Drive (C:) on 6100 Computer. To determine what drive on your PC corresponds to the 6100 RAM Drive, execute INTERLNK at the DOS prompt, as follows:

   ```
   C:\WIN>INTERLNK
   ```

You will see a chart similar to the one below. In this example, you would copy the files to drive H:.

```
Port=COM2

This Computer         Other Computer
   (Client)                (Server)
---------------       ------------------
 F:                        equals    A:
 G:                        equals    B:
 H:                        equals    C:
 I:                        equals    D: (519Kb)
```

6. Once the files are transferred, exit InterSvr on 6100 Computer by pressing [Esc]. The unit automatically resets at this time.
7. When the unit resets, it begins the flash update process. You are prompted to put the unit on charge and press a key. And then you are prompted to press the “3” key to begin the reflashing process.
8. After reflashing, the 6100 Computer resets itself and boot back to the default drive. To confirm the Flash version, observe the screen while booting from drive D. If necessary, refer to the instructions above for overriding the default boot drive.
9. You must now remove the flash files from your ramdrive by following the procedures above for formatting a ramdrive or by starting another interlink session and deleting the files manually.

To terminate INTERSVR, press the ESC key. See the release notes for information on changes in this version of Flash.

RAM Drive

RAMDFMT.EXE creates a RAM Drive on the computer. This RAM drive is a block of system memory that is treated as DOS drive C:. The data stored on the RAM drive is maintained as long as power is supplied to the system and the drive is not reformatted.

When a RAM drive is created, this results in less extended memory available for programs. No additional conventional memory is required for the RAM drive. Because this support for accessing the RAM drive is embedded in the DOS BIOS, no additional drivers need to be loaded. The extended memory test (with the corresponding boot messages) reserves memory for the RAM drive. Only one RAM drive can be created.

The size selected for the RAM drive represents the total memory required by the RAM drive, not the free space that is available on the drive. DOS file system overhead reduces the specified memory by the size of the file allocation table (FAT) and root directory created.

If CONFIG.SYS loads HIMEM.SYS and sets DOS=HIGH, the first 64 KB of extended memory is used for the High Memory Area. As a result, the maximum size of the RAM drive is 64 KB less than total extended memory.

The following switches are allowed for RAMDFMT.EXE. All values are in decimal. You must specify either size or address, but not both.

–a **a**ddress of RAM disk (in kilobytes)
–d **d**rive number (0 = A, 1 = B, ... etc.)
–e number of directory **e**ntries
–s **s**ize of RAM drive (in kilobytes)
–h **h**elp

▶ **NOTE:** *Reboot the HHC after the RAM drive is created (or altered), to make the change effective.*

If a communications option other than Accessory Card is selected, the Norand Utilities program automatically creates a RAM drive if one does not already exist. If the accessory card option is selected, the startup files on the card must create a RAM Drive if one is needed.

For a system with 6 MB (or more) of total memory, Norand Utilities creates a RAM drive of size 4 MB. For lower system RAM capacities, only 1 MB of RAM is reserved for conventional use and the remainder is allocated to the RAM drive.

The Norand Utilities program provides a menu that allows the RAM drive to be formatted in 1 MB increments.

The RAM drive created by Norand Utilities allocates 128 directory entries, that is, it specifies –e128 as a parameter to RAMDFMT.EXE.

When NPCP, TTY, NRInet, TFTP, TCP/IP bootp, or Novell communication options are used, another choice exists for creating a RAM drive. If the RAMDFMT.CTL file is downloaded, Norand Utilities executes the program RAMDFMT.EXE to create a RAM drive. RAMDFMT.CTL is a text file that must contain the command-line parameters for RAMDFMT.EXE.

For example, if RAMDFMT.CTL contains the –s2048 line, then Norand Utilities executes "RAMDFMT.EXE –s2048", which creates a 2 MB RAM drive.

When processing RAMDFMT.CTL, if RAMDFMT.EXE cannot be executed or returns an error, the communication session fails with error "F 0." Also, if the parameters in RAMDFMT.CTL cause the RAM drive to be removed (such as "–s0"), the communication session fails with error "F 0."

Norand Utilities and Communications

Norand Utilities

For a description of the Norand Utilities, refer to that topic in the *Communications and Device Support* section.

Creating a (Host) Download Include File

Most of the communication options for performing an initial program load require a download include file on the host computer. However, to create a download include file, you first need to create a download list file.

Create a download list file, as follows:

> A download list file contains the names of the files to be sent to the 6100 Computer. Each line in the file specifies one name. The file names may include path information so that all of the files need not be in the same directory on the host. However, the file is transmitted without the source path information, and placed in the current directory of the 6100 Computer.
>
> To specify a destination path or to rename a file on the 6100 Computer, use a line with the following format:
>
> `"newname=oldname"`
>
> This causes the file "`oldname`" to be read from the PC and transmitted to the 6100 Computer with the name "`newname`". Both "`oldname`" and "`newname`" may include path information.

EXAMPLE: The following is a sample download list file:

```
CONFIG.SYS
ROMINIT.BAT
6X00DOS\PC4800.SYS
MYAPP.EXE
```

The following file names cannot be used in a download list file because they are used by Norand Utilities to perform communications:

- NRTLOG.DAT
- NRUPLD.CTL
- PSROM0C.DAT
- PSROM0C.INI

Now, you are ready to create the download include file. Execute the following command:

```
IPLFMT.EXE <list file> <include file>
```

where:

<list file> is the name of the download list file created, above.
<include file> is 6100IPL.INC. This file is created by IPLFMT.EXE.

It should be noted that IPLFMT.EXE can be found in the Tool Kit.

IPLFMT.EXE concatenates files listed in the list file, inserting appropriate Norand file headers for download. Refer to *TCOM Session Overview*, in the *Communications and Device Support* section, for file header descriptions.

NPCP Network

NPCP is the NORAND Portable Communications Protocol. This protocol is supported by the 492x Tcom Packages, 6920 Communications Server, 498x Network Communication Controllers, and 6980/6985 Network Managers. You would substitute the name of your include file for the text, "<include file>."

1. Create a download include file, as described in the preceding paragraphs.
2. Create either Boot Disk, if necessary:
   - **4920 Boot Disk:**
     The only file on a 4920 Boot Disk is a download include file. The include file is copied to the DOWNLOAD subdirectory of the 4920.
   - **4980 Boot Disk:**
     Be aware there is limited space on the 4980 System. Place the 4980 System Files on the 4980 Boot Disk in the same way as is done for 4000 Series applications (using the 4980 Boot Toolkit, for example).

     Execute the following to place the download include file on the disk:

     ```
     MD A:\DATA
     COPY <include file> A:\DATA
     NCDIR.EXE A:\DATA
     ```

NOTE: *4000 Series applications are copied to a BOOT directory.*
6000 Series applications are copied to the DOWNLOAD directory.

NOTE: *NCDIR.EXE is included in the 4980 Boot Toolkit.*

Norand Utilities Internals

The Norand Utilities program, 2.00 and later, first attempts a session to NORAND_SERVER, which allows the application to be retrieved from a 498x controller. If the application is not stored on the 498x controller, a session is initiated to NORAND_HOST, which connects to the host.

TTY

TTY is another NORAND proprietary protocol, also supported by the communications servers.

- Create a 4920 boot disk, as described in the preceding paragraphs for the NPCP Network.

NRInet

NRInet is a protocol that performs a Norand file transfer session over TCP/IP Ethernet. It is supported by the 6920 Communications Server.

1. Create a download include file, as described in the preceding paragraphs, and place it in the download directory of the 6920 Communications Server.
2. Optionally, configure a DHCP server to provide information required by the hand-held computer, including the IP addresses of the client, router, and subnet mask. Any information not provided by DHCP must be entered manually by the user on the hand-held computer. A DHCP server can also provide a domain name and IP addresses of domain name servers, which allows you to enters a host name rather than an IP address. If the SERVER_NAME field on the hand-held computer is left blank, a connection to the name “Norand6920” is attempted, as a default.

Version 2.xx of the Norand Utilities program creates a NET.CFG file and a PCTCP.INI file on the RAM drive. These NET.CFG and PCTCP.INI files are not erased, so they may be used by applications.

TFTP

TFTP (Trivial File Transfer Protocol) is a standard TCP/IP protocol supported on many TCP/IP servers.

1. A TFTPD service must be running on a TCP/IP server.
2. Optionally, configure a DHCP server to provide information required by the hand-held computer (IP addresses of the client, router, and subnet mask). Any information not provided must be entered manually on the computer. A DHCP server can also provide a domain name and IP addresses of domain name servers, which allows the hand-held computer user to enter a server name rather than an IP address. If the SERVER NAME field on the hand-held computer is left blank, a connection is attempted to the name “NorandTftp” as a default.
3. Create a download list file, as described previously, and place it in the default working directory for the tftpd service. The list file must be named <workgroup>.BCF where <workgroup> is the value of the WORKGROUP field in the UNIT ID menu.

Other Intermec Software

DOS Device Drivers

The following are DOS drivers, for screen rotation, power management, pen activity, scanning, etc.

Table 1-5
DOS Device Drivers

Driver	Description
4000API.EXE	TSR needed to provide 4000 compatible API interfaces, such as the DOS NPCP Printer driver
61MOUSE.COM	DOS Pen driver
61PODSCN.EXE	DOS Scanner for the pod scanning method
61THRSCN.EXE	DOS Scanner for the tethered scanning method
ATABIOS.SYS	A DOS block device driver for removable hard disks (e.g., PC Card ATA drives)
CALIB.EXE	Pen alignment utility for DOS
ELANAPM.EXE	APM BIOS installation for DOS
ELANCFG.EXE	ELAN configuration utility for power management
ELANCSSS.EXE	NORAND ELAN card and socket driver, for supporting ATA and COM cards on the 6100 Computer.
FONTSEL.EXE	NORAND DOS font selection utility (used with VROTATE)
IRDAPDRV.EXE	DOS IrDA Printer driver
NORATA.SYS	NORAND ATA card support (such as SanDisk)
NORDOSPM.EXE	NORAND DOS APM driver
NORMOD.SYS	NORAND PC Card modem support
PC4800.SYS	Standard DOS 4800 Series Printer driver
PRDRV.SYS	DOS IrDA Printer driver
VROTATE.EXE	DOS portrait mode display driver (text only)

Windows Device Drivers

The following are drivers, for screen rotation, power management, scanning, etc.

Table 1-6
Windows Device Drivers

Driver	Description
6100DISP.DRV	Windows CGA display driver
61MAG.DRV	Windows magnetic stripe reader driver (see note, below).
61SCAN.DRV	Windows Scanner driver
NOR4800.DRV	Standard Windows 4800 Series Printer driver
NOR6805.DRV	Standard Windows 6800 Series Printer driver
NORIRDA.DRV	Windows IrDA Printer driver
NORNPCP.DRV	Windows NPCP Printer driver
NORWINPM.DRV	Windows APM driver
PENALIGN.EXE	Pen alignment for Windows
UCLKPEN.DRV	Windows Pen driver

▶ **NOTE:** *Be aware that the Magnetic Stripe Reader (MSR) software is still evolving. It reads most common codes, but is currently being modified to work with some of the less common codes. Which means, if you write an application for the current MSR, then when a new MSR is released at a later date, your application will be tied to the way the old MSR works, and may not work with the new MSR.*

Utility Programs

ELANAPM.EXE: *APM BIOS Installation for DOS*

This is a DOS TSR (taking no parameters) that installs an APM BIOS for the 6100 Computer. This program is needed if you want to run NORDOSPM.EXE. ELANAPM.EXE and NORDOSPM.EXE could be called from the AUTOEXEC.BAT file. But the alternate and preferred method is to place the following statement in the CONFIG.SYS file.

```
device=d:\elanapm.exe
device=d:\nordospm.exe
```

CALIB.EXE: *DOS Pen Calibration*

This is a DOS pen-calibration utility. For more information, refer to the *NORAND DOS Pen Driver* paragraph, in the *Supporting DOS Applications* section.

PENALIGN.EXE: *Windows Pen Calibration*

This is a Windows pen-calibration utility. For more information, refer to the *NORAND Windows Pen Driver* paragraph, in the *Supporting Windows Applications* section.

DOS Batch File Enhancers

BOOTDRV.COM: *Determines Default Boot Drive*

This application was for use in batch files that have a requirement to determine which drive was the default drive when the system was first booted. It returns an error level indicating which drive is the boot drive: `1=A, 2=B, 3=C, ...`

This program performs two simple DOS calls:

```
mov  ax, 3305h   ; Get boot drive
int  21h         ; DL = boot drive
mov  al, dl      ; errorlevel = boot drive
mov  ah, 4Ch     ; Exit( errorlevel );
int  21h
```

Batch file example:

```
BOOTDRV.COM
SET BootDrive=A:
IF ERRORLEVEL 2 SET BootDrive=B:
IF ERRORLEVEL 3 SET BootDrive=C:
IF ERRORLEVEL 4 SET BootDrive=D:
```

DD.EXE: *Disk Duplicator*

This utility copies SRAM and diskette images.

DELAY.EXE: *Display Message, Wait, Pause, Return Error Level*
Serves four functions:

- Display a message to the screen (like the Echo command)
- Wait for some amount of time to expire before continuing (like a Sleep command or DOS 6 Choice command with a time-out specified)
- Pause system execution until a key is pressed (like the Pause command)
- Return an error level based on which key, if any, was pressed (like the DOS 6 Choice command)

The first three functions can be performed even when DELAY is loaded as a device driver. Since the error level concept does not apply to CONFIG.SYS processing, the fourth function applies only to command line (or batch file) execution.

1. To display the message *Press any key to continue. . .*, no command line parameters are required. To display one or more different messages, supply the new messages in double quotes on the command line. Each quoted text string displays on a separate line. A null message (“”) can be used to display a blank line. To keep the cursor positioned immediately following the last character displayed, do not supply the trailing quote (”). To simply display a message without waiting for either a time-out or a key press, specify a delay time-out of zero (/0).
2. To force DELAY to exit after a period of time has elapsed, even if no key has been pressed, specify a “/nnnn” switch on the command line anywhere, even within quoted text. The value of nnnn is the decimal value for how many hundredths of seconds that must elapse before DELAY automatically exits. The maximum delay is about 640 seconds (or about 10 minutes).
3. No command line parameter is required to cause this program to wait for a key press. DELAY always exits whenever a key is pressed, even if it is waiting for time to elapse first.
4. When DELAY is executed from a command shell, as opposed to being loaded as a device driver, the error level set upon return is based on the key, if any, that was pressed prior to exiting. The return value is specifically geared to make it easy to tell which numeric key was pressed.

Table 1-7
DELAY.EXE Error Levels

Error Level	Key Pressed
0	‘0’
1	‘1’
2	‘2’
3	‘3’
4	‘4’
5	‘5’
6	‘6’
7	‘7’
8	‘8’
9	‘9’
10	‘:’
11	‘;’
12 to 154	The values returned consist of the key value minus 30h.
255	No key was pressed; the time expired first

Simple examples:

Standard PC Function	6100 Command(s) Required to Perform Function
Pause	DELAY.EXE
Echo “message”	DELAY.EXE /0 “message”
Sleep 10 seconds	DELAY.EXE /1000
Choice “message”	DELAY.EXE “message” IF ERRORLEVEL 3 GOTO InvalidDigit IF ERRORLEVEL 2 GOTO PressedTwo IF ERRORLEVEL 1 GOTO PressedOne IF ERRORLEVEL 0 GOTO PressedZero
Echo “message two” Pause	DELAY.EXE “message two” “Press any key to continue...”

FIXEMM.EXE: *Fix for EMM386 Memory Management*
This is a utility that fixes a CPU idling problem in EMM386.EXE. It hooks the APM CPU IDLE calls to prevent problems from using “HLT” with EMM386.

ISRAMDRV.COM: *Determine if RAM Drive Exists*
Returns an error level that answers the question, “Do I have a RAM drive?” If the error level is zero, then a RAM drive exists. This is for batch files that have a requirement to determine whether a RAM drive has already been formatted.
Batch file example:

```
ISRAMDRV.COM
IF ERRORLEVEL 1 GOTO NoRamDrive
IF NOT ERRORLEVEL 1 GOTO RamDrivePresent
```

MMBFLAG.COM: *Set/Get ROM DOS Boot Flags*
This program satisfies a couple of batch file requirements:

- The error level returned by MMBFLAG.COM indicates the current value of the ROM DOS boot flags. The boot flags keep track of the default boot drive and serve as an indicator of Master Mode Booting.
- Supply a command parameter to change the default boot drive. NORATA.SYS automatically determines which drive letter to supply, based on the media type. Drives E through H are also supported.

```
MMBFLAG.COM 0   (Drive A:)
MMBFLAG.COM 1   (Drive B:
MMBFLAG.COM 2   (Drive C:)
MMBFLAG.COM 3   (Drive D:)
```

Table 1-8
MMBFLAG.COM Error Levels

Error Level	Meaning
131	Master Mode Boot from drive B (default boot drive is D)
130	Master Mode Boot from drive B (default boot drive is C)
129	Master Mode Boot from drive B (default boot drive is B)
128	Master Mode Boot from drive B (default boot drive is A)
3	Drive D is the default boot drive
2	Drive C is the default boot drive
1	Drive B is the default boot drive
0	Drive A is the default boot drive

RESET.EXE: *Reset the System*
Provides the means for batch files or other software to reset the system.

Section 2

Supporting DOS Applications

Introduction

This section contains information about applications that run under DOS on the PEN*KEY® 6100 Hand-Held Computers.

Topic Summary

DOS Power Management Driver: NORDOSPM.EXE

Overview

NORDOSPM.EXE, which is part of the Advanced Power Management (APM) system when running under DOS, is the DOS power management driver interface between applications and the APM BIOS. NORDOSPM.EXE must be installed as a device driver at system startup time.

NORDOSPM.EXE gathers information from the APM BIOS and broadcasts it to APM-aware applications. It also monitors DOS function calls and notifies the APM BIOS when the system is busy. In addition, NORDOSPM.EXE allows the configuration of certain power management settings at installation time.

Figure 2-1 below is a simplified diagram of Power Management Software:

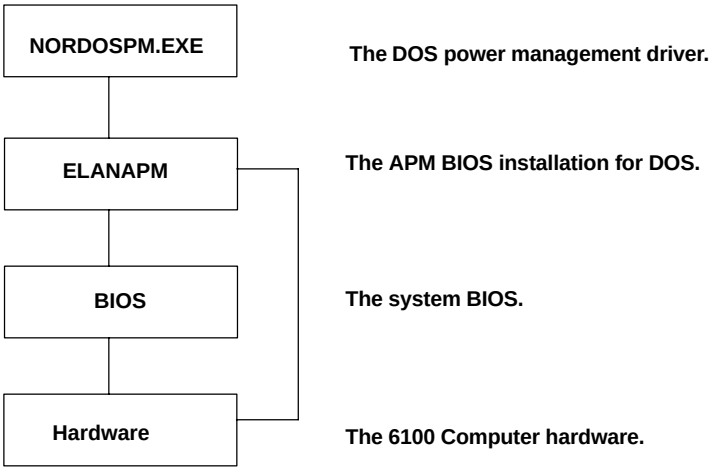


Figure 2-1
Power Management Software

Installation

NORDOSPM.EXE, which is installed as a device driver in the CONFIG.SYS file, requires ELANAPM.EXE and must be installed after ELANAPM.EXE. The following would be a typical entry in the CONFIG.SYS file:

```
device=elanapm.exe
device=nordospm.exe
```

Refer to the *APM BIOS Interfaces: ELANAPM.EXE* paragraph, in the *Conversions and Interfaces* section for a detailed description of ELANAPM.EXE, the APM BIOS driver.

Command Line Switches

Refer to the Microsoft documentation for the APM applications command line switches.

ELAN Configuration Driver: ELANCFG.EXE

ELANCFG.EXE is a DOS command line configuration utility that sets various APM options for the AMD ELAN processor. It is intended to be executed at start-up, but may be executed (or reexecuted) at any time.

Usage

```
ELANCFG.EXE <optional arguments>
```

Optional arguments may be any number of the following:

```
/? /An /Cn /Dn /En /Hn /In /Kn /Ln /Mn /Rn /Tn /Vn /Wx /S
```

Command Line Switches

The command line switches are described in partial detail, as follows:

/? **Help screen**: Displays brief command line switch descriptions.

/An **ATA Primary Activity Monitor**: Configures ATA/HDD access activity mask. Valid values for n: 0 and 1.
0= processor will time-out during ATA/HDD access.
1= count down timer is reset to zero when ATA/HDD activity is detected.

/Cn **Configure System Management**:
Valid values for n: 0, 1, 2, 3, 4, and 5.

1= Auto-Contrast enabled	0= disabled
3= Battery Hot Swap enabled	2= disabled
5= Suspend In Dock enabled	4= disabled
7= INTs are NOT activity in DOZE	6= INTs are activity in DOZE

/Dn **Set Time-out from Doze to Sleep**: Configures Doze mode timer. Valid values for n: 0, 4–1024 (nearest multiple of four seconds are used). Where n is the number of seconds between Doze mode and Sleep mode.
0= time-out is disabled. If any activity is detected during time-out period, processor goes back to High speed PLL mode and timer is reset to zero. For any other valid values, timer is set.

/En **AC Power Activity Monitor**: Configures External power (charge) activity mask. Valid values for n: 0 and 1.
0= processor does not count down or time-out, when external charge is detected.
1= processor continues to count down and time-out, while external power is detected.

/Hn **Set Time-out from High Speed to Low Speed**: Configures the High Speed PLL mode timer. Valid values for n: 0, 1–16 (seconds). Where n is the number of seconds between High Speed PLL mode and Low Speed PLL mode.
0= value disables the time-out. If any activity is detected during the countdown period, the timer is reset to zero. This may be thought of as the amount of time the processor spends in High Speed PLL Mode if no activity is detected. If the High Speed PLL timer is disabled, the processor stays in High speed mode, and never enters Low Speed PLL mode. The Low Speed PLL mode count down timer starts when the processor enters Low speed PLL mode. Therefore the processor never enters Low Speed PLL mode, Doze, or Sleep mode when the High Speed PLL timer is disabled. The same is true of all other successive time-outs. The High speed PLL mode timer starts counting down when no activity is detected, and reset to zero when activity is detected.
All other valid values set the time-out.

/IOxn **Set Activity to I/O Address**: Configures the Programmable I/O range activity mask. Valid values for n: 0 and I/O addresses in the range of 001h to 3FFh.
0= programmable I/O device activity does not reset the count down timer.
Otherwise, n is the 8 bit starting I/O address that is checked for activity. The address range extends to 7 addresses past the starting address (8 total). If activity is detected in this I/O range, the count-down timer is reset to zero.
(e.g., /IOx380 sets up I/O addresses 380h through 387h as activity).

/Kn **Keyboard Activity Monitor**: Configures Keyboard interrupt activity mask.
Valid values for n: 0 and 1.
1= processor continues to count down and times-out during keyboard activity.
0= timer is reset to zero when ever keyboard activity is detected (at COM4, port 2Eh).

/Ln **Set Time-out from Low Speed to Doze**: Configures Low Speed PLL mode timer. Valid values for x: 0, 1–16. Where n is the number of seconds between Low Speed PLL mode and Doze mode.
0 = disables the time-out.
For any other valid values, timer is set. If any activity is detected during the count down period, the processor goes back to High Speed PLL mode and the timer is reset to zero.

/Mn COM4 Activity Monitor: Configures Modem activity mask. Valid values for n: 0 and 1.
0= count down timer continues during modem access.
1= count down timer is reset to zero when modem activity is detected.

/P **UNUSED**

/Rn Configures Ram drive access activity mask. Valid values for n: 0 and 1.
0 = count down timer continues during ram drive access.
1 = count down timer is reset to zero when Ram drive access is detected.

/S **UNUSED**

/Tn Configures Timer ticks while in DOZE mode.
0 = no timer ticks occur while in Doze mode.
1 = timer ticks occur while in Doze mode.
2 = Extended timer ticks occur while in Doze.

/Vn Configures the Video memory write activity mask. Valid values for n: 0 and 1.
0 = the processor continues to count down and time-out during video memory access.
1 = timer is reset to zero whenever video memory access is detected.
Note that Windows frequently accesses video memory to repaint screens, etc. The processor will probably not time-out in Windows unless the Video activity mask is set to zero.

/Wx Configures the programmable Resume (Wake-up) event mask. Valid values for x: R, 0, 3, 4, and 8.
R = a ring-in causes the processor to wake up.
0 = all resume masks are disabled.
3 = IRQ 3 causes the processor to wake up.
4 = IRQ 4 causes the processor to wake up.
8 = IRQ 8 causes the processor to wake up.

Power States

ELANCFG.EXE has four power states:

High	The processor is running full speed
Low	The processor is running at a reduced speed
Doze	The processor is halted -- the backlight turns off, if it was on; the display is still up and the touch is still up; power is still supplied to terminal and external devices.
Suspend (sleep)	The processor is halted -- the display turns off; power to PC cards is turned off; power to communication devices is turned off; power to the touch screen and keyboard has some power on. The only way to resume/wake the unit is through the I/O button (Suspend/Resume) or the wakeup events.

The transition times for power states can be customized through the use of ELANCFG.EXE. The settings concerning time-outs are:

/H	the time at High speed before going to Low speed
/L	the time at Low speed before going to Doze
/D	the time at Doze before going to Sleep

For example, if you want to set the backlight to be on for 30 seconds, and then suspend the unit at 60 seconds — one way to configure this is as follows:

```
elancfg.exe /H10 /l20 /d30
```

This keeps the 6100 Computer operating at High speed for 10 seconds. Then, if no activity is perceived at 11 seconds, it switches to Low speed for 20 seconds. Then, if there is still no activity perceived (after that 20 seconds), it switches to Doze and the backlight turns off. Finally, after 30 seconds has passed, the unit switches to Suspend.

Also, you can restrict the 6100 Computer from reaching one of the lower states if a parameter of 0 (zero) is used at any of these levels. That means the 6100 Computer is “locked” at that power state, and cannot go to lower states.

For example, if you have a PC Card radio that needs to be active all the time, then you might consider restricting your power management to not use the Sleep mode. By using the following call, you disable the 6100 Computer from auto-suspending, and let it run at high speed for 10 seconds and, at low speed for 10 seconds, then remains in Doze mode indefinitely until activity is perceived, a Powerfail event occurs, or a uses request suspend occurs:

```
elancfg /H10 /L10 /D0
```

Obviously, these settings have a large impact on your battery life. So they should be made with battery needs in mind.

DOS Pen Driver: 61MOUSE.COM

Overview

The pen driver is actually a DOS mouse driver. It interprets the standard interrupt 33h mouse interface. Not all mouse functions are supported. Refer to the *Conversions and Interfaces* section, in the *Standard Mouse interface: INT 33* paragraph, for a list of functions available.

Also refer to the *Ralf Brown's Interrupt list* for detailed information relating to the *Standard Mouse Interface: INT 33* (see the *Reference, Open Systems Publications* section for ordering information).

Installation

Ensure the following files are located in the root directory on the 6100 Computer:

File	Directory
ELANAPM.EXE	\ (root)
61MOUSE.COM	\ (root)
CALIB.EXE	\ (root)

The pen driver installs as a TSR as a result of placing the following statement in the AUTOEXEC.BAT file:

61MOUSE.COM

If the drive and directory for this driver is not in your path, include that information, as follows:

```
d:\path\61MOUSE.COM
```

where:

d: is the driver and *\path* is the directory path to the driver.

After installation, the driver displays the following message:

```
Driver installed
6100 Digitizer enabled
```

▶ **NOTE:** *Always load IrDA driver before the pen driver.*

DOS Pen Calibration: CALIB.EXE

Overview

CALIB.EXE is the DOS Pen Calibration utility. It performs calibration of the pen interface for the 6100 Computer, as well as other PEN*KEY computers.

Calibration is simply the alignment of the location of the cursor to the same location where a finger or pen is placed on the touch screen. The 6100 Computer comes with default settings that are normally acceptable for most applications. However, some applications, such as signature capturing, may require a calibration adjustment to compensate for variations in touch panel alignment and user preferences.

Configuration

Configuration settings are stored in CMOS. If CMOS contains invalid information, the program displays the calibration screen, regardless of the existence of the calibrate command–line option.

CALIB.EXE is a DOS application program and can be invoked from the AUTOEXEC.BAT file or from the DOS command line. The appropriate mouse driver must be installed. The driver contains extensions to the standard Mouse API which provides digitizer information, screen resolution, and raw coordinate points, which is required for this implementation of CALIB.EXE.

Required Calibration Files

```
CALIB.EXE   (calibration utility)
N6100.BGI   (BGI driver)
```

▶ **NOTE:** *See **BGI Support** in **Appendix A, Sample Configuration Files** for BGI driver information*

Usage

CALIB.EXE begins the calibration process by placing one of four targets at a fixed location on the screen and waits for the user to press the pen at the center of the target three times. It is important during this process to apply moderate pressure while pressing the target. Lightly tapping or placing too much pressure generates incorrect results. The unit beeps after each successful press of the pen. Therefore, if a beep is not heard when the pen is down, lift and press again. Also use care to not allow fingers or any other object to touch the screen. When the program has acquired three good data points for the target, it then displays the next target.

When data for all four targets are acquired, the program then displays a screen which checks the calibration results. Check the match of the cursor with the location of the pen near the four corners of the screen. The best way to do this is to place the pen down near each corner of the display and see if the cursor appears at the location of the pen. If the cursor is offset from the pen location by some fixed amount, adjust it by pressing the appropriate UP, DOWN, LEFT, or RIGHT buttons on the screen. (The user may prefer to add some amount of offset to place the cursor away from the pen using the same buttons.) If the cursor position varies significantly from the pen location, as the pen is placed at the different corners, the four target calibration process was not successful. Restart the program and pay closer attention to how the pen is pressed against the targets.

When satisfied with the calibration, press the SAVE button on the screen to store the results. The results are then available for use by the pen/mouse driver. You may choose to not save the results by pressing the CANCEL button. In either case, the program exits. If for some reason the calibration program is unable to recognize the pen, you can press the Escape key and the program exits. In this case, the new calibration values are not stored.

DOS Scanner: 61PODSCN.EXE, 61THRSCN.EXE

Overview

61PODSCN is the pod scanner. 61THRSCN is the tethered scanner.

The pod scanner collects and passes scanned data to an application via the standard DOS type-ahead buffer, where the scanned bar code is available to an application as simple keystrokes. The program also manages power for the scan operations. The tethered scanner behaves just like the pod scanner, except:

- I/O is through the serial lid;
- There is no way to reprogram a key as a trigger;
- Does not display anything on the screen;
- Can be used if you need a fully decoded RS-232 scanner.

61PODSCN is generally installed from AUTOEXEC.BAT by this statement:

```
61PODSCN [-option[value]] ...
```

The program must be installed after the CardSoft drivers and TSRs are installed (if they are used) and after ELANAPM.EXE is installed. Also, if CardSoft drivers and TSRs are present in the system, the command line option on NORDOSPM.EXE that “fixes” some CardSoft bugs (/ss:1) must not be used on NORDOSPM.EXE, but instead used as an option to 61PODSCN.

Installation

Ensure the following files are located in the root directory on the 6100 Computer:

File	Directory
ELANAPM.EXE	\ (root)
61PODSCN.EXE	\ (root)
61THRSCN.EXE	\ (root)
BIOSDOT.COM	\ (root)

The driver, ELANAPM.EXE, needs to be loaded first, and BIOSDOT.COM needs to be loaded before 61PODSCN.EXE or 61THRSCN.EXE. ELANAPM.EXE should already be located on the flash, all of the other files listed above can be found in the Tool Kit.

BIOSDOT.COM handles printing a window to the screen “scanning” when the trigger is pulled. This program is only needed in a graphics environment, and not for a text environment

Configuration

Required CONFIG.SYS Entry

The following entry is REQUIRED in the CONFIG.SYS file, for DOS scanning:

```
device=elanapm.exe
device=nordospm.exe
```

Options

61PODSCN recognizes the following options:

Switches	Description
–? or –H	Either of these options results in a help screen that briefly describes the various options.
–A[n]	Enable aiming beam for 1/2 second intervals (if supported). The **n** is optional (as noted by the [brackets]), and if used, it specifies the number of intervals desired.
–E	Enable scanner immediately, and permanently, not just when enabled via software API.
–Knn	Define a keyboard key to be used to initiate the scanner, where nn = scancode for the key. For example: use –K0f for TAB key.
–On	Power off delay, where **n** is the number of seconds from release of the trigger to the time power is removed from the scanner.
–PLN	Use the PLN API interface.
–RESET	Resets scanner configuration to the factory default on start–up.
–SCANBIOS	Use the SCANBIOS API interface.

Usage

The DOS scanner uses command line switches to drive the desired functions.

For example:

```
61PODSCN –E –A2 –K16
```

where:

–E enables the scanner immediately and permanently.
–A2 enables the aiming beam for a one-second interval.
–K16 defines a keyboard key that can be pressed to initiate the scanner.

Example Scanner Application

This example scanner testware enables the scanner, then displays the DOS keystrokes until the return or newline character is encountered, where we exit.

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>
void main (void) {
   int c;
                        //
   _AH = 0x80;       // collect only one scan – 4500 compatible
                        //
   geninterrupt(0x7a);  // thread thru scan tsr
            //
   while ((c != '\n') && (c != '\r')) {
      while (kbhit()){
         c = getch();
         putchar(c);
         if (c == '\n' || c == '\r') break;
      }
   }
}
```

DOS NPCP Printing: PC4800.SYS

Overview

NPCP printing support under DOS consists of the DOS device driver PC4800.SYS, that allows DOS and PL/N applications written for the 6100 Computer to print to NORAND 4810, 4815, and 4820 Printers, using NPCP.

Driver Installation and Configuration

Required CONFIG.SYS Entry

PC4800.SYS is installed as a device driver in the CONFIG.SYS file.

For example:

```
device=pc4800.sys LPT1 1 /i1
```

If PC4800.SYS is not in the root directory, be sure to include the path.

The format of the command line for PC4800.SYS is as follows:

```
PC4800.SYS [<device name> [<port number> [/in ]]]
```

▶ **NOTE:** *The order of the command line parameters is important (for example, specify a `<device name>` in order to specify a `<port number>`.*

The following table lists the meaning of each of the parameters:

Parameter	Meaning
<device name>	The name that opens the device. This name can be anything except PRN, and can be up to eight characters long. The default name is NP4800.
<port–number>	The communications port number that the driver uses for output. Valid values are: 1 = COM1, 2 = COM2, etc. The default is COM1.
/in	This switch enables support for the interrupt 17h interface. This is needed for PL/N applications. 'n' is a digit that specifies the LPT port that accesses the device. For example, /I1 indicates that interrupt 17h calls for LPT1 are intended for this device driver. **Note:** The interface provided is not 100 percent PC-compatible. It is intended only to support PL/N applications. Non-PL/N applications should not use this switch.

Required AUTOEXEC.BAT Entry

PC4800.SYS requires that the 4000API.EXE TSR is loaded to function correctly. Refer to the 4000API documentation in the *Conversions and Interfaces* section, for installation instructions.

Functionality and Usage

Intermec Technologies Corporation supplies PL/N and C standard printer routines that handle critical errors and the printing of text to NORAND printers. The following information is for those who might want to create their own print routines.

PC4800.SYS must be opened before it can be used. For example, in the C programming language, you can use `fopen()`. Use the correct open call for the language being used that allows writing to the device. Also, the open call used should return an error if the device does not exist. Some open calls will create a file if the device driver is not present.

To write to the device, use any appropriate output function that can be directed to the device (for example, in C programming language, you can use *fprintf()*).

You should install a critical error handler that process errors from the printer. The printer driver returns only "device not ready" errors to DOS for the sake of compatibility with the standard PRINT program supplied with DOS. You can retrieve the extended error by bypassing DOS and calling the driver directly from the error handler.

Ioctl calls (DOS Function AH = 44h, AL = 02h) are supported by the device driver (refer to a DOS technical reference manual for detailed information relating to this function). Calls require the handle number of the printer device in register BX, a data item with the following structure pointed to by DS:DX, and the number of bytes to be read in register CX.

```
struct ioargs {
  far *unsigned char ioctl_cmd;
  far *unsigned char ioctl_buf;
};
```

The following commands may be pointed to by the ioctl_command field for execution by the driver:

```
ONLINE = 2
STATUS = 3
GET DEVICE FEATURES = 4
SELFTEST = 5
RESET  = 6
GET CURRENT CONFIGURATION =7
XTENDED ERROR = 255
```

Data is returned to the buffer pointed to by the ioctl_buf, for the number of bytes specified in register CX when the ioctl call is made. The first byte of the returned data represents any error encountered during the call. If the call was successful, this byte is set to zero. If the call was unsuccessful, this byte represents the extended error. The format of the rest of the data is described in the documentation for the printer being communicated with.

Notes

The printer driver must be opened and closed for each report transaction with the printer if you want to take advantage of the line-loss detection features of NPCP. Applications should status (ioctl 3) the printer after the last line of a report or a series of reports are sent to the printer to ensure that all lines were printed successfully by the printer.

The printer driver is multitasking. If the application needs to take full advantage of this feature, it must enable time slicing through the appropriate call to INT 15h (refer to the 4000API documentation).

If time-slicing is not activated, the last line buffered by the driver is not sent to the printer until the device is closed or an ioctl status call to the printer is made. You should perform a ioctl status call before all closes to ensure that the driver print buffer is sent to the printer successfully, since DOS closes to character devices do not perform output flushes as part of the close operation. The driver flushes the buffer for you upon a close, but no error is returned if the flush is unsuccessful.

DOS IrDA Printing: PRDRV.SYS, IRDAPDRV.EXE

Overview

IrDA printing support under DOS consists of the device driver, PRDRV.SYS, plus IRDAPDRV.EXE, the actual driver handler, link management wrapper, and protocol stack. Together these two software components form the IrDA printer solution, providing a standard DOS character driver to support IrDA printers.

The IrDA printer driver supports the following DOS device driver entry points, compatible with the standard input and output libraries provided by DOS compilers, handling the request as indicated by the summarized processes details:

open attempts discovery of an IrDA device, and if discovery is successful, establishes an IrLMP connection for a printer with the device discovered.

▶ **NOTE:** *Only one open connection is allowed at this time.*

read returns any data sent by the system. A return count of zero implies that no data is available.

write the written character is stored in a local coalescence buffer (with the IrLMP protocol header inserted) until the frame size is reached, at which time the frame is transmitted to the printer. If the –n switch is used, each user write byte count is sent with the IrLMP protocol header inserted without any attempt at local coalescence.

ioctl this processing is specific to Intermec Technologies Corporation and is intended to support printing applications. A printing program may hook the DOS INT 24 critical error interrupt and then call the driver's ioctl directly using the information provided by INT 24. The driver responds by setting an extended error code in the application's memory, using a pointer that was passed to it through the ioctl interface. The following ioctl call switch is currently implemented.

```
IOCMD_STATUS :                          //  0x03
   requestHeader.headerStatus = detailedErrorCode;
   break;
IOCMD_ID :                             //  0x80
   requestHeader.headerStatus = 0x0a;
   break;
default :
   requestHeader.headerStatus = detailedErrorCode;
   break;
```

where detailedErrorCode is one of the following:

```
 0  PREADY
    device driver is open -- a printer connection is established.
50  IRLAP_DISCONNECT_ERROR
    close failure -- a disconnect event state change failure occurred during the disconnect
    procedure processing.
51  DISCOVERY_FAILURE
    open failure -- no devices responded during the discovery procedure.
52  LINK_MANAGEMENT_CONNECT_FAILURE
    open failure -- a buffer flushing operation failed during the establishment of the link
    management session with a discovered device.
53  TX_BUFFER_DESC_GET_FAILURE
    write failure -- could not get a transmit buffer descriptor to send the data to the
    printer.
```

54 BUFFER_GET_FAILURE
write failure -- could not get a transmit buffer to send the data to the printer.

55 BAD_CONNECTION_HANDLE
open, write, or close failure -- a defective connection handle was encountered during the flushing of the driver's coalescence buffer.

56 UNKNOWN_TRANSMIT_FAILURE
open, write, or close failure -- the link access protocol failed for "unknown" reasons during an attempt to flush the driver's coalescence buffer.

57 WRITE_WITH_PRINTER_NOT_READY
write failure -- a write was attempted to the driver while the driver is in an error state.

58 I_QUEUE_FULL
open, write, or close failure -- the driver could not get a queue for the transmission of an information frame.

59 CONNECT_FAILURE
open failure -- after a successful discover of an IrDA device, a connection could not be established.

60 DRIVER_CLOSED
write, read, ioctl, close failure -- an attempt was made to access the driver while the driver was in a closed state.

61 NO_RECURSION
write, read, ioctl, close failure. An attempt was made to recursively access the driver.

62 BAD_LINK_CONTROL_FRAME_SIZE
open failure, link management failure -- the system did not receive the correct frame size for a connect confirmation by IrLMP after the IrLAP link session was established with the remote device.

63 LINK_MANAGEMENT_CONTROL_FRAME_EXPECTED
open failure, link management failure -- the system did not receive an IrLMP control frame from the remote device after the IrLAP link session was established.

64 NOT_LINK_MANAGEMENT_CONNECT_CONFIRM
open failure, link management failure -- the system did not receive an IrLMP connect confirm frame from the remote device after the IrLAP link session was established.

65 RECEIVE_FRAME_FAILURE
open failure, link management failure -- the system did not receive the IrLMP connect confirm frame from the remote device that had a successful IrLAP link established.

66 READ_FAILURE_PRINTER_NOT_READY
read failure -- an attempt was made to read from the character stream with the connection not in the ready state.

67 READ_FAILURE_PRINTER_IN_NDM
read failure -- an attempt was made to read from the character stream with the connection in the normal disconnect mode.

68 READ_FAILURE_BAD_CONNECTION_HANDLE
read failure -- an attempt was made to read from the core, and the connection handle used is no longer valid.

69 READ_FAILURE_UNKNOWN_STATUS
read failure -- an attempt was made to read from the core that resulted in the core posting an unknown error.

70 WRITE_REQUEST_EXCEEDS_COALESCING_BUFFER
write failure -- the user's write buffer is too large to fit in the local coalescing buffer and is being rejected. Present the data to the driver with a buffer size of less than 128 bytes at this time.

close The coalescence buffer is sent to the printer and, after having verified that all data is sent to the printer, the IrLMP and IrLAP disconnects are performed (if there is a coalescence buffer being used, see the –n switch for details).

Installation and Configuration

Make sure the following files are on the system:

```
IRDAPDRV.EXE
PRDRV.SYS
```

If your system was delivered with an application requiring these files, they should already be in flash. If not, they can most likely be found in the Tool Kit.

Required CONFIG.SYS Entry

The following entries are required in the CONFIG.SYS file:

```
device=d:\pathname\prdrv.sys
install=d:\pathname\irdapdrv.exe -t6100
```

where *d:\pathname* is your specific path to the directory where the printer drivers exist; and where *t6100* is the "technology" for the platform on which this driver is used.

Required AUTOEXEC.BAT Entry

There are no required entries in the AUTOEXEC.BAT file for IRDAPDRV.EXE.

Usage

Both PRDRV.SYS and IRDAPDRV.EXE have optional parameters.

```
PRDRV.SYS   [fileNameToUse]
```

where *fileNameToUse* is the name of the device to which this driver should respond. If this option is used, the f option (for IRDAPDRV.EXE) **must** be used, and the optional name selected **must** match this name string exactly. The default is IRDALPT.

```
IRDAPDRV  [ -? -bn -dn -fs -n -rn -t6100 -x ]
```

? provides information about version, revision level, a command line example, a list of the available switches, and brief descriptions of each.

▶ **NOTE:** *Use only the switches shown here. This driver is a multi-platform program, providing features for all of the 6000 Series platforms. None of the other switches are intended for the 6100 Computer and could cause a malfunction, if used on this platform.*

bn REMOVES a baud rate from consideration for data transfer rate negotiated during connect. The (n) part of this parameter is one of 19200, 38400, 57600, or 115200. This switch is accumulative and can REMOVE any, or all, of the above baud rates from the default negotiables. When the two systems negotiate, the largest common value is selected and that baud rate is used for communication, after completion of the negotiation.

dn specifies the disconnect time (in seconds) requested during negotiation. The (n) part of this parameter is one of 3, 8, 12, 16, 20, 25, 30, or 40. This switch is accumulative and can add any, or all, of the preceding disconnect times to the default of 3. The two systems negotiating select the largest common value.

fs defines the file name to be used by the driver. This string must be supplied, and must match the driver name option used with PRDRV.SYS when it was loaded (if used). The default name for this driver is IRDALPT.

n designates NO coalescing buffer support for devices that send data back to escape sequence.

rn specifies the number of discovery retries done at the DOS open command by the driver. For each (n) count, the user is allowed approximately 500 milliseconds to bring the system within range of the printer for data transfer discovery. The default is 2 retries.

ts identifies the 6100 Computer as the IrDA-equipped system, on which the driver is running. The required string is **`6100`**.

x specifies that the driver should disable interrupts not in the open state. This implies that the core is not discoverable and connectable, and therefore cannot perform that role.

Section 3

Supporting Windows Applications

Introduction

This section contains information about Windows Applications for the PEN*KEY® 6100 Computer. Included is a description of the minimal installation required for a Windows configuration, components required for this Windows installation, and a discussion of the *NORAND® Shell for Windows*, and some Windows applications that are designed for the 6100 Computer.

Topic Summary

NORAND Minimal Windows Installation

The following paragraphs describe the Windows 3.1 configuration on the 6100 Computer. Norand-specific issues are emphasized. General information on installing and configuring Windows 3.1 itself is available from Microsoft Corporation and other organizations.

▶ **NOTE:** *Before loading Windows, be sure to unload VROTATE.EXE, if previously loaded. Otherwise, the results could be unpredictable. To unload VROTATE, issue the VROTATE –d command.*

Installation

You can install Windows on a 6100 Computer by copying the Windows diskette to a memory card or by copying the files to the RAM drive. You can then start Windows by running the WIN.COM program. A table in subparagraph *Windows Components*, on page 3-3, lists the files provided by Intermec Technologies Corporation, as well as a brief description of each.

To illustrate a complete boot sequence from DOS into Windows, the Windows configuration also includes DOS startup files such as COMMAND.COM, AUTOEXEC.BAT, and CONFIG.SYS.

Windows Operating Modes

The Windows 3.1 minimal configuration is shipped with all the files necessary for operating Windows in both Standard and Enhanced modes. Where possible, you should run Windows in Standard mode. Standard mode can be configured into a much smaller footprint and can provide better performance. You can start Standard mode by using the "/s" option when running WIN.COM. For more information on files that can be removed from the configuration, if Enhanced mode is not required (refer to *Windows Components*, on page 3-3).

Configuration

The Windows 3.1 package, supplied by Intermec Technologies Corporation, is shipped with a default configuration appropriate for the 6100 Computer. It should require no further configuration to run.

Normal Startup (Standard Mode)

The Windows startup sequence is as follows:

1. MS-DOS boots and processes CONFIG.SYS and AUTOEXEC.BAT.
2. AUTOEXEC.BAT runs WIN.COM.
3. WIN.COM displays the Windows 3.1 logo and launches DOSX.EXE, the standard-mode DPMI server.
4. DOSX.EXE loads WSWAP.EXE and launches KRNL386.EXE, the Windows kernel for 32-bit processors.
5. KRNL386.EXE initializes GDI.EXE, USER.EXE, the system drivers (mouse, keyboard, etc.), and the installable drivers. During and after this step, the SYSTEM.INI file is referenced to locate drivers and driver configuration settings.

6. KRNL386.EXE now loads the Windows shell program. In this case, NORSHELL.EXE is loaded.
7. After referring to the WIN.INI initialization file, NORSHELL.EXE launches the user application, which, in this case, is the Windows 3.1 File Manager. Windows startup is now complete and the File Manager is open.

DOSX.EXE Startup

As mentioned earlier, you can start Windows 3.1 standard mode by running WIN.COM.

An alternative method is to run DOSX.EXE directly. If run directly, the WIN.COM and WSWAP.EXE files can be left out of the system to save space. Note that the standard mode configuration does not support execution of DOS programs. For a more detailed discussion of Windows startup, refer to chapter 1 of the book *Windows Internals*, by Matt Pietrek, published by Addison-Wesley.

Windows Components

The following tables list the Windows 3.1 files for 6100 Computer. The table titles indicate the category. Files marked Optional may be deleted from the configuration if the running applications do not need them.

Table 3-1
Windows Startup and Shell Programs

Component	Description	
WIN.COM	DOS program that launches Windows.	Optional
WINTITLE.RLE	Default desktop wallpaper. Displays Windows 3.1 logo.	Optional
NORSHELL.EXE	Norand shell program.	Optional
WINFILE.EXE	Windows File Manager shell, launched by NORSHELL.EXE.	Optional

Table 3-2
Initialization Files

Component	Description	
SYSTEM.INI	Initialization file for Windows system and drivers. [BOOT.DESCRIPTION] section of this file has version strings for drivers and applications.	Required
WIN.INI	Initialization file for Windows applications.	Required
WINFILE.INI	Initialization file for WINFILE.EXE.	Optional

Table 3-3
Windows System Kernel

Component	Description	
SYSTEM/DOSX.EXE	DPMI server.	Required
SYSTEM/WSWAP.EXE	Swaps out standard-mode Windows when a DOS application is run. *Optional, if no DOS applications are run during a Windows session.*	Optional
SYSTEM/KRNL386.EXE	Windows 3.1 kernel for 32-bit systems.	Required
SYSTEM/GDI.EXE	Windows 3.1 Graphical Device Interface code.	Required
SYSTEM/USER.EXE	Windows 3.1 user interface.	Required

Table 3-4
Windows Enhanced Mode Files

Component	Description *(All files in this group are needed only for Enhanced mode.)*	
SYSTEM/WIN386.EXE	Virtual machine manager and default virtual device drivers.	Required
SYSTEM/VTDAPI.386	Virtual timer device API. Needed for Enhanced mode multimedia.	Required
SYSTEM/NORVKD.386	Windows 3.1 system file. Runs Windows in enhanced mode.	Required
SYSTEM/VPOWERD.386	Windows 3.1 Enhanced Mode Power Management support.	Required

Table 3-5
Windows System Device Drivers

Component	Description	
SYSTEM/COMM.DRV	Windows 3.1 serial port driver.	Required
SYSTEM/KEYBOARD.DRV	Windows 3.1 keyboard driver.	Required
SYSTEM/SYSTEM.DRV	Windows 3.1 system driver.	Required
SYSTEM/YESMOUSE.DRV	Windows 3.1 driver reports a mouse is available.	Required
SYSTEM/6100DISP.DRV	PEN*KEY 6100 portrait video driver.	Required
SYSTEM/SOUND.DRV	Windows 3.1 sound driver	Optional
SYSTEM/UNIDRV.DRV	Windows 3.1 universal printer driver. Used by NOR4800.DRV. *Required only if NOR4800.DRV for NPCP printing is used.*	Required

Table 3-6
Windows Installable Device Drivers

Component	Description	
SYSTEM/NORWINPM.DRV	Windows Advanced Power Management (APM) driver for 6100 Computer. Provides interface to power management and system control hardware. Requires APM BIOS (ELANAPM.EXE).	Required
SYSTEM/UCLKPEN.DRV	Pen driver for 6100 Computer. Provides pen input for *Microsoft Windows for Pen Computing 3.1*, mouse input for Windows 3.1.	Required
SYSTEM/61SCAN.DRV	Scanner driver for 6100 Computer. Provides bar-code scanning capabilities for Windows.	Optional
SYSTEM/NORNPCP.DRV	Provides NPCP printing support under Windows 3.1.	Optional
SYSTEM/NORIRDA.DRV	Provides IrDA printing capabilities.	Optional
SYSTEM/NOR4800.DRV	Windows 3.1 4800 series printer driver, supports NPCP printing.	Optional
SYSTEM/NOR6805.DRV	Windows 3.1 6800 series printer driver, supports IrDA printing.	Optional

Table 3-7
EGA Device Fonts

Component	Description	
SYSTEM/EGAFIX.FON	Fixed-pitch device font.	Required
SYSTEM/EGAOEM.FON	OEM device font.	Required
SYSTEM/EGASYS.FON	Proportional device font.	Required

Table 3-8
Popular System DLLs

Component	Description	
SYSTEM/COMMDLG.DLL	Provides common dialogs like file open, file save, and printing.	Optional
SYSTEM/LZEXPAND.DLL	Functions for file expansion and copying.	Optional
SYSTEM/SHELL.DLL	Basic services common to Windows 3.1 shell programs, such as File Manager.	Optional
SYSTEM/VER.DLL	Windows versioning API.	Optional
SYSTEM/WIN87EM.DLL	Floating-point emulator.	Optional
SYSTEM/OLECLI.DLL	Object Linking and Embedding client.	Optional
SYSTEM/OLESVR.DLL	Object Linking and Embedding server.	Optional
SYSTEM/DDEML.DLL	Dynamic Data Exchange.	Optional
SYSTEM/TOOLHELP.DLL	Debugging services.	Optional

Table 3-9
Utilities

Component	Description	
PENALIGN.EXE	PEN*KEY 6000 Series pen calibration application.	Optional

NORAND Shell for Windows: NORSHELL.EXE

The Windows shell replacement program, NORSHELL.EXE, provides system management functions that are specific to PEN*KEY 6000 Series Computers. NORSHELL.EXE, which must be loaded in all Windows configurations, provides a method for launching multiple applications when Windows starts; this is functionality that is similar to that provided by the standard Windows shells (Program Manager and File Manager).

Installation

To install NORSHELL.EXE as the Windows shell program, edit the “shell=” line in SYSTEM.INI, as shown in the following example:

```
shell=norshell.exe
```

Configuration: WIN.INI Entries

To configure NORSHELL.EXE, make the following entries in the “[windows]” section of the WIN.INI Windows initialization file:

- `NorShellRun`
  Add the “`NorShellRun=<application command line>`” line to WIN.INI in order to launch an application. You may also specify any command line switches that required by the application.
- `NorShellRunDir`
  If you need to specify a startup drive and directory for an application, add the “`NorShellRunDir=<drive:directory>` ” line to WIN.INI. NORSHELL changes the default drive and directory as needed.
- `NorShellLaunch`
  Note that NORSHELL does not interpret the “`run=`” and “`load=`” lines in WIN.INI so that conflicts with other Windows shell programs can be avoided. For example, use the Windows File Manager as your application and have it process the WIN.INI “`run=`” and “`load=`” lines.

If you do need to launch several utilities or applications before starting a particular application, add a "NorShellLaunch=<program list>" line to WIN.INI. This line is formatted identically with the "run=" line that is used by the standard Windows shell programs. The utilities or application programs are run in the same order as they appear in the WIN.INI file; and are displayed with the SW_SHOWMINIMIZED setting. Refer to the Windows 3.1 documentation for the ShowWindow function.

NORSHELL WIN.INI Examples

All of the examples in this paragraph assume the SYSTEM.INI entry documented in the preceding paragraphs. The following example WIN.INI file demonstrates running an application, MYAPP.EXE, after changing to drive "C" and directory "\app." Before running the application, the Windows clock and calculator programs are executed.

```
[WINDOWS]
NorShellRun=MYAPP.EXE /a switch /another switch
NorShellRunDir=c:\app
NorShellLaunch=clock.exe calc.exe
```

The following example WIN.INI file launches the Windows File Manager shell application, which launches the clock and calculator in turn.

```
[WINDOWS]
NorShellRun=WINFILE.EXE
run=clock.exe calc.exe
```

Shutting Down Windows

Windows shuts down automatically whenever the shell program exits. However, because NORSHELL does not exit (in that sense), it shuts Windows down whenever it detects that the application from the NorShellRun line no longer exists.

An application may shut down Windows on its own, by calling the Windows 3.1 ExitWindows function. This is the preferred method.

Windows Power Management Driver: NORWINPM.DRV, VPOWERD.386

The NORWINPM.DRV driver provides a Windows interface to the APM BIOS extensions, for both the Standard and Enhanced modes. VPOWERD.386 is also included for the Enhanced mode. The driver's functions include the following:

- Idling the CPU when Windows has nothing to do.
- Warning the user about critical power management events, such as low batteries, unless an application wants to take over that responsibility.
- Broadcasting APM event codes to Windows applications and drivers and DOS TSRs.

The information in this paragraph corresponds to version 1.05 of NORWINPM.DRV.

Installation

NORWINPM.DRV is a Windows 3.1 installable driver. It requires that the APM 1.1 BIOS extensions be installed before Windows is started. The APM BIOS (that NORWINPM needs) may come from the built-in system BIOS or from a separate DOS TSR, ELANAPM.EXE, which must be run before *starting* Windows.

To install NORWINPM.DRV:

Place a keyword on the "drivers=" line and add a line to the "[drivers]" section of SYSTEM.INI, equating the keyword with the actual path to NORWINPM.DRV. This is the typical approach. The power management entry should come first on the "*drivers*=" line.

The following is an example of a section of the SYSTEM.INI file, showing the use of the SYSTEM.INI commands.

```
[boot]
drivers= power pen penwindows scanner npcp irda
```

where:

drivers= tells Windows to load the installable drivers. This list of drivers is the standard driver list (in the order that it needs to be). The significant drivers here, are **power** and **pen**. These are merely token names for the drivers. The actual driver names are specified in the following [drivers] section of the SYSTEM.INI file:

```
[drivers]
power=norwinpm.drv
pen=uclkpen.drv
[386 enh]
device=vpowerd.386
```

where:

power= tells Windows the actual name of the installable Power Management driver named in the [boot] section of the SYSTEM.INI file (listed above).

pen= tells Windows the actual name of the installable Pen driver named in the [boot] section of the SYSTEM.INI file (listed above).

▶ **NOTE:** *Do not load the DOS Power Management driver, NORDOSPM.EXE, if the 6100 Computer is running Windows.*

Configuration

NORWINPM.DRV supports the SYSTEM.INI entries described below. Place them in the [POWER Driver] section of SYSTEM.INI. Note: case is insignificant.

Fuel Gauge Settings

WindowX, WindowY

Valid value: 0

It defaults to the lower right corner of screen (last position).

These entries set the initial screen position of the fuel gauge. They can be set to offscreen coordinates, as well, so the gauge can be hidden, if desired. These entries are updated whenever the user moves the fuel gauge.

OnTop

Valid values: 0, 1

Default value: 1

This determines whether the fuel gauge is a “topmost” window or not. A “topmost” window appears above all other non-topmost windows, typically floating above the active window. Setting this entry to 0 allows other windows to cover the fuel gauge whenever they are selected.

TextOptions

Valid values: 0x0000 to 0x000F

Default value: 0x0006

This is an Options Bitmap that indicates the items to be displayed in text form beneath the fuel gauge icon. The bits are:

- 0x0001: Show battery life remaining percentage. “60%”.
- 0x0002: Show battery status: “High/Low/Critical”.
- 0x0004: Show charge status: “Charging”.
- 0x0008: Show AC status: “AC”.

Miser Settings

Miser

Valid values: 0, 1

Default value: 0

This entry enables the Power Miser by setting this entry to 1. The power miser can increase battery life when running an application that does not allow the Windows kernel to idle. It does this by forcing the Windows kernel to idle at appropriate times. If the “busy” bar on the fuel gauge icon is always topped out, try enabling this entry. The Miser can be somewhat invasive so the best approach is to implement a well power-managed application that allows the system to idle on its own. See the description below, for MiserLimit.

MiserLimit

Valid values: 0 to 255

Default value: 2

As mentioned above, the Miser mode can be invasive to the system. It might force too many idles to occur. This entry limits the amount of idling that Miser mode forces. Higher values cause fewer idles. If it appears that the Miser is slowing down or interfering with an application, increase this value to lessen the interference (which also lowers power savings.) If the interference cannot be cleared, disable the miser mode.

Message Output Settings

ApmSuspendDialog

Valid values: 0, 1

Default value: 1

When the user presses the on/off button, NORWINPM.DRV polls all of the drivers and programs in the system with a USERSUSPEND message (see below) to determine if it is OK to suspend. This configuration entry determines whether NORWINPM.DRV displays a dialog notifying the user if an application or driver refuses to suspend.

Value	Description
0	No warnings or debug messages, if the user suspend request fails. It is assumed that the system component that fails the request, either notifies the user on its own, or suspends the system after finishing the critical activity in which it was involved. It is very impolite to ignore the button press and not give the user any feedback.
1	NORWINPM.DRV notifies the user that the suspend request failed by displaying a message (see messages below) in a dialog box. Execution continues after the user presses an OK button.

MsgLevel

Valid values: 0, 1, 2, 8, 10, 15 (see table below)

Default value: 1

This configuration entry indicates which APM events cause a dialog to be presented to the user by NORWINPM.DRV. Note that each dialog level includes all the messages from the levels below it.

Value	Description
0	No Warnings or Debug Messages: appropriate when an application wants to control the timing and presentation of power management event notification. The application would catch the "WM_POWER" messages broadcast by NORWINPM.DRV and deal with them accordingly.
1	Warnings: only display dialogs for APM warnings, such as battery status. See *MsgRepeatMinutes* below. This is the driver default. Note that if the user does not clear the dialog by pressing the OK button or by pressing a key, the dialogs continue to stack up.
2	Infrequent Dialog Information: display dialogs for warnings and informational messages, such as suspend/resume, etc. This is a debug-mode switch. The only messages that it does not show are messages that repeat frequently. In addition, this setting causes the system to beep periodically when system activity is detected such as keystrokes, pen activity, or communications messages. This can determine if suspend or backlight time-outs are held off correctly.
5	A debug setting that displays messages that repeat frequently (every 10 seconds or so) but are different each time.
8	Frequent, But Non-Repeating Debug Information:
10	Frequent, Repeating Debug Information: a debug setting that additionally displays messages that repeat frequently and do not vary. This setting usually causes a steady stream of dialogs to appear at short intervals (10 seconds or less).
15	All Messages:

NORWINPM.DRV SYSTEM.INI Configuration Example

To display only warnings (the typical mode), turn on the power miser, and allow other windows to cover the fuel gauge, place the lines shown below into SYSTEM.INI:

```
[Power Driver]
MsgLevel=1
Miser=1
OnTop=1
```

User Notifications

Installation Messages

The following messages are displayed in a dialog box when an error occurs while NORWINPM.DRV is initializing.

```
Could not find APM BIOS, Windows power management not installed!
APM BIOS connect failed!
Power management disabled!
Could not allocate a timer for polling APM BIOS!
Power management disabled!
```

The message below appears in a dialog when the user presses the “On/Off” (Suspend/Resume) button and the system cannot suspend because a Windows application, driver, or DOS TSR refused the suspend request.

```
An application or driver is refusing to suspend
```

Informational Messages

The messages below appear in a dialog box whenever the *ApmEventDialogs* entry in SYSTEM.INI (see above) is set to the value of 2, or higher. They are intended for power management debugging. See the *Standard APM Event Codes* paragraph, on page 3-13, for descriptions of APM event codes.

```
APM Event <Hex APM Event ID>.
APM OEM Event <Hex OEM Event ID>.
```

Fuel Gauge Display

The NORWINPM.DRV fuel gauge icon, depicted below, shows all of the possible elements:

- The battery (on right side of gauge) displays the charge level. When it is completely filled in, the battery is fully charged.
- The top terminal of the battery contains a “+” symbol when charging.
- The smaller column to the left of the battery is the “Busy Meter”. It contains a bar that moves up when the system is busy. If the bar “pegs”, a blinking dash “ ” symbol appears at the top of the bar to show that the system is not idling.

The optional text below the battery and busy meter can be enabled or disabled with the *TextOptions* SYSTEM.INI entry.

Keep in mind that the position of the fuel gauge on the screen is controlled by parameters in the SYSTEM.INI file. And, is therefore under application control. It can be located anywhere on the screen or can be moved off of the screen, using the parameters in the SYSTEM.INI file. In addition, the user can move it around at will, simply by dragging it around with the finger or stylus.

Also, the "`Always on Top`" feature can be enabled or disabled, using the "`OnTop=`" setting in the "[`Power Driver`]" section of the SYSTEM.INI file.

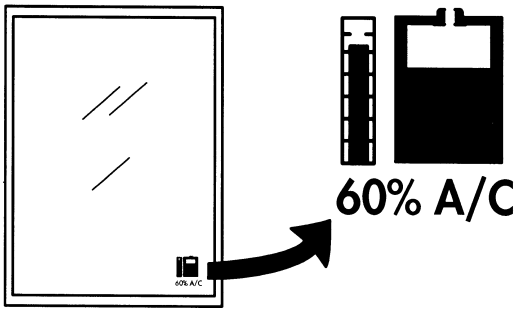


Figure 3-1
Fuel Gauge Display

Figure 3-1 depicts a 6100 Computer screen with a typical location for the fuel gauge, showing an expanded view of the gauge details to the right of the screen.

NORAND Power Management Programming Interface for Windows

Windows Power Management

The APM operating system driver, NORWINPM.DRV, cooperates with the APM BIOS, to save as much power as possible without application or driver intervention. This allows a system to be built with reasonable power consumption, without needing to create power-aware drivers and applications. Even better power management can often be achieved by cooperating with applications and drivers.

CPU Power Management

NORWINPM.DRV monitors the Windows task scheduler to determine when the CPU is idle. When idleness is detected, the CPU and other components are placed into a low power state until the next hardware interrupt occurs. The system then returns to full speed to process the interrupt and any Windows events created by the interrupt. When an application is idle, 95% of CPU current can usually be saved.

Power Management by Windows Applications

Windows applications can benefit from system power management, because it frees up the Windows kernel and allows NORWINPM.DRV to idle the CPU.

Avoiding busy loops is a requirement to maintain system response in a cooperative multitasking system like Windows;. but busy loops also waste system power. Applications should be careful to never perform busy-wait when implementing time-outs or polling routines. Instead, use an event-based approach. Perhaps by acquiring a system timer, so the system can idle during periods of inactivity.

PeekMessage() loops are often used instead of the usual GetMessage() loop whenever an application has background processing to do. An application that uses PeekMessage() must be careful to call GetMessage() or WaitMessage() whenever there are no messages waiting and there is no background processing to do. Applications that call PeekMessage() continuously does allow other Windows ap-

plications to run if they have messages, which keeps your application Windows-friendly. But if there are no pending message for anyone, PeekMessage() immediately returns to its caller and does not allow the Windows kernel to become idle. The result is an application that calls PeekMessage() more or less continuously with short breaks while other applications process their messages. In other words, a "cooperative busy loop" is the result. The power-friendly way to implement a PeekMessage() loop is to implement the following:

```
Call PeekMessage()
   if no messages, perform background processing
   if there is no background processing to do, then
      call GetMessage()

   or

      call WaitMessage()
```

▶ **NOTE:** *Some Windows application libraries are not written with mobile computing in mind and may make use of PeekMessage() busy-loops in their message handlers.*

Holding Off Suspend Time-outs

If an application is busy with communication or processing, for extended periods of time with no user interaction, the system might time out into suspend during processing. In order to prevent this, the application should fail *SUSPENDREQUEST* broadcasts while performing CPU-bound tasks. As mentioned above, *USERSUSPEND* requests can also be failed, but the application should inform the user about this type of action, otherwise the user may think that the system is locked up (since the Suspend/Resume button does not work) and might reset the system or take some other drastic measure.

APM Event Broadcasts

The APM BIOS specification defines event codes for various power management events. NORWINPM.DRV collects these events from the APM BIOS and either handles them directly or broadcasts them to the system components (Windows drivers, applications and DOS TSRs). System components can use the events to monitor system power state changes. Some APM events have responses which can implement cooperative power management.

Receiving APM Event Broadcasts

NORWINPM.DRV broadcasts APM events to Windows applications, system drivers, installable drivers, VxDs, DOS drivers, and DOS TSRs loaded installed in the System VM. Each device has a different broadcast API. The most common APIs are documented below.

Windows Applications

Windows applications receive APM event broadcasts via the message procedures of any top-level windows that they own. The message format is:

Msg:	WM_POWER (defined in WINDOWS.H)
wParam:	PWR_* event code defined in APMCODES.H or WINDOWS.H.
lParam:	Undefined
Returns:	Application should return PWR_FAIL (from WINDOWS.H) if it wishes to reject an event. Otherwise, PWR_OK should be returned.

Windows Installable Drivers

Windows installable drivers receive APM event broadcasts via their *DriverProc* entry points. The message format is:

Msg:	DRV_POWER (defined in WINDOWS.H)
lParam1:	PWR_* event code defined in APMCODES.H or WINDOWS.H.
lParam2:	Undefined
Returns:	The driver should return PWR_FAIL or PWR_OK

DOS Real-Mode Drivers and TSRs

DOS real-mode system components receive APM event broadcasts through software interrupt 2Fh. A driver or TSR that hooks the broadcast should nest to previous owners of the interrupt before handling it; even if it wants to fail the call. This approach allows other drivers an opportunity to see the event; even if it results in failing. The message format is:

AX:	530Bh, the APM event broadcast ID
BX:	APM_* event code defined in APMCODES.H.
CX:DX:	Optional 32-bit Norand APM event parameter
Returns BX:	Set BX to 0 for success or 80h to fail the event.

Windows System Drivers

Windows system drivers receive APM event broadcasts through a defined entry point. NORWINPM.DRV does not currently broadcast to system drivers, since none of them has this entry point.

Standard APM Event Codes

These are the most common APM event codes defined and is described in the Intel/Microsoft APM Specification, versions 1.0 and 1.1. There are additional events, but these are the ones typically handled by drivers and applications. Events marked "(1.1)" are only valid in APM 1.1, and may not be present on open systems that only implement 1.0 events. Additional clarification of NORWINPM.DRV's use of these events is given below. There are other APM codes that may also be broadcast.

SUSPENDREQUEST, System Suspend Request Notification

A *suspend time-out* is pending. Drivers and applications should fail the event if they are busy. If the event is failed, NORWINPM.DRV restarts the suspend time-out suspend request but does not suspend the system.

SUSPENDRESUME, Normal Resume System Notification

The system has returned from a suspend time-out or a user suspend. Drivers and applications should respond to system changes at this time.

PWRCHANGE, Power Status Change Notification

Power status has changed. The optional parameter of this event message contains the *first* four bytes of the current `tagPOWER_STATUS`(refer to NORAPM.H, in the Tool Kit). You can use the message to determine whether the system is running on battery power or line power without having to poll NORWINPM.DRV.

BATLOW, Battery Low Notification

The system battery is running low. The optional parameter of this event message contains the *last* four bytes of the current `tagPOWER_STATUS`. Use the message to determine the current battery status without polling NORWINPM.DRV.

USERSUSPEND, User System Suspend Request Notification

The user has pressed the suspend/resume switch and wants to suspend the system. Drivers or applications that attempt to fail this request probably become an annoyance to the user unless the drivers put up some sort of polite explanation as to why the system does not turn off when the switch is pressed. In any case, the driver or application must prepare for an impending suspend because the APM driver is allowed to override the fail request and suspend the system.

TIMEUPDATE, Update Time Notification

This event is issued whenever the time has been updated to correct for "time-warp" that is not due to a suspend. Applications that perform processing tied to real time should check the time whenever they see this event, in addition to checking the time whenever they see any of the resume events.

CRITICALSUSPEND, Critical System Suspend Notification

This event is never broadcast. NORWINPM.DRV immediately suspends the system when it receives it and broadcasts a `CRITICALRESUME` event upon resuming.

CRITICALRESUME, Critical Resume System Notification

The system was suspended without sending a notification broadcast in advance. This event is usually caused by a power failure. Drivers should typically disable and reenable themselves when receiving this message.

STANDBYREQUEST, System Standby Request Notification

Because of the efficacy of the suspend mode, NORAND systems do not currently implement a *standby* mode. In addition to `STANDBYREQUEST`, the `USERSTANDBY` and `USERSTANDBYRESUME` messages are also ignored or are never generated.

APM Event Code Broadcast Values

Refer to the *APM Event Code Broadcast Values* paragraph, in *Appendix A, Sample Configuration Files*, for a listing of the event codes and their values.

Generally, Intermec Technologies Corporation attempts to use APM 1.1 standardized event codes whenever appropriate. However, Intermec Technologies Corporation may choose to make additional events available on systems with extended functionality. The OEM codes listed in APMCODES.H represent currently defined codes, but does not imply that all or any OEM events are available on any particular Intermec system.

Windows Pen Driver: UCLKPEN.DRV

The Windows pen driver (UCLKPEN.DRV) interfaces the pen digitizer to Windows 3.1. The digitizer may be based on resistive touch-panel technology; or it may use active electronics.

Pen Applications

When *Microsoft Windows for Pen Computing* is executing, the driver acts as a pen device. At other times, the driver acts as a mouse device. Usually, the pen simply appears as a mouse device. Any Windows application that can use a mouse can use the pen. Note that "pen-down" is equivalent to pressing the left, or "inner," button on a desktop mouse.

If *Microsoft Windows for Pen Computing* is loaded, the pen can appear to be both a mouse and an ink device. The *Microsoft Windows for Pen Computing* developer's kit documents the inking API that captures and manipulates ink input.

Now, which mode should your application use: mouse or inking device? In general, applications that do not require handwriting recognition should simply treat the pen as a mouse. Mouse input is quite adequate for pointing and drawing tasks such as signature capture. *Microsoft Windows for Pen Computing* requires additional storage and processing resources; therefore, it should be used when handwriting recognition is a requirement.

Installation

Place the files in the following directories:

File	Directory
PENALIGN.EXE	WINDOWS\
NORWINPM.DRV	WINDOWS\SYSTEM
UCLKPEN.DRV	WINDOWS\SYSTEM
PENWIN.DLL	WINDOWS\SYSTEM

Configuration

You can configure the Windows pen driver by editing the appropriate entries in SYSTEM.INI. Unless otherwise noted, all entries are placed in the "`[Pen Driver]`" section of SYSTEM.INI.

To install UCLKPEN.DRV, place a keyword on the "`drivers=`" line, in the `[boot]` section of SYSTEM.INI, and add a line to the "`[drivers]`" section of SYSTEM.INI equating the keyword with the actual path to UCLKPEN.DRV. If you are using *Microsoft Windows for Pen Computing*, place the pen driver entry in the [boot] section of SYSTEM.INI (drivers=...) before the (penwindows) entry, in the [drivers] section of the SYSTEM.INI. Although the following example shows *Microsoft Windows for Pen Computing* being installed, it is not required for digitizer use.

Required SYSTEM.INI Entries

SYSTEM.INI:

```
[boot]
drivers= power irda pen penwindows scanner npcp

[drivers]
power=norwinpm.drv
pen=uclkpen.drv
penwindows=penwin.dll
```

drivers= tells Windows to load the installable drivers. This list of drivers is the standard driver list (in the required order). The significant driver here, is **penwindows**. This is just a token name for the driver. The actual driver name is specified in the [drivers] section of the SYSTEM.INI file:

pen=UCLKPEN.DRV tells Windows the actual name of the installable driver named in the [boot] section of the SYSTEM.INI file.

penwindows= tells Windows the name of the installable PENWIN driver.

Refer to the example in the *SYSTEM.INI Configuration Example* paragraph or see the `[Pen Driver]` section in the example SYSTEM.INI file, in *Appendix A, Sample Configuration Files*, for further details on how to configure the Windows Pen driver.

Hardware Interface

These entries, in the SYSTEM.INI file, initialize the hardware interface for the PEN*KEY 6000 Series Computer.

Unless otherwise specified, any of these entries used should go into the [pen driver] section of the SYSTEM.INI file.

Most of these hardware interfaces default to the values needed for the PEN*KEY 6000 Series Computer and should not need to be set. Note the “required values”.

PortAddr

Valid values: 0 to 1023

Default value: 520

Required value for 6100 Computer: 512

This entry sets the decimal digitizer base I/O port address.

IrqLevel

Valid values: 0 to 15

Default value: 12

Required value for 6100 Computer: 14

This entry sets the decimal IRQ level used by digitizer.

BaseClock

Valid values: 0 = 33 MHz
1 = 2 MHz
2 = 14 MHz

Default value: 0

▶ **NOTE:** *This entry sets the base clock value for Gazelle/Logitech digitizer. The only value that is supported by the PEN*KEY 6000 Series Computer is the default value of 0.*

PointsPerSecond

Valid values: 1 to approximately 200

Default value: 50

This entry sets the initial digitizing rate, in packets per second. The default value is sufficient when using the pen only as a pointing device. Values between 100 and 150 give better results, when using handwriting recognition, such as when *Microsoft Windows for Pen Computing* is installed. This value must be in the range of 120–150 for handwriting recognition.

MaxIPS

Valid values: 0 to approximately 150

Default value: 75

This entry sets the fastest movement allowed, in inches per second. The default value should be adequate for PEN*KEY 6000 Series Computers.

Digitizer Calibration

These entries, in the SYSTEM.INI file, set the calibration of the digitizer for the PEN*KEY 6000 Series Computer.

Unless otherwise specified, any of these entries used should go into the `[pen driver]` section of the SYSTEM.INI file. Most of these hardware interfaces default to the values needed for the PEN*KEY 6000 Series Computer, and should not need to be set. Note the “required values”.

cxRawWidth

Valid values: 0 to 32767

Default value: 4500

This entry sets the physical width of the digitizer, in thousandths of an inch, in the display’s native (non-rotated) hardware orientation. The default value should be adequate for PEN*KEY 6000 Series Computers.

cyRawHeight

Valid values: 0 to 32767

Default value: 3000

This entry sets the actual height of the digitizer, in thousandths of an inch, in the display’s native (non-rotated) hardware orientation. It should not need to be specified for the PEN*KEY 6000 Series Computers.

wDistinctWidth

Valid values: 0 to 32676

Default value: 3500

This entry sets the width of the display, in its non-rotated orientation, in digitizing units. It is set by the PENALIGN.EXE calibration applet.

wDistinctHeight

Valid values: 0 to 32767

Default value: 3045

This entry sets the height of the display, in its non-rotated orientation, in digitizing units. It is set by the PENALIGN.EXE calibration applet.

wOffsetX

Valid values: 0 to 32767

Default value: 1000

This entry sets the digitizer value that corresponds to the leftmost point on the rotated display, plus 1000. Negative offsets, to 1000, are specified by using values less than 1000.

wOffsetY

Valid values: 0 to 2767

Default value: 1000

This entry sets the digitizer value that corresponds to the uppermost point on the rotated display, plus 1000. Negative offsets to –1000 are specified by using values less than 1000.

UseNV

Valid values: 0, 1

Default value: 1

When this entry is nonzero, Norand nonvolatile memory reads and stores the digitizer calibration information.

When this entry is set to 0, it overrides the calibration stored in nonvolatile storage, with the calibration values from SYSTEM.INI.

Display Orientation

These entries, in the SYSTEM.INI file, match the orientation of the PEN*KEY 6000 Series digitizer to the screen display. The Pen driver must take into account any rotation of the screen display, as well as additional rotational offset of the digitizer relative to the display. The calibration values given above are swapped and flipped to match the final user orientation of the digitizer.

Unless otherwise specified, any of these entries used should go into the [pen driver] section of the SYSTEM.INI file.

DisplayOrientation

Valid values for degrees of rotation:
0 = 0
1 = 90 *(default)*
2 = 180
3 = 270

This entry specifies the value for the orientation of the display. 0 is the default hardware orientation. Values from 1 to 3 indicate successive 90-degree clockwise rotations. Locate this entry in the [`display driver`] section of SYSTEM.INI.

PenOrientation

Valid values for degrees of rotation:
0 = 0
1 = 90
2 = 180 *(default)*
3 = 270

This entry specifies additional rotation of the pen digitizer orientation relative to the display, which may already be rotated by the preceding *DisplayOrientation* value. The pen orientation, relative to the default hardware display orientation, is the *DisplayOrientation* value (preceding) added to the *PenOrientation* value.

FlipX

Valid values: 0, 1

Default value: 0

Required value for 6100 Computer: 1

When nonzero, the X-axis values are inverted.

SYSTEM.INI Configuration Example for UCLKPEN.DRV

The following is an example section of SYSTEM.INI, to configure the 6100 Computer with the default values for the Windows Pen Driver:

```
[Pen Driver]
; Increase wOffsetX to move cursor down relative to pen.
wOffsetX=696
; Increase wOffsetY to move cursor to left relative to pen.
wOffsetY=709
; Increase wDistinctWidth to make cursor move slower relative to pen in the
; up/down direction on screen.
wDistinctWidth=3504
; Increase wDistinctHeight to make cursor move slower relative to pen in the
; left\right direction on screen.
wDistinctHeight=3680
; Sets initial digitizing rate (packets / second)
; Range: 1 200.
; Set to 120 150 if you install handwriting recognition.
PointsPerSecond=135
; Set to required value.
PortAddr=512
; Set to required value.
IrqLevel=14
PenOrientation=3
FlipX=1
DoEOI=SMART
UseNV=1
cxRawWidth=4500
cyRawWidth=3000

[Display Driver]
DisplayOrientation=1
```

Windows Pen Calibration: PENALIGN.EXE

PENALIGN.EXE is the Windows pen-calibration utility (with no parameters). There are two screens, the first prompts to tap three times in marked screen locations; the second prompts to overlay (match) the cursor and the pen position.

Alternatively, you can set the calibration by adjusting the calibration entries in SYSTEM.INI. This is a fairly tedious approach that requires an understanding of digitizer orientation and digitizer coordinate systems. To enable use of the calibration entries in SYSTEM.INI, set the *UseNV* setting to 0.

Pen for Windows: PENWIN.DLL

The Windows pen program, properly known as *Microsoft Windows for Pen Computing*, can be installed and used on the 6100 Computer. The Pen for Windows files are located on the fourth diskette in the Tool Kit. Place the PENWIN.DLL file into the WINDOWS\SYSTEM directory.

Required SYSTEM.INI Entries

```
[boot]
drivers= power irda pen penwindows scanner npcp
```

drivers= tells Windows to load the installable drivers. This list of drivers is the standard driver list. The significant driver is "penwindows," a token name for the driver. The actual driver name is in the SYSTEM.INI [drivers] section:

```
[drivers]
penwindows=PENWIN.DLL
```

penwindows=PENWIN.DLL tells Windows the actual name of the installable driver named in the `[boot]` section of the SYSTEM.INI file (listed above).

▶ **NOTE:** *A recognizer must be purchased and installed for this application to be useful. Intermec Technologies Corporation offers both CIC and Synaptics. Refer to the "Getting Started" section of this publication for information on handwriting recognition.*

Required PENWIN.INI Entries

This is a sample PENWIN.INI file to use with Windows 3.1 on a 6100 Computer:

```
[Current]
User=User 1
InkWidth=1
InkColor=0
SelectTimeout=500

[*User 1]
TryDictionary=100
ErrorLevel=25
EndRecognition=8000
TimeOut=500
WriteDirection=103
MenuDropAlignment=0
Preferences=0
IntlPreferences=0
Recognizer=mars.dll

[Dictionary List]
MAINDICT.DLL=

[Recognizer List]
MARS.DLL=

[MsSpell]
MSSPELL.DLL=

[MsMainDict]
enuMain=

[User List]
User 1=

[sysges]
C!=xx,0,{Ctrl}{Ins}
P!=xx,0,{Shift}{Ins}
X!=xx,0,{Shift}{Del}
U!=xx,0,{Alt}{Bs}

[Pen Palette]
SKBPos=15 92

[MS-DOS]
EnableMSDOS=1
```

Integrated Scanner: 61SCAN.DRV

This is a Windows driver. NORAND integrated scanner support consists of two basic components:

- 61SCAN.DRV -- Windows device driver.
- SYSTEM.INI -- file containing scanner-specific configuration information.

Installation

Place the scanning files in the following directories:

File	Directory
SYSTEM.INI	\windows
NORWINPM.DRV	\windows\system
61SCAN.DRV	\windows\system

NOTE: *NORWINPM.DRV needs to be loaded before 61SCAN.DRV.*

The Windows scan driver, 61SCAN.DRV, currently supports the Pod scanner.

Configuration

SYSTEM.INI Entries

The following entry is required In the SYSTEM.INI file. It allows the opening and closing of the scanner by an application.

```
[boot]
drivers=power irda pen penwindows scanner npcp
```

where:

drivers= tells Windows to load the installable drivers. This list of drivers is the standard driver list (in the order that it needs to be). The significant driver here, is scanner. This is just a token name for the driver.

The actual driver name is specified in the following [`drivers`] section of the SYSTEM.INI file:

```
[drivers]
scanner=61SCAN.DRV
```

scanner=61SCAN.DRV tells Windows the actual name (61SCAN.DRV) of the installable driver named in the preceding [`boot`] section of the SYSTEM.INI file.

Entries in [scanner driver] Section of SYSTEM.INI

```
[Scanner Driver]
ScannerHardwareType=PEN*KEY
MessageBeepScanVerification=INTERNAL
MessageBeepStatusNotification=INTERNAL
MessageBoxStatusNotification=TRUE
EnableScannerWhenDriverLoads=FALSE
DisplayScanningDataDialog=TRUE
ShowWindowOnLoad=TRUE
EnableScanCodes=TRUE
;ScanKey = nnn
ExternalFlashOnScan=FALSE
AimingBeamDuration=0
```

ScannerHardwareType: Controls the type of hardware the scanner is using. This value must be set or the driver does not load. There is no default value. Valid values are PEN*KEY/33, PEN*KEY, and TETHERED.

MessageBeepScanVerification: Controls the type of beep generated when a good scan is obtained. Valid values are OFF, INTERNAL, EXTERNAL, and ALL. ALL has the same effect as both internal and external.

MessageBeepStatusNotification: Controls the type of beep generated when a status change occurs. Valid values are OFF, INTERNAL, EXTERNAL, and ALL. ALL has the same effect as both internal and external.

MessageBoxStatusNotification: Controls whether or not a message box is generated when a status change occurs. Valid values are TRUE and FALSE.

EnableScannerWhenDriverLoads: Controls conditions under which the scanner is enabled. Valid values are TRUE and FALSE. If this parameter is set to TRUE, the following conditions are in effect:

- The scanner is enabled when it is loaded by Windows, and is always active until Windows shuts down.
- Scanned data is directed to the keyboard buffer, so applications can pick up scanned data as if it had been input through the keyboard.
- OpenDriver and CloseDriver calls are not required to be issued by the application to use the scanner.
- Multiplexing of the scanner and external COM2 connections is *not* allowed.
- This option uses more power.

DisplayScanningDataDialog: Controls whether or not the Scanning Data... dialog is displayed when the trigger is pulled. Valid values are TRUE and FALSE.

ShowWindowOnLoad: Controls whether or not the scanner window icon is displayed on the screen. If this is FALSE, it is not possible to get to the scanner window. Valid values are TRUE and FALSE.

EnableScanCodes: Controls whether or not the scanner includes Scan Codes in the key messages that it generates. Valid values are TRUE and FALSE.

ScanKey: "nn" is the hexadecimal scan code value for the key to fire the scanner (i.e., you can program the "Enter" key to fire the scanner).

ExternalFlashOnScan: Controls whether the Good Scan light is flashed manually by the scanner driver when the data is received. Valid values are TRUE and FALSE.

AimingBeamDuration: Controls the length of time (in milliseconds) that a long-range scanner using a dedicated UART emits an aiming beam. All other scanners should have this option set to 0 (the default value).

Usage

Because the integrated scanner has its own dedicated UART, the communications with the scanner are completely separated from the external communications ports on the 6100 Computer.

When an *OpenDriver* is issued to the scanner, the dedicated UART hardware is initialized.

When the *CloseDriver* call is issued, the scanner driver closes the dedicated UART communications hardware. Any application multiplexing with the integrated scanner is required to "open" the UART hardware, to initialize it at a later time when communications to the scanner is needed.

An example of an *OpenDriver* call is as follows:

```
hDRVR = OpenDriver("61SCAN.DRV", (LPCSTR)NULL, (LPARAM)NULL);
```

The scanner is powered on, and scanner data is delivered to the application with the input focus as *WM_CHAR* messages. These messages are distinguishable from "standard" *WM_CHAR* messages in that the virtual key code in *wParam* is the ASCII value of a scanned character and *lParam* is sent as *0x00000000* (in particular, the OEM nibble (bits 16–23) that contains the OEM scan code of 0).

An example of an *CloseDriver* call is as follows:

```
CloseDriver(hDRVR, (LPARAM)NULL, (LPARAM)NULL);
```

This turns the scanner off.

▶ **NOTE:** *See the Borland C++ help for OpenDriver / CloseDriver. See also Windows 3.1 SDK for additional information.*

NPCP Printing for Windows: NOR4800.DRV, UNIDRV.DLL

NPCP printing support under Windows consists of DOS device driver NORNPCP.SYS, a DOS TSR NORPAPI.EXE, and Windows printer driver NOR4800.DRV/UNIDRV.DLL. These device drivers work together to provide transparent NPCP printing. Applications use the standard Windows printing API *(StartDoc()*, *EndDoc()*, etc.) to print to a printer set up for LPT1.DOS output.

Installation

Place the files in the following directories:

File	Directory
NORNPCP.DRV	WINDOWS\SYSTEM
NOR4800.DRV	WINDOWS\SYSTEM
UNIDRV.DLL	WINDOWS\SYSTEM

Configuration

Required WIN.INI Entries

```
[windows]
spooler=
DosPrint=no
device=NORAND 4800 Series,NOR4800,LPT1.DOS
```

where:

spooler= tells Windows not to use Print Manager to spool printing.

DosPrint=no tells Windows not print directly to the printer port.

device=NORAND 4800 Series,NOR4800,LPT1.DOS specifies the default printer for Windows. See the following `[PrinterPorts]` and `[devices]` sections of the WIN.INI file.

```
[PrinterPorts]
NPCP Printer=NOR4800,LPT1.DOS,15,45
```

NPCP Printer=NOR4800,LPT1.DOS,15,45 tells Windows that there is a printer named *NPCP Printer* that uses the NOR4800 printer driver to print, using the DOS device LPT1 with 15-second device time-out and 45-second retry time-out.

```
[devices]
NPCP Printer=NOR4800,LPT1.DOS
```

NPCP Printer=NOR4800, LPT1.DOS tells Windows that there is a printer named *NPCP Printer* that uses the NOR4800 printer driver to print, using the DOS device LPT1. This section is included for compatibility with older Windows applications and should match the entry in the `[PrinterPorts]` section of the WIN.INI file (listed above).

Required SYSTEM.INI Entries

```
[boot]
drivers=power irda pen penwindows scanner npcp
```

drivers= tells Windows to load the installable drivers. This list of drivers is the standard driver list (in the order that it needs to be). The significant driver here, is **npcp**. This is just a token name for the driver. The actual driver name is specified in the following `[drivers]` section of the SYSTEM.INI file:

```
[drivers]
npcp=nornpcp.drv
```

npcp= tells Windows the actual name of the installable driver named in the `[boot]` section of the SYSTEM.INI file (listed above).

```
[NPCP Driver]
PrtPort= 0
deviceName=LPT1
CommAddress=0x318
CommVector=0x0C
FIFODepth=16
```

PrtPort= PrtPort specifies the printer port to use; 0=LPT1, 1=LPT2, 2=LPT3. This option must match the options set in the CONFIG.SYS file and the WIN.INI file (above).

deviceName= name of printer device to output to (LPT1, LPT2, or LPT3).

ComAddress= Communications port address.

ComVector= Communications IRQ vector.

FIFODepth= Determines FIFO usage on UART.

Usage

Communications Port Usage

The NPCP printer driver uses COM1 communication hardware directly. The COM1 hardware is initialized and the external RS-232 driver hardware is powered on, thus allowing the driver a “free and clean” data path to the printer. This makes it necessary to be careful when sharing the COM1 hardware with any other driver. Currently, there is no method of arbitrating COM1 access, so drivers must be enabled and disabled only when they are needed. This means that printing in the background and trying to use another driver that talks to the COM1 hardware causes undetermined problems.

Basic Windows Printing

Once Windows is loaded, printing is accomplished through the standard printing API. No special processing by the application is needed. The following sequence demonstrates the correct method for printing under Windows:

1. Create a device context for the printer. *GetPrinterDC()* creates the device context.
2. Set up the device context with the desired fonts, etc. Keep in mind that different fonts cause Windows to print completely in graphics mode, which may provide less performance than desired. *SelectObject(...)* sets the desired font.
3. Set an abort procedure to handle errors during printing and allow the user to cancel the print job. *SetABortProc(...)* sets an abort procedure.
4. Begin the printing operation by calling *StartDoc(...)* to inform GDI that you want every output to be one job.
5. Begin each page of output by calling *StartPage(...)* to allow GDI to begin sending data to the printer on a page-by-page basis.
6. Create your page using standard GDI API like *TextOut(...)*. The output operations used take text and other objects and convert them into a binary data stream that contains printer-specific escape codes. This data is "spooled" until the entire page is composed.
7. Use *EndPage(...)* to finish off a page. The entire page is then sent to the printer. A form feed is generated to move the paper to the top of the next page.
8. Use *EndDoc()* to complete the printing operation. *AbortDoc()* can cancel a job. GDI cleans things up, if possible.
9. Free the instance of your abort procedure using *FreeProcInstance(...)*.

Refer to the Windows SDK manual for more API information.

Default Error Handling Mode

Once the drivers are loaded as defined above, no special processing by the application is needed. This makes it possible for off-the-shelf packages to take advantage of NPCP. In this mode, all printer-related errors are handled by the NPCP driver. A message is displayed and the user has the ability to Cancel or Retry. If the user selects Retry, the driver attempts to continue printing. If the user selects Cancel, the driver attempts to clean up as best it can. The standard Windows API returns a standard error, indicating that a problem occurred. This is the application's signal to abort the print job.

Application-Defined Error-Handling Mode

This is probably the most common mode of operation. In this mode, the application can link to the NPCP driver in the same manner that it would link to any Windows DLL. Once linked, the application can use the driver's API to perform various operations. The application registers a callback with the driver. The application needs to perform the following actions to define a new error handler:

1. Link to driver and obtain PrtService entry point.
2. Call PrtService to enable the driver using *PRT_ENABLE (0x0001)*.
3. Call PrtService to register the new handler using *PRT_SETPROC (0x0010)* and the address of the new handler. The new handler should have the following prototype:

```
extern "C" WORD _export FAR PASCAL ShowPrtError (WORD wCurErr)
```

This callback is called for each error and must return *IDCANCEL (0x02)*, *IDRETRY (0x04)*, or 0. If the application returns 0, the default handler processes the error. For consistency, the application should handle all the errors. You can use any method for displaying the message you want. It is best to make your dialog model if possible. Remember that, depending on the method of displaying the message, you may need to translate the return values into *IDCANCEL* or *IDRETRY* so the printer driver knows how to deal with it. See the Error Codes and Messages topic later in this subsection for the various error conditions that can occur.

4. Perform normal printing operations.
5. Call *PrtService* to remove the error handler, using *PRT_SETPROC (0x0010)* and a NULL value for the handler address.
6. Call PrtService to disable the driver, using *PRT_DISABLE (0x0002)*.

See the following *Printer Services API* topic for more information about available printer services.

Printer Services API

The NPCP printer driver includes an Applications Program Interface (API) to provide access to the features of the NPCP driver that are not available through the standard Windows API.

Retrieving the API Entry Point PrtService

To use the API, the application must first obtain the entry point for the PrtService procedure as described in the following code fragment:

```
hInstNorPrnt = LoadLibrary( “NORNPCP.DRV” );
if ( hInstNorPrnt <= HINSTANCE_ERROR ) {
  MessageBox( NULL,
              “Could not open printer driver.”,
              “ERROR”,
              MB_OK | MB_ICONEXCLAMATION );
  return FALSE;
}
lpfnPrtService =
  (fpPrtService)GetProcAddress(hInstNorPrnt, “PrtService”);
if ( lpfnPrtService == NULL ) {
  MessageBox( NULL,
              “Unable to get address for\nPrtService”,
              “ERROR”,
              MB_OK | MB_ICONEXCLAMATION );
  return FALSE;
}
```

Calling PrtService

NORNPCP.DRV contains an exported procedure *PrtService (HINSTANCE hInst, WORD wOpt, LPARAM lParam1, LPARAM lParam2)* that provides the API for the driver. The parameters to the procedure are as follows:

HINSTANCE hInst	instance handle to our application.
WORD wOpt	option for action you want to perform.
LPARAM lParam1	doubleword that depends on the particular option.
LPARAM lParam2	doubleword that depends on the particular option.

The value returned by PrtService depends on the option selected. See specific options for more details. The entry point can be called just like any Windows procedure, as shown in the following fragment:

```
if ( lpfnPrtService (hInst, PRT_ENABLE, (LPARAM)NULL,
                     (LPARAM)NULL) < 1) {
  MessageBox( NULL,
              "Unable to enable printer driver",
              "ERROR",
              MB_OK | MB_ICONEXCLAMATION );
  return FALSE;
}
```

Supported PrtService Options

PrtService provides the following services:

- *Enable Driver.* Enables the driver and installs the default error handler.
  wOpt should be *PRT_ENABLE (0x0001)*
  1Param1 should be 0
  1Param2 should be 0
- *Disable Driver:* Disables the driver.
  wOpt should be *PRT_DISABLE (0x0002)*
  lParam1 should be 0
  lParam2 should be 0
- *Query Driver for support of an option:* Determines whether the driver supports a particular feature.
  wOpt should be *PRT_GETSUPPORT (0x0003)*
  lParam1 should be the value of the option you want to check.
  lParam2 should be 0
- *Install/Remove External Error handler:* Installs or removes the applications error handler.
  wOpt should be *PRT_SETPROC (0x0010)*
  lParam1 should be far pointer to your error handling procedure. A value of 0 removes the handler. Call *MakeProcInstance(...)* to obtain the value.
  lParam2 should be 0
- *Flush Driver:* Flushes all the data buffers so that everything that has been sent to the printer actually gets printed.
  wOpt should be PRT_FLUSH (0x0020)
  lParam1 should be 0
  lParam2 should be 0
- *Forward:* Advances the paper a certain number of lines. Use this option with caution because Windows is not aware of the paper movement.
  wOpt should be *PRT_FORWARD(0x0030)*
  lParam1 should be the number of lines to advance
  lParam2 should be 0
- *Rewind:* Rewinds the paper back into the printer a certain number of lines. Use this option with caution because Windows is not aware of the paper movement. It may also cause a head jam.
  wOpt should be *PRT_REVERSE(0x0040)*
  lParam1 should be the number of lines to advance
  lParam2 should be 0

Special Paper Handling

The PrtService API supports two paper-handling options, *PRT_FORWARD* and *PRT_REVERSE*, that should be used with care. They are provided to allow reports to be "voided" after they have been printed and inspected. Windows maintains an internal value that "knows" where the current page is relative to the top of form and end of form. If you use these options, Windows can get confused.

The printer also knows where it is on the page (every 11 inches is a new page, which may not match the actual paper if the top-of-form button is used incorrectly). Windows always produces a form feed at the end of a page to advance the paper to the top of the next page. These options are typically used around the printers top-of-form marker, which confuses things even more. Different sizes of paper behave differently and may jam the printer.

1. Generate your entire page (may be only half of the page) and call *EndPage(...)*. This flushes everything printed to the printer and performs a form feed. This places the print position up at the top of the next page.
2. Call StartPage(...) to reset the Windows device context so that you can do more output.
3. Use *PRT_FORWARD* to advance the paper far enough so that the user can determine whether the page is correct. Use a small number of lines, if you can, especially if you are using multipart forms. If possible, do not advance at all. This may be possible if you have not printed on the bottom inch of the page or if the information is not critical. Ask the user to determine whether the page is correct.
4. Use *PRT_REVERSE* to rewind the paper back into the printer the same number of lines used in step 3 (assuming the paper is advanced in step 3). This moves the print position to the top of the form.
5. Use *PRT_REVERSE* to rewind the paper back into the printer to get to the spot at which to print your validation information. Rewind past the top-of-form marker so that the *EndPage(...)* in step 7 moves the print position to the top of the next page. If not rewinded far enough, *EndPage(...)* causes it to skip an entire page. To print on the last line of the page, a value of 2 works, assuming *TextOut(...)* starts at 0 for the Y value.
6. Use *EndPage(...)* to flush everything printed to the printer and performs a form feed. The print position moves to the top of the next page and, if right, Windows is reset so that the next *StartPage(...)* gets things going again.

Print something on the page regardless of whether the user said the page was correct. This validates the page only if the valid message is on the page. Errors can occur while the system attempts to print the void message which may not appear on the paper. If the absence of a void message means that it is valid, an invalid page may be accepted as valid if the void message can not be printed.

NPCP Printer Driver Error Codes and Messages

Error #	Error Code	Error Message	Explanation
102	PNRDY	Printer Not Ready Check Connection	Printer not properly connected.
104	RXTMO	Not Receiving Check Connection	Printer stopped sending data in a packet.
106	TXTMO	Not Transmitting Check Connection	Printer stopped receiving data in a packet.

Error #	Error Code	Error Message	Explanation
111	BADADR	MAC Address Error	Printer rejected MAC address sent in a packet.
112	GAPERR	MAC Interchar Gap Error	Length between bytes too long.
113	LSRPE	MAC Length Parity Error	Actual and declared packet sizes did not match.
120	IFTS	Invalid Comm Frame	Frame sent in wrong sequence.
121	NS_NE_VR	NS <> VR Error	Received NS did not match expected NR.
122	NR_NE_VS	NR <> VS Error	Received NR did not match expected NS.
123	RLENERR	Receive Length Error	Actual and declared packet sizes did not match.
124	MAC_CRCERR	CRC Error	Invalid CRC.
200	FRMERR	Frame Reject Error	Frame was rejected.
201	FRMERR_IF	Invalid Frame	Frame receive was invalid for current mode.
202	FRMERR_NR	NR Mismatch	NR did not match.
203	FRMERR_NS	NS Mismatch	NS did not match.
204	NDMERR	Disconnect Error	Printer is in NDM.
205	FRMERR_SF	Frame Too Short	Frame was too short.
206	FRMERR_LF	Frame Too Long	Frame was too long.
210	BINDERR	Bind Error	Invalid bind sequence.
221	IPLDUR	Invalid PLDU	Invalid Presentation Layer Data Unit.
222–227	(see below)	(see below)	Messages for DATA WAS RCVD.
242–247	(see below)	(see below)	Messages for DATA WAS LOST.
255	DEVERR	Device Error	A device error occurred.
XXX		Unknown Error	Unrecognized error occurred.

The following messages are applicable when DATA WAS RECEIVED:

Error #	Error Code	Error Message	Explanation
222	HEADJAM1	Head Jam	Head jam. All data was flushed.
223	PAPEROUT1	Paper Out	Printer is out of paper.
224	LOWVOLTS1	Low Voltage	Printer undervoltage condition.
225	HIVOLTS1	Over Voltage	Printer overvoltage condition.
226	LOWBAT1	Low Battery	Printer battery is low.
227	COVEROFF1	Cover Off	Cover was removed.

The following messages are applicable when DATA WAS LOST:

Error #	Error Code	Error Message	Explanation
242	HEADJAM2	Head Jam	Head jam. All data was flushed.
243	PAPEROUT2	Paper Out	Printer is out of paper.
244	LOWVOLTS2	Low Voltage	Printer undervoltage condition.
245	HIVOLTS2	Over Voltage	Printer overvoltage condition.
246	LOWBAT2	Low Battery	Printer battery is low.
247	COVEROFF2	Cover Off	Cover was removed.

IrDA Printing for Windows: NOR6805.DRV

The IrDA printing support, under Windows, consists of the DOS device driver, NORIRDA.DRV, and the Windows printer driver, NOR6805.DRV. These device drivers work together to provide transparent IrDA printing. Applications use the standard Windows printing API (*StartDoc(), EndDoc(),* etc.) to print to a printer set up for LPT2.DOS output.

Installation

Place the files in the following directories:

File	Directory
NORIRDA.DRV	WINDOWS\SYSTEM
NOR6805.DRV	WINDOWS\SYSTEM
UNIDRV.DLL	WINDOWS\SYSTEM

Configuration

Required WIN.INI Entries

The sections of the WIN.INI file, that appear on the next few pages, are necessary to set up the printer device parameters for IrDA printing:

```
[windows]
spooler=
DosPrint=no
device=IrDA Printer,NOR6805,LPT2.DOS
```

where:

spooler= tells Windows not to use Print Manager to spool printing.

DosPrint=no tells Windows not to print directly to the printer port.

device=IrDA Printer,NOR6805,LPT2.DOS specifies the default printer for Windows.

The following section of the WIN.INI file are necessary to set up the paper parameters for the printer.

```
[NORAND 6805,LPT2.DOS]
PaperSize=256
SizeUnit=1
PaperWidth=480
PaperLength=1450
```

where:

PaperSize= specifies the size that Windows uses. For the 6805 Printer -- 258 selects “standard roll”, which has a fixed width of 1.89 inches and a fixed length of 11 inches. 256 selects a user-defined paper size, which mandates the use of the SizeUnit, PaperWidth, and PaperLength statements.

SizeUnit= establishes the units that specify paper width and length. For the 6805 Printer:

- “1” indicates a unit of 0.1 millimeter (the default)
- “2” indicates a unit of 0.01 inch

PaperWidth= For the NORAND 6805 Printer, paper width is fixed at 480 (SizeUnit=1; 48 millimeters = 1.89 inch) or 189 (for SizeUnit=2)

PaperLength= For the NORAND 6805 Printer, length values can be from 60 to 2794 (6 to 279.4 millimeters for SizeUnit=1) or 24 to 1100 (0.24 to 11.00 inches for SizeUnit=2. A value of 1450 for SizeUnit=1 sets the length at 5.7 inches.

NORAND 6805=NOR6805,LPT2.DOS,15,45 tells Windows that there is a printer named *NORAND 6805* that uses the *NOR6805* printer driver to print, using the DOS device LPT2 with 15-second device time-out and 45-second retry time-out.

```
[devices]
NORAND 6805=NOR6805,LPT2.DOS
```

The following `[PrinterPorts]` and `[devices]` sections of the WIN.INI file sets up the printer name and printer driver.

```
[PrinterPorts]
IrDA Printer=NOR6805,LPT2.DOS,15,45
```

where:

IrDA Printer=NOR6805,LPT2.DOS,15,45 tells Windows that there is a printer named *IrDA Printer* that uses the *NOR6805* printer driver to print, using the DOS device LPT2 with 15-second device time-out and 45-second retry time-out.

```
[devices]
IrDA Printer=NOR6805,LPT2.DOS
```

where:

IrDA Printer=NOR6805,LPT2.DOS tells Windows that there is a printer named *IrDA Printer* that uses the *NOR6805* printer driver to print, using the DOS device LPT2. This section is included for compatibility with older Windows applications and should match the entry in the [PrinterPorts] section of the WIN.INI file (listed previously).

Required SYSTEM.INI Entries

```
[boot]
drivers= power irda pen penwindows scanner npcp
```

where:

drivers= tells Windows to load the installable drivers. This list of drivers is the standard driver list (in the required order). The significant driver here, is irda. This is a token driver name. The actual driver name is specified in the [drivers] section of SYSTEM.INI:

```
[drivers]
irda=norirda.drv
```

irda= tells Windows the actual name of the installable driver named in the `[boot]` section of the SYSTEM.INI file.

```
[IRDA Driver]
deviceName=LPT2
UARTAddress=1000
UARTIRQ=14
Technology=6100
PrinterFlush=TRUE
PrtPort=0
```

DeviceName= is the name of the printer device for output (LPT1, LPT2, LPT3). This option must match settings in WIN.INI (listed previously).

UARTAddress= sets the base address on which the driver looks for the IR UART. The default setting is 760 (2f8h). Valid settings are 1016 (3f8h for COM1), 760 (2f8h for COM2), 1000 (3e8h for COM3). Required setting for the 6100 Computer: 1000 (3e8h for COM3).

UARTIRQ= sets the IRQ line on which the UART interrupts. The default setting is 3 (COM2). Other valid settings: 4 (COM1), 14. Required setting for the 6100 Computer: 14.

Technology= sets the type of the hand-held computer. The default setting is PENKEY. Required setting for the 6100 Computer is: 6100.

PrinterFlush= tells Windows to flush the printer's input buffer, as part of the printer initialization stream.

PrtPort= PrtPort specifies the printer port to use; 0=LPT1, 1=LPT2, 2=LPT3. This option must match the options set in CONFIG.SYS and WIN.INI.

The following section of the SYSTEM.INI file sets up the parameters for the NORAND 6805 printer:

```
[NORAND 6805 Printer]
DoGraphicsOnly=FALSE
Timeout=10
WakeupChars=200
```

DoGraphicsOnly= forces the drivers to use Graphics commands for the entire document. The default value is FALSE. Valid settings are TRUE and FALSE (that is, graphics commands are used whenever device fonts are not used).

Timeout= sets idle time in seconds for a period of printer inactivity, after which a wakeup is issued to the printer. *Default is 10*. Valid settings are 10 to 50.

WakeupChars= controls the number of NULL characters sent to the printer after the Initialize command, for it to wake up properly. *Default is 200*. Valid settings are 200 to 500. Setting a lower value may disrupt graphics printing, resulting in garbled output to printer.

Usage

Default Error-Handling Mode

Once the drivers are loaded as defined in the preceding paragraphs, no special processing by the application is needed. This makes it possible for off-the-shelf packages to take advantage of the IrDA protocol. In this mode, all printer-related errors are handled by the IrDA driver. A message is displayed and the user has the ability to Cancel or Retry. If the user selects Retry, the driver attempts to continue printing. If the user selects Cancel, the driver attempts to clean up. The standard Windows API returns a standard error, indicating that a problem occurred. This is the application's signal to abort the print job.

Application-Defined Error-Handling Mode

This is probably the most common mode of operation. In this mode, the application can link to the IrDA driver in the same manner that it would link to any Windows DLL. Once linked, the application can use the driver's API to perform various operations. The application registers a callback with the driver.

The application needs to perform these actions to define a new error handler:

1. Link to driver and obtain PrtService entry point.
2. Call PrtService to enable the driver, using PRT_ENABLE (0x0001).
3. Call PrtService to register the new handler, using PRT_SETPROC (0x0010) and the address of the new handler. The new handler should have the following prototype:

```
extern "C" WORD _export FAR PASCAL ShowPrtError( WORD wCurErr )
```

This callback is called for each error and must return IDCANCEL (0x02), IDRETRY (0x04) or 0. If the application returns 0, the default handler processes the error. For consistency, the application should handle all errors. Use any method for displaying the message. Make the dialog model, if possible. Note, based on the method of displaying the message, you may need to translate the return values into IDCANCEL or IDRETRY for the printer driver.

4. Perform normal printing operations.
5. Call PrtService to remove the error handler, using PRT_SETPROC (0x0010) and a NULL value for the handler address.
6. Call PrtService to disable the driver, using PRT_DISABLE (0x0002).

Printer Services API

The IrDA printer driver includes an API to provide access to the features of the IrDA driver that are not available through the standard Windows API.

Retrieving the API Entry Point PrtService

To use the API, the application must first obtain the entry point for the `PrtService` procedure as described in the following code fragment:

```
hInstNorPrnt = LoadLibrary( "NORIRDA.DRV" );
if ( hInstNorPrnt <= HINSTANCE_ERROR ) {
  MessageBox( NULL,
              "Could not open printer driver.",
              "ERROR",
              MB_OK | MB_ICONEXCLAMATION );
  return FALSE;
}
lpfnPrtService =
  (fpPrtService)GetProcAddress( hInstNorPrnt, "PrtService" );
if ( lpfnPrtService == NULL ) {
  MessageBox( NULL,
              "Unable to get address for\nPrtService",
              "ERROR",
              MB_OK | MB_ICONEXCLAMATION );
  return FALSE;
}
```

Calling PrtService

NORIRDA.DRV contains an exported procedure `PrtService( HINSTANCE hInst, WORD wOpt, LPARAM lParam1, LPARAM lParam2)` that provides the API for the driver. The parameters to the procedure are as follows:

HINSTANCE hInst	instance handle to our application.
WORD wOpt	option for action you want to perform.
LPARAM lParam1	double word that depends on the particular option.
LPARAM lParam2	double word that depends on the particular option.

The value returned by `PrtService` depends on the option selected. See specific options for more details. The entry point can be called just like any Windows procedure, as shown in the following fragment:

```
if (lpfnPrtService(hInst, PRT_ENABLE, (LPARAM)NULL,
                                  (LPARAM)NULL ) < 1) {
   MessageBox( NULL,
               "Unable to enable printer driver",
               "ERROR",
               MB_OK | MB_ICONEXCLAMATION );
   return FALSE;
}
```

Supported PrtService Options

PrtService provides the following services:

- *Enable Driver*. Enables the driver and installs the default error handler.

  wOpt should be set to *PRT_ENABLE (0x0001)*
  lParam1 should be set to 0
  lParam2 should be set to 0

- *Disable Driver*. Disables the driver.

  wOpt should be set to *PRT_DISABLE (0x0002)*
  lParam1 should be set to 0
  lParam2 should be set to 0

- *Query Driver for support of an option*. Determines whether the driver supports a particular feature.

  wOpt should be set to *PRT_GETSUPPORT (0x0003)*
  lParam1 should be set to the value of the option you want to check.
  lParam2 should be set to 0

- *Install/Remove External Error handler*. Installs or removes the applications error handler.

  wOpt should be *PRT_SETPROC (0x0010)*
  lParam1 should be a far pointer to your error handling procedure. A value of 0 removes the handler. Call MakeProcInstance(...) to get the value.
  lParam2 should be set to 0

- *Flush Driver*. Flushes all the data buffers so that everything sent to the printer actually gets printed.

  wOpt should be set to *PRT_FLUSH (0x0020)*
  lParam1 should be set to 0
  lParam2 should be set to 0

Error Codes and Messages

As pointed out in the topic "Default Error-Handling Mode," the IrDA printer driver can provide the error handler and thereby translate all printer-related errors into a *Cancel* or *Retry* option for any application.

However, if the application provides the error handler, custom messages can be attached to the error returns. As shown in this table, seven different numbered returns are made available by the driver as generic hook points for messages.

Error #	Error
3	Wait
255	Failed
254	Error Queue Full
253	Error Node Active
252	Error Bad ConHandle
251	Media Busy
xxx	Unknown Error

Any of the listed errors can appear under 12 different operational phases or categories such as: Initialize, Discover, Discover Time-out, Connect, Connect Status Error, Flush – Allocate, Flush – Get, Flush –Append, Flush Queue Full, Send, Disconnect, and Shutdown.

Section 4

Power Management

Introduction

This section contains information about Advanced Power Management (APM) on the PEN*KEY® 6100 Computer. The following information may help you locate information in this section:

Topic Summary

List of Tables

Power Management BIOS: ELANAPM.EXE

Overview

ELANAPM.EXE is the APM BIOS. The APM system consists of one or more layers of software. The APM BIOS resides at the lowest layer.

To provide portability at the higher layers of the APM system, ELANAPM.EXE, an installable DOS device driver, provides a software interface to the hardware that is independent of the hardware interfaces.

To provide diagnostics with proprietary applications, some functions specific to Intermec Technologies Corporation are also provided; and may not be independent of the hardware interfaces.

The layer immediately above the APM BIOS includes the operating system power management driver. For DOS-based systems, the device driver is NORDOSPM.EXE. For Windows applications, the device driver is NORWINPM.DRV, and VPOWERD.386 is included. The operating system driver is responsible for the power management strategy. This includes system power state and device power state transitions. The driver is also responsible for the application interfaces.

The highest layer includes applications that are *APM-aware*. Applications receive power messages only from the APM driver. Applications can communicate with either the APM driver or the APM BIOS.

The objective of APM is to control the power usage of the system according to system activity. As system activity decreases, APM reduces the power consumption of system resources until the system is brought into a suspend state.

There are two methods of power-level control:

- The APM BIOS manages CPU/core logic, display, and backlight power in the background, based on device activity.
- An APM driver (NORDOSPM.EXE or NORWINPM.DRV and VPOWERD.386) participates in managing power levels via function calls to the APM software interface.

System Power States

The APM driver has three default system power states:

- Ready (full on)
- Idle
- APM suspend.

The primary difference between states is the latency needed for the system to reach full operation. Power consumption and performance are greatest in the Ready state and decrease with each succeeding state.

- System Ready is the default mode when the system is not engaged in power management.
- System Idle is the first level of power management. The CPU is halted and the system clock speed is reduced. Transitions from Idle to Ready are instantaneous.
- System Suspend is the second level of power management. The CPU clock is stopped and devices are placed into the power state defined by the application for System Suspend. The device power state for suspend can be any power state defined for that device. The exact definition of a device power state during suspend is performed by using the *Set Power State, Get Power State,* and *Enable/Disable Device Power Management* functions. Transitions from suspend to ready generally take 100 milliseconds.

System Power State Management

System power states are managed through the use of activity monitors, activity timers, and function calls from APM device drivers or APM-aware applications.

Activity monitors are hardware-specific devices that watch for I/O to predefined I/O devices and I/O addresses. The PEN*KEY 6100 system has activity monitors for the following I/O devices. Whenever one of these devices is accessed, a bit in the activity monitor for the device becomes set.

I/O Address (Hex)	Device
000–00F	⎫
080–08F	⎬ DMA (I/O activity to both the controller and DMA page
0C0–0DF	⎭ registers)
022-023	ELAN
060, 064	Keyboard (60h and 64h)
1E8–1EF	Internal scanner
1F0–1F7	PC Card ATA drive
200 207	Touch Screen
2F8–2FF	COM2 (2F8–2FFh)
3B0–3DF	VGA (I/O and memory activity (0A0000h–0BFFFFFh)
3E0–3E1	PC Card (I/O activity, slot control, and page frame access)
3E8–3EF	COM3
3F8–3FF	COM1 (3F8h–3FFh)
220–224	SST radio
2E8–2EF	COM4

Activity timers are defined for System Idle and System Suspend. These timers are enabled by default and their time-out period can be set or retrieved by OEM APM functions *Set Device Activity Timer* or *Get Device Activity Timer*. These timers decrement when no system activity occurs within 0.125 second and are reset to the defined period when activity occurs within 0.125 second. If either Idle or Suspend time-out is set to zero, the activity timer is disabled. When activity timers decrement to zero, the system power state associated with the timer is entered (either System Idle or System Suspend).

Activity monitors can be masked so that activity on a given device are ignored and not cause the activity timers associated with System Idle or System Suspend to be reset when activity occurs on the masked device. OEM APM functions *Set Device Activity Mask* and *Get Device Activity Mask* are provided for this purpose.

Functions that affect system power states are *CPU IDLE*, *CPU BUSY*, and *System Suspend*. *System Suspend* is invoked by the *Set Power State* function for device 0001h. APM Drivers or APM-aware applications may make these function calls.

The *CPU IDLE* call causes the system to enter a low-power state by decreasing the system clock frequency and issuing a HALT instruction to the CPU. The idle state may automatically be entered whenever there is a lack of system activity if the IDLE activity timer is nonzero. The IDLE state is exited whenever an interrupt service routine is invoked that causes I/0 activity or performs a CPU BUSY call. The IDLE state returns to the READY (full power state) in this condition.

The *CPU Busy* call cases the Activity timers associated with System Idle and System Suspend to be reset. This call acts as though an I/O activity was monitored by the APM BIOS, in that it resets all system activity timers and causes the system either to enter READY or remain in the Ready state.

The *Set Power State* function call with a device of 0001h causes the system to enter System Suspend. System Suspend is also entered whenever the SUSPEND activity timer reaches zero or the ON/OFF button is pressed. The System Suspend state is exited only by a wakeup event.

Possible wakeup events (sources) are plentiful and are defined by the 6100 Computer as follows:

- ON/OFF button press
- MODEM ring indicator
- PC Card MODEM ring indicator
- POD 1 ring indicator
- Real Time Clock Alarm expiration
- Application of external power (charge)

Wakeup events can be set or masked via OEM APM function calls *Get Wakeup Mask* and *Set Wakeup mask*.

Device Power Control

Devices may be power-managed either by the system APM BIOS, APM device driver, or APM-aware application via the *Set Power State* function.

Device drivers or applications that control device power should be aware that, when the system enters System Suspend, all peripheral devices managed by the driver or application loses power during System Suspend, but is restored to the previous power state upon System Resume.

Device drivers or applications that use the *Enable/Disable Device Power Management* function can override this functionality during Suspend. These drivers or applications take sole responsibility for device power management during Suspend. This feature allows system devices, the operating system, or application to tell the BIOS to stop power managing a particular device, after which operating-system-specific drivers or applications for that device may directly take over power management.

Devices may be requested to be placed into any one of the following states via the *Set Power State* function.

Table 4-1
Power States (General Definitions)

State	Description
Ready	The device is in its full-power state.
Standby	The device is capable of doing some work but is in a low-power state.
Suspend	The device is not capable of doing work and is in its lowest power state
Off	The device is not capable of doing work and is in its lowest power state

The power states for each device in a 6100 Computer are defined as follows:

Table 4-2
Power States (Display)

Power State	Description
READY	Enables LCD and places VGA controller into full-power state
STANDBY	VGA registers and memory are accessible. LCD is OFF.
SUSPEND	VGA controller placed into hardware suspend
OFF	Same as SUSPEND

Table 4-3
Power States (PC Card)

Power State	Description
READY	PC Card is in the full-power state
STANDBY	PC Card is in software-activated hardware suspend state
SUSPEND	PC Card is in software-activated hardware suspend state
OFF	PC Card is in software-activated hardware suspend state

Table 4-4
Power States (Serial Port)

Power State	Description
READY	If either RS-232 or RS-485 is enabled, the UART CLOCK is enabled. If RS-232 is enabled, the RS-232 transceiver is enabled.
STANDBY	If both RS-232 and RS-485 are disabled, the UART CLOCK is turned off. If RS-232 is off, the RS-232 transceiver is turned off.
SUSPEND	Same as STANDBY
OFF	Same as STANDBY

Table 4-5
Power States (Digitizer)

Power State	Description
READY	The digitizer is placed in its full-power state.
STANDBY	The digitizer is placed in hardware suspend state
SUSPEND	Same as STANDBY
OFF	Same as STANDBY

Table 4-6
Power States (Pod)

Power State	Description
READY	Voltage is distributed to the POD.
STANDBY	(not defined)
SUSPEND	(not defined)
OFF	Voltage is removed from the POD

Table 4-7
Power States (PC Card Slot)

Power State	Description
READY	Power is routed to the slot
STANDBY	(not defined)
SUSPEND	(not defined)
OFF	Power is removed from the slot

Table 4-8
Power States (System)

Power State	Description
READY	No action
STANDBY	No action
SUSPEND	The system is placed into SUSPEND. Any APM-disabled devices are left in the power state they were set to before entering suspend.
OFF	Same as SUSPEND
PROCESSING	No action is performed
REJECTED	No action is performed

Table 4-9
Power States (Backlight)

Power State	Description
READY	The backlight is turned on.
STANDBY	(not defined)
SUSPEND	(not defined)
OFF	The backlight is turned off

APM Software Interface

The following paragraphs describe the software interface between the APM BIOS and the APM driver. Refer to the *Conversions and Interfaces* section of this publication for details of the actual BIOS interrupts.

APM Connection

The APM driver provides an interface between the APM BIOS and APM-aware applications. When the interface between the APM BIOS and device driver is established, the APM driver receives power-related events from the APM BIOS, which the driver in turn broadcasts to APM-aware applications.

The *APM Installation Check* function determines whether ELANAPM.EXE is present in the system, the version of APM that ELANAPM.EXE supports, and the attributes of the implementation. The APM driver issues an APM Connection call to establish the appropriate cooperative connection with the APM BIOS. The *APM Interface Disconnect* function disconnects the cooperative connection between the APM Driver and the APM BIOS. Only one APM connection can be active at a time.

Power Management Events

Power Management Events are communicated from the APM BIOS to the APM driver through a call by the APM driver to the APM BIOS function *Get PM Event*. It is the responsibility of the APM driver to broadcast Power Management Events to its APM-aware applications.

Table 4-10 is a list of the defined power management events that the APM BIOS can report:

Table 4-10
Power Management Event Codes

Code (Hex)	Description
0002	System Suspend Request Notification
0003	Normal Resume System Notification
0004	Critical Resume System Notification
0005	Battery Low Notification
0006	Power Status Change Notification
0008	Critical System Suspend Notification
000A	User System Suspend Request Notification

System Suspend Request Notification – notifies the APM Driver that the APM BIOS wants to put the system into the Suspend State when the System Suspend activity timer times out. If APM BIOS power management is disabled, it is the APM Driver's responsibility to determine whether the system should be placed in the Suspend State. The APM Driver must notify APM-aware applications of the System Suspend request. This notification allows the device drivers the opportunity to prepare for the suspend state change or to reject the request.

To process a suspend request, the APM Driver calls the *Set Power State* function with BX = 0001h (all devices) and CX = 0002h when all processing is complete. The APM BIOS (ELANAPM.EXE) does not enter the suspend state on its own unless APM BIOS power management is enabled via the *Enable/Disable Power Management* call.

Normal Resume System Notification – indicates that a System Resume from a press of the SUSPEND/RESUME key after a normal System Suspend has occurred. The APM driver must update the time from the real-time clock when it receives this notification and notify APM-aware applications of the event. The applications are then responsible for preparing devices for use.

Critical Resume System Notification – indicates that a Critical Resume System operation occurred. The APM driver updates the time from the real-time clock when it receives this notification and notifies APM-aware applications of the event. These applications then prepare their devices for use.

Battery Low Notification – informs the APM driver that the system's battery is running low. The APM driver broadcasts this message to APM-aware applications. The status of the battery must be retrieved by the Get Power Status call.

Power Status Change Notification – informs the APM driver that the system's power status has changed. The APM driver must issue the Get Power Status call to determine the change to the charge status. ELANAPM.EXE signals a Power Status Change Notification event when the system's charger input changes status, when the battery flag changes, or the charge remaining changes.

Critical System Suspend Notification – notifies the APM driver that the APM BIOS has detected a situation in which the system must be suspended and all devices placed in an off state, as defined by the device. The APM must broadcast this message to all APM-aware applications. The driver must not refuse the suspend operation, even if APM-aware applications refuse the notification, and must verify that all devices are in the off state prior to issuing the suspend operation.

After a system has been suspended and the system is resumed, a Critical Resume System Notification power management event is posted by the APM BIOS. The APM driver must notify APM-aware applications and device drivers when the system is resumed from a critical suspend.

User System Suspend Request Notification – notifies the APM driver that the SUSPEND/RESUME button has been pressed and the user wants to put the system into the Suspend State. The APM driver must notify APM-aware applications of the System Suspend request. This notification allows the device drivers the opportunity to prepare for the suspend state change or to reject the request. If the request is to be accepted, the APM driver calls the Set Power State function to set the system state to suspend. The BIOS does not enter the suspend state on its own unless BIOS power management has been enabled via the Enable/Disable Power Management call.

APM Include Files

APMEVENT.H

APMEVENT.H is a C++ include file that defines APM 1.1 event codes for BIOS, DOS, and Windows.

APM 1.1 BIOS event codes	These codes are broadcast to DOS TSRs by the APM OS Driver.
Windows APM 1.1 event codes	These events are broadcast to the Windows drivers and parent windows by the Windows APM driver. Note that these codes are one less than the corresponding BIOS codes. Also, only three codes are actually defined in the Windows 3.1 WINDOWS.H header file.
BIOS, DOS, and Windows OEM-defined APM codes (specific for Intermec Technologies Corporation)	Many of these codes are used by the APM BIOS to communicate with the APM OS driver and are not actually broadcast to DOS or Windows system components.

PMEVENTS.H

PMEVENTS.H contains power management event constants.

For an example of these files, refer to *Appendix B, Common PEN*KEY 6000 Series Information*.

Firmware Error Codes

Table 4-11
Firmware Error Codes

Error (Hex)	APM Function Error Messages	Applicable APM Calls
01	Power Management functionality disabled	Enable/Disable Power Management
02	Real Mode interface connection already established	APM Protect Mode 16-bit Interface Connect
		APM Protect Mode 32-bit Interface Connect
		APM Real Mode Interface Connect
03	Interface not connected	APM Interface Disconnect
		Enable/Disable Power Management
05	16-bit protected mode interface already established	APM Protect Mode 16-bit Interface Connect
		APM Protect Mode 32-bit Interface Connect
		APM Real Mode Interface Connect
06	16-bit protected mode interface not supported	APM Protect Mode 16-bit Interface Connect
07	32-bit protected mode interface already established	APM Protect Mode 16-bit Interface Connect
		APM Protect Mode 32-bit Interface Connect
		APM Real Mode Interface Connect
08	32-bit protected mode interface not supported	APM Protect Mode 32-bit Interface Connect
09	Unrecognized device ID	APM Installation Check
		APM Interface Disconnect
		APM Protect Mode 16-bit Interface Connect
		APM Protect Mode 32-bit Interface Connect
		APM Real Mode Interface Connect
		Enable/Disable Device Power Management
		Enable/Disable Power Management
		Engage/Disengage Power Management
		Get Power State
		Get Power Status
		Restore APM BIOS Power-On Defaults
		Set Power State
0A	Parameter value out of range	Enable/Disable Device Power Management
		Enable/Disable Power Management
		Engage/Disengage Power Management
		Set Power State
60	Unable to enter requested state	Set Power State
80	No power management events pending	Get PM event
86	Reserved – No APM present	APM Installation Check

Refer to the *APM BIOS Interface Specification 1.1* publication, for further details. The *Reference, Open Systems Publications* section contains information for ordering that document.

Section 5

Communications and Device Support

Introduction

This section presents a variety of *process* and *procedural* topics that deal with many of the basic operational features of the PEN*KEY® 6100 Computer. Included are topics such as cold booting, warm booting, the NORAND® Utilities, keyboards, and communications, as well as some information about Screen Emulation.

Topic Summary

Communications Support

Using INTERLNK and INTERSVR

INTERLNK interconnects a PEN*KEY 6000 Series Computer and a PC through serial ports. INTERSVR is the INTERLNK server, a communication option in the NORAND Utilities program.

For detailed information for installing INTERLNK, refer to the *Development Resources* paragraph in the *Getting Started* section. For more details of INTERLNK and INTERSVR topics, refer to the DOS online help.

NORAND Utilities: PSROM0C.EXE

The NORAND Utilities program, PSROM0C.EXE, provides the basic functions required to prepare the PEN*KEY 6000 Series Computer for use. Refer to the *PEN*KEY Model 6100 Hand-Held Computer User's Guide* (P/N: 961–028–085) for detailed information on the use of NORAND Utilities.

The following paragraphs provide information relating to the incorporation of communications facilities of the NORAND Utilities into the PEN*KEY 6000 Series system.

System Setup Requirements

To use PSROM0C.EXE, the file NRTCMERR.TBL must exist in the current working directory or in the PATH.

When using PSROM0C.EXE Ver. 2.00 or later, if the application uses CardSoft device drivers to access PC Cards, it must reassign the drive letters A and B. For example: `ASSIGN.COM A:=E: B:=F:`

NPCP

To use NPCP, you must load the program MININET.EXE. You can install this program by inserting the following statement into the CONFIG.SYS file:

```
INSTALL=MININET.EXE
```

or by placing the following statement into the ROMINIT.BAT or AUTOEXEC.BAT file: `<path>\MININET.EXE`

Where:
<path> is the drive letter and pathname where MININET.EXE is located.

TTY

To use the TTY protocol, the NRTTYM.TBL file must exist in the current directory or in the Path. NRTTYM.TBL is the modem table. For information about creating a customized modem table, see the *PSMDM0C, DOS HHC MODEM TABLES* product, P/N: 215-968-001.

If the application uses NORMOD.SYS or NGENMOD.SYS to support PC Card modems but uses different command line settings than the default Norand Utilities configuration, the command line for the driver must include "–NMODEMn" where "n" is the COM port number used by the driver.

NRInet Using PSROM0C Version 3.xx

To use the NRInet protocol version 3.xx, the PATH must include the directories of the following drivers:

- LSL.COM
- ODIPKT.COM
- RS485ODI.COM

Optionally, configure a DHCP server to provide IP information required by the computer, such as the IP addresses of the client, server, router and domain name servers.

The NET.CFG and WATTCP.CFG files are overwritten by PSROM0C.

Some computers support two network interfaces for NRInet: Ethernet and RS-485. The network interface is selected in a menu in Norand Utilities during the IPL sequence. This setting is stored on the RAM drive so it is lost any time the RAM drive is removed or formatted. For example, applications that boot from a PC Card often format the RAM drive before installing the application. In these cases, return to the Norand Utilities menu to reselect the setting in order for NRInet sessions to work correctly.

NRInet Using PSROM0C Version 2.xx

To use the NRInet protocol, the PATH must include the directories of the following drivers, which are loaded and unloaded by PSROM0C.EXE:

```
BOOTP.EXE
DHCP.EXE
ETHDRV.EXE
INET.EXE
LSL.COM
ODIPKT.COM
RS485ODI.COM
```

BOOTP.EXE and DHCP.EXE are required only if IP information is to be retrieved from a BOOTP server or a DHCP server as described below. PSROM0C.EXE executes BOOTP.EXE first. DHCP.EXE is executed only if BOOTP.EXE fails.

The files NET.CFG and PCTCP.INI are required to load these drivers. NET.CFG must exist in the current working directory, and must include the following two lines, indented as shown:

```
Link driver RS485ODI
  Frame Ethernet_II
```

PCTCP.INI must include:

```
[pctcp ifcust 0]
async-send = yes
ip-address = nnn.nnn.nnn.nnn
subnet-mask = nnn.nnn.nnn.nnn
router = nnn.nnn.nnn.nnn

[pctcp general]
etc-dir = d:\tcp\etc
domain = xxxxxxxx.com

[pctcp addresses]
domain-name-server = nnn.nnn.nnn.nnn

[pctcp kernel]
interface = ifcust 0
```

The “etc-dir” field must specify a directory that contains a SERVICES file. A minimal SERVICES file exists in the flash at `d:\tcp\etc`.

For NRInet, SERVICES must contain a protocol entry for “nrinet” with a protocol type of “tcp.” For example:

```
nrinet 44965/tcp #Norand Inet File Transfer
```

The fields “domain” and “domain-name-server” are not required if the host is specified by IP address rather than domain name. Also, the fields that specify IP addresses (nnn.nnn.nnn.nnn) and the “domain” field can be omitted if BOOTP.EXE or DHCP.EXE loads the information from a BOOTP server or a DHCP server.

Set the environment variable PCTCP to the path of PCTCP.INI. For example, in AUTOEXEC.BAT:

```
SET PCTCP=C:\PCTCP.INI
```

Some computers support two network interfaces for NRInet: Ethernet and RS-485. The network interface is selected in a menu in Norand Utilities during the IPL sequence. This setting is stored on the RAM drive so it disappears any time the RAM drive is removed or formatted. For example, applications that boot from a PC Card often format the RAM drive before installing the application. In these cases, you must return to the Norand Utilities menu to reselect the setting in order for NRInet sessions to work correctly.

When the RS-485 network interface is used on a computer with an Ethernet network interface, PSROM0C.EXE powers off the the Ethernet adapter. If the application uses the Ethernet adapter outside of PSROM0C.EXE, it is recommended to unload the Ethernet drivers before calling PSROM0C.EXE, then reload them afterwards.

TCOM Session Overview

Under the NORAND standard file transfer session, the HHC always uploads first. Once all upload files are sent, the line is turned around and the host then sends any download files. Each file is preceded by a header record that gives the receiving computer information about the format used by the file.

The HHC must first send a session control file, which identifies the HHC to the host computer. The HHC may then send a download request file. This optional file prepares the host computer to download one or more files to the HHC. Upload files can then be sent to the host.

Session Control File

To identify itself to the host, the HHC sends the session control file at the beginning of a TCOM session. The application-dependent HHC ID is used by the host to determine which files to downloading to the HHC. The format of the session control file is as follows:

```
<DSCNTRL00001Xnnn>PPPPPPPPPPPPPPTTTTTTTTTTTTTTTTTTYYMMDDHHMMSS . . .
```

where:

```
               < = beginning of file header
               D = file type (fixed)
          SCNTRL = file name (fixed)
           00001 = decimal number of records in file (fixed)
               X = data type (ignored by host)
             nnn = decimal number of bytes in file record
               > = end of file header
  PPPPPPPPPPPPPP = program identification
TTTTTTTTTTTTTTTT = hand-held computer ID (determined by application)
          YYMMDD = date
          HHMMSS = time
           . . . = additional HHC information
```

The application determines the actual data in the session control file. The data fields shown in the preceding list represent a convention used in applications from Intermec Technologies Corporation. However, the session control header always consists of 18 bytes. Also, NORAND host communication packages usually expect to find the HHC ID starting in byte 33 of the data stream. Although, this location is configurable.

Download Request File

An HHC may directly request specific files from the host. To do this, it sends a download request file to the host immediately after sending the session control file. The requested files are not downloaded at that moment, but rather after the line is turned around.

If a download request file is sent, the host does *not* use the HHC ID in the session control file to determine which files to download to the HHC. Only files requested in the download request file are downloaded.

▶ **NOTE:** *The download request file must be the first or second file sent. Otherwise, it is not treated as a special file by the host.*

The format of the download request file is as follows:

```
<DDWNLRQnnnnnX016>[--filename1---][--filename2---] . . .
```

where:

```
               < = Beginning of file header
               D = File type (fixed)
          DWNLRQ = File name (fixed)
           nnnnn = Decimal number of records in file
               X = Data type (X for character)
             016 = Decimal number of bytes in record (fixed)
               > = End of file header
[--filename?---] = Name of file to download, must be left-justified and blank-
                   padded to 16 characters
```

You can include as many filename records as are specified in the number of records field of the header. This permits requests for more than one file to be batched together.

Upload and Download Files

The general format of a file header is as follows:

```
<tffffffmmmmmtnnntnnntnnn . . . >
```

where:

```
              < = Beginning of file header
              t = File type
                  'D' for unpacked data
                  'E' for unpacked executable
                  'P' for packed data
                  'B' for packed executable
         ffffff = File name
          mmmmm = Decimal number of records in file, right-justified and zero-
                  padded
              t = Field data type (described below)
            nnn = Unpacked length of field, right-justified and zero-padded
              > = End of file header
          . . . = Additional HHC information
```

The actual file name created on the handheld has ".DAT" or "P.PL6" appended to the ffffff file name. File types 'D' and 'P' have ".DAT" extensions. File types 'E' and 'B' have "P.PL6" appended to the name. Valid field data types are:

Type	C data type	Comments
B	unsigned char	Field length in the file header can be 001 to 003. Valid field values are 0–255.
D	signed long	Field length in the file header can be 001 to 011. Valid field values are –2147483647 to 2147483647. '+' is not uploaded for positive values. Leading zeroes are uploaded if necessary to meet the specified field length.
I	signed int	Field length in the file header can be 001 to 006. Valid field values are –32767 to 32767. '+' is not uploaded for positive values. Leading zeroes are uploaded if necessary to meet the specified field length.
N	N/A	Field type is specific to the PL/N programming language from Intermec Technologies Corporation. See the PL/N Reference Manuals for more information.
S	char[ ]	Specifies a NULL terminated string. Field length does not include the NULL, as the NULL is not included in the unpacked file. The NULL is inserted when the field is written to the HHC file.
W	N/A	Specific to the PL/N programming language from Intermec Technologies Corporation. For further information, see the PL/N Reference Manuals. This type is equivalent to an unsigned integer in field size, but is packed in the opposite byte order.
X	char[ ]	Specifies a character buffer that does *not* have a NULL terminator.
(	N/A	Marks the beginning of a repeated field descripter sequence. The field length is the number of repetitions of the sequence. For example, the sequence: B002X004B002X004 can be written as: (002B002X004)000 "(002" marks the beginning of a pattern repeating twice. ")000" marks its end.
)	N/A	Marks the end of a repeated sequence begun with a "(nnn" descripter. Field length is the number of repetitions of the sequence. For example, the sequence: B002X004B002X004 can be written as: (002B002X004)000 "(002" marks the beginning of a pattern repeating twice. ")000" marks its end.

In the NORAND Host Communication packages, a record length of one indicates that the file contains variable length records, where the first character of each record identifies the type of record. These single character record types are used by the formatting utilities of the communications packages to format the file into logical records.

PL/N File Descriptor for Binary Files

For the support of full DOS file names and for better support of non-PL/N binary files, an expanded header structure is defined:

```
<DDOSFIL00001Xmmm>F[--dosfilename--] S[filesize]data. . .

                < = beginning of file header
                D = file type (fixed)
           DOSFIL = file name (fixed)
            00001 = fixed
                X = data type (fixed)
              mmm = decimal number of bytes of file information between end
                    of file header and start of file
                > = end of file header
                F = F indicates the file name parameter
[--dosfilename--] = complete DOS file name (may be of any length)
         <space> = a space must separate the parameters
               S = S indicates the file size parameter
      [filesize] = the exact number of bytes in binary file (this parameter
                    may be up to 8 digits long)
       data. . . = the binary file starts immediately after filesize parameter
           . . . = additional hand-held computer information
```

For example, to send a file to be named \DATA\DATAFILE.DAT that has a size of 102,000 bytes, the following header would precede the file:

```
<DDOSFIL00001X027>F\DATA\DATAFILE.DAT S102000
```

It is the responsibility of the PEN*KEY 6000 Series application program to ensure that the file name specified is unique on the host. This could be accomplished by incorporating the hand-held computer ID as part of the file name or path name:

```
<DDOSFIL00001X027>F\DATA\DATA0001.DAT S102000
<DDOSFIL00001X031>F\HH000001\DATAFILE.DAT S102000
```

Usage

To use PSROM0C.EXE, use the following syntax on the command line:

```
PSROM0C.EXE ctl-file
```

where *ctl-file* is the name of a text file containing parameters that control the telecommunications session. Valid parameters are described in the following paragraphs. Parameters that are not applicable may be omitted.

The return value from PSROM0C.EXE is a session status code. Values for this code are described in the *Communications Log File* paragraph, page 5-11.

COM=

Valid only if two-way TTY protocol is specified. It indicates the COM port to use for two-way TTY communications. Any value can be supplied; valid values are determined by the application (such as the DOS power management driver) that utilizes this parameter. *(Example: COM=4)*

Default value: 1 (COM1)

CONFIG=

This is a modem command string to configure a Hayes-compatible modem. *(Example: CONFIG=ATE0V0Q0X4&C1&D2&M0&RS0=0)*

DATABITS=

Valid only if two-way TTY protocol was specified. *(Example: DATABITS=7)*

Valid values:
7, 8

IPCLIENT=

Supported in PSROM0C V3.xx only. This is the IP address of the client computer. If this parameter is specified, the NETMASK parameter is also required. If this parameter is omitted, PSROM0C attempts to retrieve IP information from a DHCP server or a BOOTP server.

IPHOST=

This is the IP address or domain name of the Ethernet host. The value can be at most 18 characters long. If this parameter specifies a domain name, the IPCLIENT parameter must be omitted because DHCP information about domain name servers is required to resolve the host name. For PSROM0C V3.xx, if IPHOST and IPCLIENT are both omitted, the host name is retrieved from DHCP if available. Otherwise, this parameter defaults to “Norand6920.”

MODEMSELECT=

Valid only if two-way TTY protocol was specified. This is the modem ID of of a record in the modem table file NRTTYM.TBL. If this parameter is specified, the COM= and CONFIG= parameters may be omitted. The parameters MODEMSELECT= and MODEMTYPE= are mutually exclusive; therefore only one may be specified in a control file. *(Example: MODEMSELECT=30)*

MODEMTYPE=

Valid only if two-way TTY protocol was specified. It indicates the type of modem, if any, to which the hand-held computer is connected. The parameters MODEMSELECT= and MODEMTYPE= are mutually exclusive; therefore only one may be specified in a control file. *(Example: MODEMTYPE=1)*

Valid Values:

0 = No modem, i.e. direct connect (default)
1 = NORAND modem, or other Hayes-compatible modem

NETMASK=

Supported in PSROM0C V3.xx only.
This is the local subnet mask and is required if IPLCIENT is specified. *(Example: NETMASK=255.255.240.0)*

NPCPHOST=

NPCP host name.

Valid values:

`NORAND_HOST`
`NORAND_SERVER`

PARITY=

Valid only if two-way TTY protocol was specified. *(Example: PARITY=2)*

Valid values:

0 = none (default)
1 = odd
2 = even

PHONE=

Dials a Hayes-compatible modem. This value can be at most 20 characters long. *(Example: PHONE=ATDT3693361)*

PROTOCOL=

Indicates the protocol to be used. *(Example: PROTOCOL=3)*

- 3 = TTY
- 4 = NPCP (LAN)
- 14 = NRInet

ROUTER=

Supported in PSROM0C V3.xx only.
This is the router IP address.

SPEED=

Indicates the TCOM speed for two-way TTY. This parameter should be omitted if NPCP protocol is specified. *(Example: SPEED=2400)*

Valid values:

1200, 2400, 9600, 19200, 38400, 57600, or 115200.

STOPBITS=

Valid only if two-way TTY protocol was specified. *(Example: STOPBITS=1)*

Valid values:

1 = One stop bit
2 = Two stop bits
3 = 1–1/2 stop bits

TRIES=

Specifies the number of times to attempt a successful communication session. If TRIES is exhausted before a session is successful, PSROM0C.EXE returns the error of the last session. A parameter value of 0 indicates that the retry is to continue until a session is successful or the user aborts.

EXAMPLE: Example Control File for TTY

```
PROTOCOL=3
COM=1
SPEED=9600
PARITY=0
DATABITS=8
STOPBITS=1
```

EXAMPLE: Example Control File for NPCP

```
PROTOCOL=4
```

EXAMPLE: Example Control File for NRInet

```
PROTOCOL=14
IPHOST=nnn.nnn.nnn.nnn
```

Upload Control File

Information about files being transferred must be in a NRUPLD.CTL file. Valid parameters for this file are described in the following paragraphs.

Descriptions of File Entries

HEADER=

This file header record precedes the files subsequently specified in FILE parameters. This header record provides the host information on data formatting.

If the header parameter is omitted or blank, the files (subsequently specified) are transferred *as is*. That is, no header precedes the file. Any header information is assumed to be embedded in the file itself.

When you specify the binary file descriptor (DOSFIL), you do not need to completely specify the header. If the size is not specified or is zero, the entire file is uploaded, and the actual size is inserted into the header that is uploaded to the host. If the filename is not specified, the file name specified in the FILE parameter is inserted into the header that is uploaded to the host.
(Example: HEADER=<DBYPRD 00000x040>

RECTYPE=

RECTYPE and FORMAT upload files that contain variable-length records, in which the first character is each record identifies the type of record. RECTYPE and FORMAT parameters remain in effect only until the next HEADER parameter is encountered.

RECTYPE is a single, printable ASCII character; it indicates the record type to which the following FORMAT applies. *(Example: RECTYPE=A)*

FORMAT=

Defines the format of variable-length records whose record types match the preceding RECTYPE parameter. The format of a record consists of the data type and length of each field within the record, specified as follows:

EXAMPLE:

```
FORMAT=tnnntnnn. . .
```

where:

```
    t = data type
  nnn = field length, right-justified and zero-padded
```

Although the NORAND file transfer protocol supports a number of data types, many of these are specific to the NORAND PL/N language. For the sake of simplicity, specify a data type of X (for character) and the record length, which does not include the record type character. If you need more information on PL/N file formats, refer to the NORAND document, *Writing TCOM Modules in PL/N for the Hand-Held Computer*, or refer to the host TCOM manual.

EXAMPLE:

```
FORMAT=X020
FORMAT=N012X016N004
```

FILE=

The name of an upload file. Files are uploaded according to the preceding header information. Multiple FILE parameters may follow a single HEADER parameter, if the header applies to all the specified files. *(Example: FILE=BYPRD.DAT)*

Minimum NRUPLD.CTL

At a minimum, the NRUPLD.CTL file must specify a session control file, as described earlier.

```
HEADER=<DSCNTRL00001X042>
FILE=SCNTRL.DAT
```

The file SCNTRL.DAT would contain the session control data of the form:

```
PPPPPPPPPPPPPPTTTTTTTTTTTTTTTTTYYMMDDHHMMSS
```

as described earlier under “Session Control File.” This file could have any name. SCNTRL.DAT is used as an example.

EXAMPLE:

Example NRUPLD.CTL

```
HEADER=<DSCNTRl00001X042>
FILE=SCNTRL.DAT
HEADER=<DBYPRD 00000X040>
FILE=BYPRD.DAT
HEADER=<DBYTRXN00000X001>
RECTYPE=A
FORMAT=N004N005
RECTYPE=B
FORMAT=X010
RECTYPE=C
FORMAT=N004X005
FILE=BYTR00.DAT
FILE=BYTR01.DAT
file=bytr02.dat
FILE=BYTR03.DAT
HEADER=<DDOSFIL00001X010>FIMAGE.PCX
FILE=IMAGE.PCX
HEADER=<DDOSFIL00001X020>F\RT00001\COMMON.DAT
FILE=COMMON.DAT
HEADER=<DDOSFIL00001X000>
FILE=MYFILE.DAT
```

Communications Log File

A log of the communications is output to a text file named NRTLOG.DAT. The information contained in this file is described in the following subparagraphs.

BEGS=nnnn

Indicates the start of a TCOM session, where *nnnn* = session number. Currently, the session number is always 1. *(Example: BEGS=1)*

UP=d:\pathname\filename,nnn

Indicates a file upload was attempted, where *nnn* = error code. *(Example: UP=C:BYTRXN.DAT, 0)*

Valid values:

0 indicates a successful upload.

DOWN=d:\pathname\filename,nnn

This parameter indicates that a file download was attempted, where *nnn* = error code. *(Example: DOWN=C:\CUST.DAT, 23)*

Valid Values:

0 indicates a successful download.

ENDS=x,m,n

Indicates the end of a TCOM session, where:

```
x = Session status.  It is a single character code.
m = Stage of the communications session.
n = Protocol error code.
```

The following table shows the valid character codes for (x). This is the returned value from the application (PSROM0C.EXE).

Code for (x)	Meaning
'G'	Good session.
'T'	Unexpected end of transmission.
'H'	An incorrect file header was encountered.
'F'	A file error was encountered.
'L'	Telecom was aborted before the first file header was received.

The following table shows the valid values for (m).

Value for (m)	Meaning
5	Sign-on started.
4	Data send started.
3	Turn-around started.
2	Data receive started.
1	Sign-off started.

Protocol Errors

The following tables show valid Protocol error code for (n). This is a number that specifies the protocol error.

Table 5-1
TTY Protocol Errors

Error Code	Description
0	No error.
6	User aborted communications by pressing EXIT.
11	An invalid parameter was specified in the control file.
23	End of transmission.
101	Line lost.
102	Parity error.
103	Character gap too long.
104	Data loss.
105	Excessive NAKs.
106	Block count error.
107	Block check error.
108	Block framing error.
109	Control character error.
2xx	Modem error. xx is Hayes response code or program-defined code:
	03 No carrier.
	04 Command not recognized.
	06 No dial tone.
	07 Dialed number is busy.
	08 No answer.
	86 Error sending command to modem.
	87 Expected numeric response was not numeric.
	88 Invalid response format.
	89 No significant response from modem.
	97 System disabled COM port — low battery, PC Card modem removed.
	98 Unrecognized English response.
	99 Memory allocation error.

Table 5-2
NPCP Protocol Errors

Error #	Meaning
0	No error.
1	MININET.EXE is not installed.
6	User aborted communications by pressing EXIT.
11	An invalid parameter was specified in the control file.

The following errors are returned by MININET.EXE. 100 is added to the error returned by MININET.EXE to avoid conflict with other defined errors.

Table 5-3
MININET Protocol Errors

Error #	Meaning
101	Illegal buffer length
103	Invalid command
105	Command timed out
106	Message incomplete
108	Illegal local session number
109	No resource available
110	Session closed
111	Command canceled
113	Duplicate name in local name table
114	Name table is full
115	Name is deregistered, command completed
117	Local session table full
118	Session open rejected
119	Invalid name number
120	No answer
121	Name not found
122	Name in use on remote adapter
123	Name deleted
124	Session ended abnormally
125	Name conflict
126	Incompatible remote device
133	Network interface is busy
134	Too many commands outstanding
135	Invalid LAN adapter number
136	Command completed while cancel occurring
138	Command not valid to cancel
164–179	Unusual network condition
180–354	Adapter malfunction

Table 5-4
NRInet Protocol Errors

Error #	Meaning
0	No error.
1	PSROBOC.EXE could not be loaded. Ensure the directory for PSROB0C.EXE is in the PATH.
6	User aborted communications by pressing [NO].
11	An invalid parameter was specified in the control file.
800	TCP/IP kernel could not be loaded. Possible causes for this are: ♦ *(PSROM0C V2.XX only)* NET.CFG or PCTCP.INI does not exist. ♦ *(PSROM0C V2.XX only)* The PCTCP environment variable is not set. ♦ The PATH does not include the directories of the drivers. ♦ The computer does not have an Ethernet ID or the Ethernet ID could not be accessed. ♦ *(PSROM0C V3.XX only)* Invalid client IP address. Verify the IPCLIENT parameter, or make sure the DHCP or BOOTP server is running.
801	Invalid client IP address. *For PSROM0C V2.XX*, verify the entry for 'ip-address' in PCTCP.INI, or ensure the DHCP or BOOTP server is running. *For PSROM0C V3.XX*, verify the IPCLIENT parameter, or ensure the DHCP or BOOTP server is running.
802	*(PSROM0C V2.XX only)* Invalid Service or Service Type, or invalid port number. Verify the 'etc-dir' entry in PCTCP.INI specifies the correct path for the SERVICES file, and that an entry exists for 'nrinet.'
803	Invalid host name or IP Address. Ensure the host computer is running. *For PSROM0C V2.XX only,* verify the IPHOST parameter and the entries for 'subnet-mask,' 'router,' 'domain,' and 'domain-name-server' in PCTCP.INI. *For PSROM0C V3.XX*, verify the IPHOST, NETMASK, and ROUTER parameters.
804	*(PSROM0C V2.XX only)* Could not create socket. Check all cables and network connections.
806	Block sent was incomplete, or block received was incomplete.
807	Client/server negotiation failed.
808	Server specified an unsupported block size.
809	Invalid buffer pointer.
810	All server connections are already in use. Try again later.
811	Timeout while sending data. Connection to remote machine dropped. Ensure host is still running; check all cables and network connections.
812	Timeout while receiving data. Connection to remote machine dropped. Ensure host is still running; check all cables and network connections.
813	*(PSROM0C V2.XX only)* Attempt to send data to the server failed due to a closed connection. Ensure server is still running, and check all cables and network connections.
814	Attempt to receive data from the server failed due to a closed connection. Ensure server is still running, check all cables and network connections.
815	Could not access the network attach information. Ensure the Access Point, such as 6710, is connected and running.
816	An error occurred reading the network attach information. Ensure the Access Point, such as 6710, is connected and running.
817	Server did not respond to the connect request. Ensure the server is still running, and check all cables and network connections.
818	An error occurred reading the TCP/IP kernel information.
935	Client/server negotiation failed.
939	Destination address required.
940	Message too long.
948	Address already in use.

Table 5-4 (Continued)
NRInet Protocol Errors

Error #	Meaning
950	Network is down.
951	Network is unreachable.
952	Network dropped connection or reset.
954	Connection reset by peer.
955	No buffer space available.
960	Connection timed out.
961	Connection refused.
962	Too many levels of symbolic links.
963	File name is too long.
964	Host is down.
965	Host is unreachable.
966	Directory not empty.

Serial Communications

Option Connector

The option connector is used on the serial lid, the scanner lid, the MSR (Magnetic Stripe Card Reader) and the terminal. The option connector provides the mechanical and electrical interconnection for power and/or data communications between the 6100 Computer and the various lids.

Serial Lid Installation

The 6100 Serial Lid, with RJ-11, 9-pin RS-232 D-sub Serial Port, provides feedback to the user when it is correctly attached to the 6100 Computer.

With the 6100 Computer in “Suspend” mode (with no external charge attached and a properly charged and well operating Main Battery), attaching this specific serial lid causes the unit to “Power on” or wake up from “Suspend” mode. Thus, you can verify proper installation. If it does not “Power on” when the serial lid is attached, then one or both of the following problems may exist:

- The lid is attached improperly and the user must repeat the attachment process.
- The Main Battery is low or not operating properly.

The reason for this is because, as the serial lid is attached, a loop-back connection in the serial lid causes an interrupt to be sent to the microprocessor. This causes the 6100 Computer to “Power on” or wake up.

▶ **NOTE:** *This procedure works only when the unit is not receiving charge from an external power source.*

Serial Ports

The following serial ports are provided on the 6100 Computer:

- RS-232 or RS-485 through the 25-pin dock connector
- RS-232 through the serial endcap (or serial lid)

Refer to the *Reference, System Information* section, for port mappings.

IrDA Communications

For information on IrDA printing, refer to the *DOS IrDA Printing* paragraph, in the *Supporting DOS Applications* section of this publication.

6000 Series LAN Communications

Overview

MININET.EXE provides communications over Local Area Networks (LANs), using the following switches:

–s speed
–c comport
–t toggle

4000 Backwards Compatibility

Use these switches to disable the interrupt override from 4000API:

`/10 /14 /16 /PC /C3`

Device Support

6100 Display

The display is a Liquid Crystal Display (LCD) with a touch screen. The display is rotated to landscape mode to facilitate a wider screen. The touch screen accepts input from stylus or your finger.

Screen Rotation

Screen rotation is produced, in the DOS text mode, using VROTATE.EXE. For information on VROTATE, refer to the *4000 Series Screen Emulation* paragraph, in the *Conversions and Interfaces* section of this publication.

Screen rotation is produced, in the DOS graphics mode, using the Borland BGI driver, N6100.H and N6100.BGI.

Screen rotation is produced, in Windows, using the video driver 6100DISP.DRV.

Docks and Modems

Modem Device Driver: NORMOD.SYS

NORMOD.SYS is a PC Card card services client handling the identification and initialization of serial device PC Cards (e.g., modems, radios, etc.). It is loaded when you need to use a modem. This includes landline and wireless modems.

Command line switches are indicated by a preceding slash or dash and are case insensitive (e.g. –i and /I both indicate the same switch). Some switches are followed by data (text or numeric), in which case the data immediately follows the switch (no space in between). This is indicated by “T” for a text character, “H” for a hexidecimal digit, and “D” for a decimal digit. For example, a switch shown as NTTTTTTTT indicates that the “N” switch is followed by up to eight characters (e.g. –nMODEM1).

All switches are optional, and if omitted will default to NOT the switch or to the value indicated below.

NTTTTTTTT – device Name switch (defaults to MODEM)
S – save UART registers on suspend, restore on resume
IH – set IRQ to hex value H (defaults to 5)
CH – set to COM H (defaults to COM3)
BDDDD – fail device open if voltage less than DDDD millivolts
A – allow suspend when device open
DHHHH HHHH HHHH – is for cards that have an unreliable ready line. Use this format, if the manufacture's code and info field match the first two hexidecimal numbers respectively, delay for the third hexidecimal number of ticks (i.e., 55 ms per tick).
R – leave card on and enable RI during suspend (non functional on ELAN based terminals), since card power is off during suspend.

Charge Indicator

On all docks there is an indicator on the front of each dock or docking position that indicates the status of charge on the computer in the dock. This indicator does not turn on unless a computer is installed. The indicator states are:

- Amber Charge in progress, one hour rate
- Green Charge complete
- None No computer in dock or no power applied to dock

The charge indicator is driven by a small processor in the dock. When the computer is installed, it sends charge status information over a two wire interface (BCLK and BDAT) to the dock processor. The dock processor then turns on the indicator to show the user the state of the charge in the computer. The dock processor does not control charge, but is simply used to run the indicator.

RS-485 Connections

The 6100 Computers do not have any electronic RS-485 transceivers themselves. This level conversion is done in the dock. The normal COM1 signals RX and TX from the computer are sent through RS-485 transceivers in the dock. These drivers are enabled by tying the –485MUX line low in the external cable attached to the 25-pin D-sub connector on the back of the dock. This works on all 6100 single and vehicle docks. The RS-485 transceivers are always enabled on the 6100 multidocks.

Port Definitions

The data connectors on the back of the single and vehicle docks have been selected so that each external cable cannot be unintentionally crossed at time of install. See the *Reference, System Information* section for port definitions.

Dock	Data Connector	Computer Comm Port
6100 Single and Vehicle	25-pin D-sub male	COM1
6100 Single and Vehicle	9-pin D-sub female	COM3

Terminal to Dock Connector Pinouts

The following pinouts are for the 6100 docking connectors (the custom connector from the terminal to the dock itself.

6100 Docking Connector Pinout

When looking into the dock from the top, and from the indicator side of the dock, the pinout is right-to-left, pins 1, 3, 5, and 7 on the row toward rear of dock; then right-to-left, pins 2,4,6, 8 on the row toward front of dock. Below, is a depiction of the signals and functions for these pins.

Table 5-5
Docking Connector Pinout Descriptions (8-Pin)

Pin	Signal	Function
1	BCLK	Battery Clock
2	Dock Pwr 12 Vdc	Power to Dock from Terminal
3	GND	Ground
4	BDAT	Battery Data
5	TX	Transmit for Serial Port
6	RX	Receive for Serial Port
7	RTS	Ready to Send
8	CTS	Clear to Send

6100 Single/Vehicle Dock 25-Pin Female D-Sub Connector

With screw-in type retention posts (like a PC connector). For use with either RS-232 or RS-485 systems. This connector is COM1 on the 6100 Computer.

Pin	Signal	Function / Implementation
1	chassis ground	
2	TXD	
3	RXD	
4	RTS	
5	CTS	
6	DSR	Looped back internally to DCD and DTR
7	signal ground	
8	DCD	Looped back internally to DTR and DSR
9		No connect
10		No connect
11	RS-485+	
12–17		No connect
18	RS-485–	
19	–485Mux	
20	DTR	Looped back internally to DSR and DCD
22	RI	No connect
21, 23–25		No connect

6100 Single/Vehicle Dock 9-Pin D-Sub Female Connector

This connector is COM3 on the 6100 Computer, and is a two-wire RS-232 port. The 6100 IrDA output is converted to RS-232 levels for RX/TX in the dock.

Pin	Signal	Function / Implementation
1	DCD	Not connected
2	IRRX	Data input to dock
3	IRTX	Data output from dock
4	DTR	Permanently asserted in the dock
5	Ground	Ground
6	DSR	Controls baud rate on 9-pin port
7	RTS	Permanently asserted in the dock
8	CTS	Controls baud rate on 9-pin port
9	RI	Not connected in dock

Keyboard Definition and Redefinition

The following paragraphs present the definitions for the 6100 keys and specify the interface used for keyboard redefinition. Keep in mind that the 6100 Computer you are using may not be defined exactly as described below. The keyboard is almost totally redefinable, except for the “Gold” key (yellow shift), which returns key codes 15 or 79.

References

Refer to a book such as: *System BIOS for IBM PC’s, Compatibles, and EISA Computers, Second Edition* Phoenix Technologies LTD, for the scan codes, make codes, and break codes for a standard keyboard, or *Ralf Brown’s Interrupt List*.

Keyboard Definitions

Logical Keyboard

Alt Suspend/Resume	Contrast Up 7	Contrast Down 8	Backlight Off / On 9
Home Tab	Page Up 4	Up Arrow 5	Page Down 6
(period) Backspace	Left Arrow 1	Down Arrow 2	Right Arrow 3
“Gold” Key (Yellow Shift)	Control Escape	(minus sign) 0	End Enter

The logical keyboard is a view of the keyboard as represented to the user. The above chart shows an example of a Logical Keyboard layout, without overlays. The layout on your keyboard may (or may not) be the same. Depending on the unit, either the [I/0] or the [Suspend/Resume] key is in upper-left corner of the keyboard. On some units, press the [Gold] key before the [Suspend/Resume] key.

The darkly shaded area, in the above diagram, represents the “Gold” key (or the “Yellow Shift” key). The lightly shaded areas represent functions produced by shifting (holding down the “Gold” key), then pressing one of the other keys. Unshaded areas show the functions produced by pressing a key without shifting.

Physical Keyboard

The physical keyboard is defined as the mechanical 16-key keypad built into the 6100 Computer. Each key of the keyboard has a specific key number.

Keyboard Redefinition

All of the keys on the keyboard can be remapped, except for the suspend/resume key (upper-left) and the "Gold" key (lower-left). All of the keys on the keyboard can be shifted and unshifted, except for the "Gold" key.

The key numbers and scan codes for the keys on the 6100 Keyboard can be determined using the charts shown below.

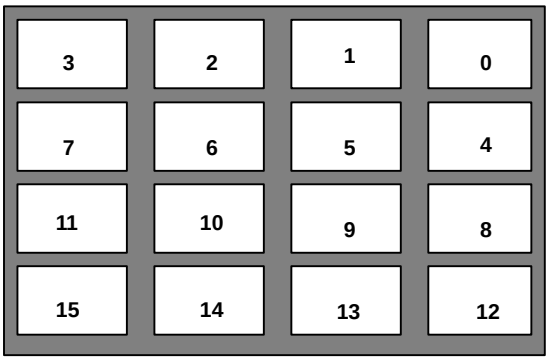
Unshifted Keys

Table 5-6 shows key numbers and associated scan codes for the unshifted keys.

Table 5-6
Unshifted Keys

Key Number	Keyboard Legend	Scan Code
0	9	0Ah
1	8	09h
2	7	08h
3	Suspend / Resume	70h
4	6	07h
5	5	06h
6	4	05h
7	Tab	0Fh
8	3	04h
9	2	03h
10	1	02h
11	Backspace	0Eh
12	Enter	1Ch
13	0	0Bh
14	Escape	01h

The following is a logical diagram of the keyboard, showing the relationship between the *Unshifted* key numbers and their physical locations on the keyboard.



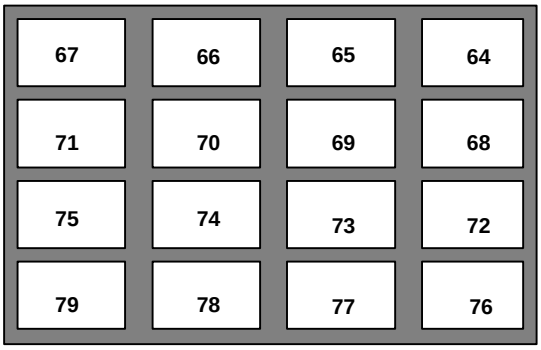
Yellow Shifted Keys

The following chart shows the key numbers and the associated scan codes for the *Yellow shifted* keys.

Table 5-7
Yellow Shifted Keys

Key Number	Keyboard Legend	Scan Code
64	Backlight; Off / On	6Dh
65	Contrast Down	6Eh
66	Contrast Up	6Fh
67	Alt	38h
68	Page Down	51h
69	Up Arrow	48h
70	Page Up	49h
71	Home	47h
72	Right Arrow	4Dh
73	Down Arrow	50h
74	Left Arrow	4Bh
75	(period)	34h
76	End	4Fh
77	(minus sign)	0Ch
78	Control	1Dh
	"Gold" key	

The following diagram is a logical representation of the keyboard, showing the relationship between the *Yellow Shifted* key numbers and their physical locations on the keyboard.



For additional information relating to using the scan codes in an application, refer to the *Standard Keyboard Interface* topic, in the *Conversions and Interfaces* section of this publication.

Remapping Keys for a Soft Reset

The 6100 Computers provides the capability for Software developers to add a soft reset action. Like the "CTL-ALT-DEL" for standard PC's, this soft reset can be a useful tool to avoid opening the Endcap when resetting the 6100 Computer.

▶ **NOTE:** *The Soft Reset function is not recommended for users booting off an SRAM card!*

The soft reset function is only recommended for a 6100 using SanDisk cards or booting from the RAM drive (C: drive).

Remapping the 6100 keypad for a soft reset can be done by assigning the [Ctrl], [Alt], and [Del] keycodes to any three keys on the 6100 keypad, using the KEYMAP.EXE utility. For example, if you wanted to use the bottom row of keys for the soft reset, then perform the following steps:

1. **Make a Copy of KEYS.INI**

   First, copy the file KEYS.INI from the 6100 Tool Kit, giving it a new name during the copy (such as: NEWKEYS.INI). This is the standard key definition file. (Refer to the sample listing of KEYS.INI, in *Appendix A, Sample Configuration Files*) . You can name this new file anything you want.

2. **Modifying NEWKEYS.INI**

   Substitute the following three definitions for the last three lines in the file. This changes the bottom three (shifted) keys to [Ctrl], [Alt], and [Del].

   - 76 = 0x53 ; Delete
   - 77 = 0x38 ; Alt
   - 78 = 0x1D ; Control

3. **Remapping the 6100 Keypad**

   Place this modified file on the 6100 system and call the remapping utility, KEYMAP.EXE, passing the name of the new key file as a parameter. Be sure to include the path if necessary.
   For example: `KEYMAP.EXE [\path\]NEWFILE.INI` where [\path\] is replaced with the actual path to the .INI file.

4. **Implementing the Soft Reset**

   Simply hold down the entire bottom row of keys (including the yellow shift key) on the 6100 keypad, and the system performs a soft reset.

Notice that after the modification, there may be a potential for duplicate key assignments (such as: [Alt] and [Ctrl]), or missing key assignments (such as: “End” and “Minus”). Assign (or reassign) keys, as needed.

Execute this remapping utility as many times as needed to remap on the fly; or place the command into your AUTOEXEC.BAT file to perform the remapping each time you reboot the 6100 Computer.

▶ **NOTE:** *Using “CTL-ALT-DEL” to reset is as robust as it is in conventional PC. If the 6100 processor is hung up, a soft reset may not bring the terminal back.*

This information and other helpful hints, are available in the 6100 Forum on the Customer Support BBS.

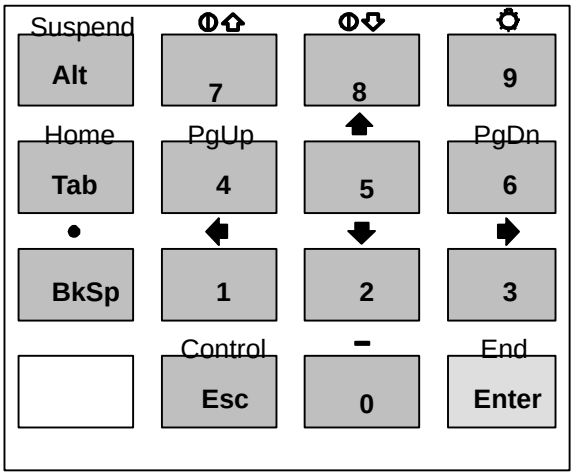
This method of remapping keys could also be useful for redefining other keys.

For another example of remapping keys, see *Keyboard Remapping with ANSI.SYS*, in *Appendix B, Common PEN*KEY 6000 Series Information*.

Keyboard Overlays

One example of a keyboard overlay is shown below. The text and symbols above the keys in the top row (Suspend, Contrast Up, Contrast Down, and Backlight On/Off) are the gold-plane values; the text and symbols on the keys are the keycap legends.

The overlay shown is not necessarily a standard. It is merely an example overlay. Any overlay can be made to fit the needs of any application or user.



6100 Memory

Overview

Sometimes, while installing programs, if you load some of them into the *High Memory Area (HMA)*, this frees some conventional memory for use by your application.

To better understand the use of these memory areas, refer to the description and the diagram in the *Memory Overview* paragraph of *Appendix B, Common PEN*KEY 6000 Series Information*.

Keep in mind that using the *HMA* could prevent you from creating the maximum RAM drive on the 6100 Computer. There is a relationship between RAM drives and Extended memory. A RAM drive uses Extended memory to store its information. So, when you create a one-megabyte RAM drive, you are using one megabyte of Extended memory. On a two-megabyte 6100 Computer, there is only one megabyte of Extended memory. Therefore, using up to 64K of HMA to load programs high takes away from the memory available for the RAM drive.

Using Expanded Memory on the 6100 Computer

The only reason you should use *Expanded memory* is if your application can access it. PenRight applications are good examples of programs that have the ability to use *Expanded memory.*

For information on *Expanded memory*, refer to the *Memory Overview* paragraph, in *Appendix B, Common PEN*KEY 6000 Series Information.*

The following information describes how to setup a CONFIG.SYS file so your applications can use extended memory.

EXAMPLE: CONFIG.SYS file

```
device=d:\himem.sys /machine:2
rem The following line is for configurations loading NORCS drivers:
device=d:\elanump.sys /x=C000,D400,D800,DC00,E000
device=emm386.exe x=A000-D3FF i=D400-E3FF frame=D400 x=E400-FFEF
DOS=UMB
   .
   .
   .
install=fixemm.exe
```

▶ **NOTE:** *If FIXEMM.EXE is not loaded, 6100 Computer locks up on Suspend.*

If using card and socket drivers, change the ELANUMP line to the following:
`device=d:\elanump.sys /x=C000,C400,D400,D800,DC00,E000`

If not using any card and socket drivers, change the ELANUMP line to the following: `device=d:\elanump.sys /x=D400,D800,DC00,E000`

Use ELANUMP to access the Upper Memory Area (UMA), because it knows what hardware it is running, and there could be problems with programs that are loaded high and access INT 15, if EMM386 is used for access to the UMA.

Upper Memory Provider: ELANUMP.SYS

ELANUMP.SYS is an *upper memory provide*r. It supports the minimal set of XMS function calls required to provide upper memory to DOS when you have a "`DOS=UMB`" statement in the CONFIG.SYS file.

The upper memory provider (UMP) supplies all upper memory block elements, in the range of C000–EFFF, on a PEN*KEY 6000 Series Computer. It can be used with HIMEM.SYS, or as a stand-alone program. The UMP does not support the XMS calls that deal with the deallocation or reallocation of upper memory blocks.

The UMP is similar to EMM386.EXE for loading devices into HMA. Unlike EMM386 with the NOEMS switch, the UMP supplies DOS with memory from the first megabyte of the system memory. This is memory that would otherwise be available for use as shadow RAM. EMM386 does not know how to program the memory controller to access this memory, so it uses some of the extended memory to emulate upper memory.

The UMP is smaller and simpler than EMM386. Since EMM386 is a virtual mode system extension, it also slows down processing as it handles special exception interrupts that are required when providing virtual memory to DOS. If you want upper memory, and do not need expanded memory, the UMP uses less system memory; therefore making more memory available to the system.

Option ROM blocks are scanned for and reserved (that is, left untouched). A single command line switch is provided to force the UMP to reserve memory block elements, other than those with option ROM signatures in them. The syntax for this switch is: `/X=aaaa[,aaaa]`

where the addresses specified are hexadecimal starting addresses for each block of memory to be reserved. The complete list of the blocks available in hardware is C800, CC00, D000, D400, D800, DC00, E000, and E800. Note that the last two blocks are 32KB in size instead of 16KB. This is a hardware constraint defined by the memory controller in the processor chipset.

The 6100 Computer provides upper memory (C800–CFFF, E000–EFFF) and HMA memory, using both HIMEM.SYS and ELANUMP.SYS:

```
DEVICE=HIMEM.SYS
DOS=HIGH
DEVICE=ELANUMP.SYS /X=C000, C400
DOS=UMB
DEVICEHIGH=...
```

The preceding configuration reserves the C000, C400 address range used by CardSoft to support PC Cards. You must reserve the address space required by CardSoft for it to function normally. If you have customized your CardSoft configuration, you may need to exclude a different set of addresses; check the MEM switch in your CSALLOC.INI file.

If using NORAND Card and Socket services, you only need to reserve C000 for that configuration, as follows: `device=d:\elanump.sys /X=C000`

Section 6

Conversions and Interfaces

Introduction

This section contains information applicable for converting 4000 Series applications for use on the PEN*KEY® 6100 Computers. The first topic deals with the subject of converting applications. This includes 4000 Series files and programs that are still supported, some that are no longer supported, some that have been changed, and a few new files. The remaining topics deal with interfaces and the applications that support them. These applications include 4000API.EXE, VROTATE.EXE, and ELANAPM.EXE. The interfaces include display services, disk services, serial communications, system services, multitasking, power management, keyboard, mouse, and others.

Only the major paragraphs containing interfaces are listed in the Contents (front of book) and the Topic Summary. The detailed interrupt list has been left out of those lists. Several methods for locating the interrupts and their definitions have been provided, as described below:

- The ***Topic Summary*** (on page 6-2) contains pointers to each of the major topics containing interfaces and converting 4000 Series applications for the 6100 Computer.
- The ***List of Tables*** (page 6-2) contains pointers to interface summaries and the master cross-reference for these interrupts.
- There is a ***Summary Table*** located at the top of each topic relating to applications that support interrupts. These tables are organized by interrupt number.
- The ***Interrupt Cross-Reference*** table is located at the beginning of the *Combined Interrupt List* paragraph, and is organized by interrupt number.
- The ***Interrupt Index***, (at the end of this publication) consists of indexes to all of the interrupt definitions within this section, and is organized alphabetically by interrupt name.

Refer to the *List of Tables,* below, to locate these interrupts.

The following is an outline of the section. The name of the supporting application is included in the first column, when applicable.

Topic Summary

The following tables provide additional assistance in locating interfaces:

List of Tables

Converting 4000 Series Applications

Introduction

When applications are converted from a 4000 Series System to a PEN*KEY 6100 System, the information in the following paragraphs should be considered:

Files No Longer Supported

The following 4000 Series files are not used in 6100 configurations:

File	Comments
10X16.EXT	Use VROTATE.EXE and FONTSEL.EXE (see *New 6000 Series Files*, page 6-4)
BOOT.SYS	
BOOTCFG.SYS	
BOOTPH0.SYS	
CATFILES.EXE	Use IPLFMT.EXE (see *New 6000 Series Files*, on page 6-4)
CATMAKE.BAT	Use IPLFMT.EXE
CATPREAM.BIN	
DEXIO.BIN	Use PC-DEXIO.BIN
FONTBUF.COM	
FPNOP.COM	
IO.SYS	ROM DOS 5 is loaded from flash
MAXI-DOS.SYS	ROM DOS 5 is loaded from flash
MEMIO.EXT	XLMEMIO is not supported
MINI-DOS.SYS	ROM DOS 5 is loaded from flash
MINI-IO.SYS	ROM DOS 5 is loaded from flash
MINI-NET.COM	Use MININET.EXE (see *New 6000 Series Files*, on page 6-4)
NOR-ANSI.SYS	
NORANDBB.EXE	
NP4805.EXT	The 4805 Endcap Printer is specific to 4000 Series systems
NPRTBIOS.EXT	Use PC4800.SYS (see *Files that Have Changed*, on page 6-4)
NT4800.SYS	Use PC4800.SYS (see *Files that Have Changed*, on page 6-4)
PRTBIOS.EXT	RCT printers (NP207, NP111) are not supported
RAMCARD.SYS	
RAMCFMT.EXE	
RAMCUTIL.EXE	
RAMDISK.EXT	A RAM drive is created as drive C: by the NORAND Utilities. The RAM drive on 4000 Series Computers is drive B:.
RPLHOST.EXE	Use INTERLNK.EXE

File (Continued)	Comments
SCAN4000.EXE	
SETDISP.EXE	Use VROTATE.EXE and FONTSEL.EXE (see *New 6000 Series Files*, on page 6-4)
SOFTBIOS.EXT	BIOS is loaded from flash.
TNETBIOS.EXE	
XLMEMIO.BIN	XLMEMIO is not supported.

Files that Have Changed

The following 4000 Series Files are used differently on the 6100 Computer. You can make the changes as specified.

CONFIG.SYS

The extended CONFIG.SYS commands in the 4000 Series version of IO.SYS are no longer supported:

Command	Comments
EXT=	BIOS is loaded from flash. Remove these lines from CONFIG.SYS.
TSR=	Use INSTALL= or execute TSRs from ROMINIT.BAT or AUTOEXEC.BAT.
VERIFY	The VERIFY ON command, a feature of COMMAND.COM, can be used. However, this command is not identical with VERIFY. VERIFY causes MINI-DOS/MAXI-DOS to write through the disk buffers and to write directory information to the media whenever it changes. These actions do not occur in DOS. The application is responsible for performing file commits. Refer to the *PL/N Application Changes* topic, on page 6-7.

Refer to the sample CONFIG.SYS listing in *Appendix A, Sample Configuration Files*, for an example of the CONFIG.SYS, as used in an application for the 6100 Computer. Also, refer to the *ROM DOS 5* paragraph, in the *Reference, System Information* section, for additional information.

CPLNI.COM

Do not use the *-d* command line switch. Do not delete driver files from the RAM drive after they are loaded. All files needed to restart the application after a reset must remain on the RAM drive.

PC4800.SYS

Use Version 1.80 or later for a PEN*KEY 6000 Series platform. This driver is used for NORAND Portable Communications Protocol (NPCP) printer support for both PL/N and non-PL/N applications. 4000API.EXE must be loaded when PC4800.SYS is used.

PC-DEXIO.BIN

Use Version 1.26 or later for a PEN*KEY 6000 Series configuration. 4000API.EXE must be loaded when PC-DEXIO.BIN is used.

New 6000 Series Files

The following files are new 6000 Series files that were not used on the 4000 Series platform:

AUTOEXEC.BAT

This is a standard ROM DOS 5 configuration file. Refer to the *ROM DOS 5* topic, in the *Reference, System Information* section, for a description of this file.

4000API.EXE

This program provides functionality for certain applications written for the 4000 Series to run unmodified on the PEN*KEY 6000 Series platform.

VROTATE.EXE and FONTSEL.EXE

Use VROTATE.EXE for rotating the display screen. Use FONTSEL.EXE to change the screen font. These programs also provide functionality expected by applications written for the 4000 Series so that these applications can run unmodified on the PEN*KEY 6000 Series Computer. Refer to the *4000 Series Screen Emulation* topic, on page 6-24, for additional information.

*.FNT

Font files for use with FONTSEL.EXE.

IPLFMT.EXE

This program concatenates a list of files into a single file that is suitable for download in a Norand communications session.

MININET.EXE

This program replaces the 4000 Series MINI-NET.COM on the PEN*KEY 6000 Series platforms. It provides the NET BIOS interface to the NPCP LAN.

Unchanged Files

The following 6000 Series files have not changed from the 4000 Series.

File	Comments
CLKIO.BIN	
DELETE.COM	
HOSTIO.BIN	
INT15.EXE	For a PC only. Do not run on a PEN*KEY 6000 Series platform.
KBDIO.BIN	
LZEXE.DOC	
LZEXE.EXE	
MEMIO.BIN	
MV.EXE	
NORSESS.COM	
PRN2COM.COM	
PRTIO.BIN	
TTYIO.BIN	
XYXFER.COM	

C++ Application Changes

Keyboard

Application programmers should consider making use of keys that are available on the 6000 Series Computers, but not available on the 4000 Series, such as PageUp, PageDown, Home, and End.

Display

The current font files used with PEN*KEY 6000 Series Computers generate different display sizes than the 4000 Series supports. Application screen changes should be considered to better utilize the new sizes.

Files

Since each PEN*KEY 6000 Series Computer has a reset button, there is a possibility that the user could press it at any time. To maintain data integrity the application must make sure that all file writes are committed to disk. This should be done at the completion of every transaction, e.g. a sales invoice. Applications should `fflush()` all open streams. Also, because of DOS disk buffers and the fact that DOS does not update the directory entry for a file with each write, applications must do one of the following:

- Close and reopen all files.
- Call DOS function 68h for each open file. All data still in DOS disk buffers is written to disk immediately, and the file's directory entry is updated. Example code for Borland C++:

```
asm mov ah,0x68
asm mov bx,filehandle
asm int 0x21
```

- Use DOS function 5D01h, which flushes DOS buffers and updates the directory entries for all open files. Example code for Borland C++:

```
int FileCommit(void)
{
REGS  regs;
SREGS sregs;
// Parameter table used by function 5d01h.  Only process_id and
// computer_id are used.
struct {
unsigned int ax;
unsigned int bx;
unsigned int cx;
unsigned int dx;
unsigned int si;
unsigned int di;
unsigned int ds;
unsigned int es;
unsigned int RESERVED;
unsigned int computer_id;
unsigned int process_id;
} dos_parm;
   dos_parm.computer_id = 0;          //set current computer
   dos_parm.process_id = getpsp();//set current process
   sregs.ds = FP_SEG(&dos_parm);      //set up address of parameter table
   regs.x.dx = FP_OFF(&dos_parm);
   regs.x.ax = 0x5d01;                //commit all files to disk
   intdosx(&regs,&regs,&sregs);
   if (regs.x.cflag)
return regs.x.ax;                     // if error return code
return 0;                             // completed ok
}
```

Even if files are committed to disk, a reset can still occur at such a time as to leave a partial record in a file. Files such as the transaction stream should be verified at program start, and any invalid records should be truncated or marked as void.

Remove any code used for formatting memory cards for the 4000 Series. PC memory cards can be formatted on the PEN*KEY 6000 Series Computer by loading the CardSoft software and executing FORMAT.COM.

Drives

The RAM drive is now drive C:, not drive B:.

If CardSoft software is used, use only the CardSoft drives (E: and F:) to access the PC memory cards. Using drive A: or B: can corrupt the cards.

Applications that use drive tables may need to change the file maintenance done after communications. Applications which attempt to move files from one drive to another, to conform to the drive table, should be changed to ignore drive D:, which is the Flash, a read-only drive.

Printers

The 4805 Endcap Printer is not supported by the PEN*KEY 6000 Series Computers. Remove any code that is specific to the 4805 Printer.

Communications

References to COM3 must be changed. On the 4000 Series, COM3 refers to the same port as COM1, but ignores the modem control signals RTS and CTS. INT 14h function 5, provided by 4000API.EXE, can ignore modem control signals on COM1 when needed.

Remove any reference to internal modems for the 4000 Series. This includes turning on/off switch V+ (Norlib function NRSwitchVplus).

PEN*KEY 6000 Series applications can utilize PC modems.

Reset

It is important to remember that all variables lose their values when the 6100 Computer is reset. Any information needed to recover from a reset must be written to a file so that it can be retrieved.

Memory

Verify that the memory requirements of the application can be met. On the 4000 Series, it is possible to achieve executable memory sizes of greater than 640K. This is not true on the PEN*KEY 6000 Series, which uses ROM DOS 5. Because of this, and due to differing system software requirements, some applications could encounter problems of insufficient memory.

Power Management

For maximum battery life, power to system components should be carefully managed. Drivers provided in the 6100 Tool Kit, such as PC4800.SYS and MININET.EXE, turn power on as needed, but they restore power to its previous state when finished. Therefore, applications should initially turn power off to certain components. For example: COM ports, the LAN adapter, and the touch screen. NORDOSPM.EXE provides this capability, or the Norlib function `NRApmSetPower` can be used within an application. Refer to the *Power Management* section of this publication.

Norlib

Use Norlib V2.00 or greater (or prerelease V1.20) for the PEN*KEY 6000 Series hand-held computers. *NRCursorSet* does not support a blinking cursor on PEN*KEY 6000 Series Computer.

PL/N Application Changes

In the interest of maintaining a single set of source code for applications that require changes that are different for each platform, these changes should be implemented using the *"++ifdef"* conditional compile facility of the PL/N compiler preprocessor. By convention, use *"++ifdef 6000"* to encase code that should execute only on a PEN*KEY 6000 Series system. For further information on the *"++ifdef"* facility, refer to the *PL/N 4000 Series Reference Manual, Volume 1*.

General Source Changes

In the HISTORY section, include explicit comments on changes and modules that were changed.

KBDIO

The current font files used with FONTMAP.EXE generate different display sizes than those fonts supported on the 4000 Series. Applications that use `GETCTL 3 (KB_PHYSIZE)` need to do the following:

- Include code to handle the new display size generated by FONTSEL.EXE. Applications that support multiple display sizes most commonly check the display size in menu functions or other places where a larger screen is used.

MEMIO

Because the PEN*KEY 6000 Series Computer has a reset switch, to maintain data integrity, the application must make sure that all file writes are committed to disk. Because of DOS disk buffers and the fact that DOS does not update the directory entry for a file with each write, a new standard routine, IPFCMT6, is provided. This routine flushes DOS buffers and updates the directory entries for all open files. Applications should call this routine at the completion of every transaction; for example, a sales invoice. IPFCMT6 is described under *New Standard Routines* paragraph, on page 6-10.

All files that are written by OUT instead of PUT require additional measures to properly commit them to disk. The use of OUT causes MEMIO to do buffering of its own. To cause MEMIO to flush its buffers, close and reopen these files. Alternatively, replace all OUT statements with calls to the standard routine PZUFT1P, which ensures that partial buffers are written correctly.

PRTIO

The 4805 Endcap Printer is not supported on the PEN*KEY 6000 Series platform. Remove references to PUTCTL 1047 (PR_4805) as well as any other code that is specific to the 4805 Printer.

RCT printers, e.g. NP207 and NP111, are not supported on the PEN*KEY 6000 Series platform. Remove references to PUTCTL 1044 (PR_NONNPCP) as well as any other code that is specific to these printers.

Set PRT.LAST = 256 or 0 at the start of a report. Many applications set PRT.LAST = 256 when a formfeed is done to reset the line counter. This causes problems on the PEN*KEY 6000 Series platform. Instead, set PRT.LAST = 257 after the first line of the page printed.

The standard routine PGWTE1P should be used at the end of each report to wait for the printer completion, or printer errors might be missed.

The standard routines PGAPR2P/PGPER2P should be used. Some applications may still be using the older routines PGAPR1P/PGPER1P.

SYSIO

Remove references to PUTCTL 1026 (SY_POWER_ON) and PUTCTL 1027 (SY_OFF_POWER). These were used on a 4000 Series Platform to enable the internal modem. Remove any references to the internal modems for the 4000 Series Platform.

XLMEMIO

XLMEMIO is not supported on the PEN*KEY 6000 Series platform. Applications must be converted to be DOS-compatible.

Adding 6805 Printer Support to PL/N Applications

These steps explain how to add 6805 Printer support to a PL/N application.

1. **Install the IrDA Drivers.**

   For installation instructions for these device drivers, refer to the *DOS IrDA Printing* paragraph, in the *Supporting DOS Applications* section, or the Windows *IrDA Printing* paragraph, in the *Supporting Windows Applications* section.

EXAMPLE: 6100 CONFIG.SYS file:

```
buffers=50
files=99
stacks=64,256
DEVICE=D:\ELANUMP.SYS /X=C000,C400
DOS=HIGH,UMB
DEVICEHIGH=D:\ELANAPM.EXE
DEVICEHIGH=D:\NORDOSPM.EXE
DEVICEHIGH=D:\CLOCK.EXE
DEVICEHIGH=D:\PC4800.SYS LPT1 1 /I1
DEVICEHIGH=D:\PRDRV.SYS NP6805$
```

EXAMPLE: 100 AUTOEXEC.BAT file:

```
@echo off
D:\4000API.EXE /10 /16 /PC /C3
LH D:\MININET.EXE –c1 –s1152 –t0
LH D:\IRDAPDRV.EXE –fNP6805$ –b19200 –r25 –d8 –t6100
LH D:\61MOUSE.COM
C:\CALIB.EXE
C:\VROTATE.EXE 12 20 159 240 –PLN
LH D:\FONTSEL.EXE 1
C:\PSEKM0C.EXE KBDFILE.DAT
LH D:\DOSGAS.EXE 25 1 –u5000 –C
LH D:\ELANCFG.EXE /H12 /L1 /D4 /V0 /T2 /R1 /C0
LH D:\BKSP2DEL.COM
DPLNI.COM PBRRR0P.PL6 CONFIG.PLN –p –w2
```

2. **Use PC-PRTIO.BIN instead of PRTIO.BIN**

   A new PL/N driver named PC-PRTIO. BIN supports the use of the IrDA drivers. Use this instead of PRTIO.BIN. This driver can be found in PLN4000 V1.07 or greater.

▶ **NOTE:** *As of this writing, PC-PRTIO.BIN does not support printing to 4810/4815/4820 Printers and PRTIO.BIN does not support the 6805 Printer. Include only one of these drivers in a configuration, not both. Work is in progress to support all printers in a single PL/N driver.*

3. **Change the Printer Open Procedure**

   A new putctl is added to PRTIO which selects the 6805 Printer. As of this time, no constant is defined in PCPRCNP.

EXAMPLE:

```
OPEN (PRT, PRTBUF) PR_NAME
IF CONFIG.PRINTER = PRT_6805 THEN
   PUTCTL (PRT) 1049             ;for 6805 IR printer
ELSE
   PUTCTL (PRT) PR_NPCPROT       ;for 4810/4815/4820 printer
ENDIF
IF CONFIG.PRINTER = PRT_4815 THEN PAGELEN = 66
```

▶ **NOTE:** *As of this publication, PC-PRTIO.BIN does not support printing to 4810/4815/4820 Printers and PRTIO.BIN does not support the 6805 Printer. If putctl PR_NPCPROT is used with PC-PRTIO.BIN, or putctl 1049 is used with PRTIO.BIN, the application locks up in the printer error procedure. Work is in progress to support all printers in a single PL/N driver.*

4. **Open and close the driver for each report.**
   In most PL/N applications, the printer is opened at the start of a program and closed only when chaining to another program. IrDA drivers required that the printer be opened just before printing a report and closed just after finishing the report. For applications which currently have a common PRINTCTL routine, often named PAxxP9P, this is accomplished by moving the open/close logic to this routine. Remove all other opens and closes.

   This change is compatible across all NORAND hand-held computers, e.g. 4000 Series, so there is no need to include a ++IFDEF.

5. **Remove Excess Linefeeds.**
   The report formats used for the 6805 are the same as previous 40-column printers. However, the 6805 requires fewer linefeeds to sufficiently eject a report. Many applications feed eight lines at the end of a report on a 40-column printer. Reduce this to five lines for a 6805 Printer. Also, reduce the number of linefeeds used prior to “TICKET OK?” prompts. This saves paper and makes your reports look nicer.

Unsupported Standard Routines

IPFMCF6, the memory card format routine for the 4000 Series. Memory on PC Cards cards can be formatted on the PEN*KEY 6000 Series Computer by loading the CardSoft drivers (or NORMOD drivers) and executing the FORMAT.COM program from ROM DOS 5.

New Standard Routine Numeric Function IPFCMT6

This function commits all outstanding writes on all open files and updates the DOS directory. It uses DOS function 5D01h. This function can be used only on a PEN*KEY 6000 Series platform. On a 4000 Series, using MINI/MAXI-DOS, use of this function causes the system to display the message “Unimplemented DOS call” and hang.

Parameters:
None

Return Value:
Zero (0) if successful. A return value of 257 indicates that the calling routine incorrectly attempted to pass parameters to this function. Any other return value is the return value of DOS function 5D01h.

EXAMPLE:

```
EXTERNALS
++ifdef 6000
   NUMERIC FUNCTION  FILECOMMIT = IPFCMT6
++endif
LOCAL VARIABLES
   01 RESULT       BINARY WORD
. . .
++ifdef 6000
;Commit files to disk when transaction is complete
PERFORM FILE_COMMIT
++endif
. . .
++ifdef 6000
PROCEDURE FILE_COMMIT
   RESULT = FILECOMMIT
   IF RESULT THEN
      OUT (KBD) (KB_HOME)
      &  'FILE COMMIT',
      &  'FAILED!',
      &  'DOS ERROR #', RESULT,
      &  'DISK MAY BE',
      &  'CORRUPT.  NOTIFY',
      &  'SUPERVISOR', BEL, BEL
      ;Do not allow user to continue
      REPEAT
         RESULT = INPUT(E)
      UNTIL 0
   ENDIF
END; OF FILE_COMMIT
++endif
```

Power Management BIOS Interfaces: ELANAPM.EXE

Overview

This paragraph describes the register interfaces to the APM BIOS functions. Functions are accessed through INT 15h. The carry flag is set and an error code is placed in the AH register when an error condition is detected. The carry flag is reset to zero upon return from successful calls. The contents of the AH register depend on the particular call.

This interface has a convention for identifying a device class and units within that class to allow for direct control of devices. For example, all disk devices are a class and the units are the physical unit numbers. The device ID parameter is passed in a word-length register (BX), where BH is the device class and BL is the device unit.

Power Device IDs

The APM Power Device Class/Subclass IDs are defined as follows:

Value in Register BX	**Power Device ID** *(General Definitions)*
XXh	Unit number (Zero based)
FFh	All devices in this class

ID (Hex)	**Description**
00XX	System
00	APM BIOS
01	All devices power-managed by the APM BIOS
0100	Display
0200	Secondary storage (PC Card)
03XX	Parallel ports
0400	RS-232 serial port
0410	RS-485 serial port
0500	Reserved for Ethernet network adapters
0600	PC Card socket A
0601	PC Card socket B
0700–DFFF	(Reserved)
E000–EFFF	OEM-defined power device IDs
E000	POD1
E100	Digitizer
E200	Backlight
F000–FFFF	(Reserved)

APM Function Summary

The following is a summary of the interrupt 15h functions that are available for Advanced Power Management (APM) in the 6100 Computer.

Table 6-1
APM Interrupt Summary

INT	**DOS APM Interrupts (INT 15h)**	**Function AL=**	**Requires Connection**
15h	APM Installation Check	00h	No
15h	APM Real-Mode Interface Connect	01h	No
15h	APM Interface Disconnect	04h	Yes
15h	CPU Idle	05h	No
15h	CPU Busy	05h	No
15h	Set Power State	07h	No
15h	Enable/Disable Power Management	08h	No
15h	Get Power Status	0Ah	No
15h	Get PM Event	0Bh	No
15h	Get Power State	0Ch	No
15h	Enable/Disable Device Power Management	0Dh	No

APM CPU Idle Interrupt

CPU Idle, INT 15h, AX=5305h, is an APM BIOS function. Refer to the CPU Idle definition, on page 6-78, for details on usage.

A call to this function causes the processor to halt (via the HLT instruction) until the next hardware interrupt. This causes good power savings, if it is used during input polling loops, or other low activity points in an application.

Refer to the Intel APM BIOS Interface specification 1.1 or Ralf Brown's Interrupt List for more information.

Refer to *Appendix B, Common PEN*KEY 6000 Series Information,* for a sample listing (IDLE.CPP) showing the use of the CPU Idle interrupt.

6100 BIOS Interfaces

Overview

This is the core BIOS supported for the 6100 Computer. Starting with flash version 1.11, the following features are added to the core BIOS:

- Suspend/resume coordination (suspend/resume key mapped to I/O key, if IO2SUS.COM is loaded):
  - to work with NORAND Card and Socket services
  - and for radio integration
- COM4 timeout for the BIOS data area

The detailed definitions for the PC BIOS interfaces supported for the 6100 Computer are described in the paragraph. The following paragraphs summarize the supported interrupts, as well as the unsupported interrupts.

Only some of the 4000 Series BIOS interrupts are supported by the 6100 Computer, as described in the following paragraphs:

Supported BIOS Interfaces

There are two indexes provided for the details of the BIOS interface functions supported for the 6100 Computer. To find an interrupt by its name, refer to the *Interrupt Index,* at the end of this publication. To locate an interrupt by the interrupt number, refer to the *Interrupt Cross-Reference* table, on page 6-28.

System Timer Interface: Interrupt 08h

The system provides a periodic system timer tick and issues interrupt 8 at every timer tick. The interrupt occurs approximately every 55 milliseconds. This rate is close to 65535 ticks per hour.

The interrupt service routine maintains the count of interrupts from power-on at BIOS data location 0x40:0x6C. After 24 hours of operation, location 0x40:0x7C is nonzero when the counter overflows to the next day. The number of ticks in 24 hours is 0x1800B0.

The system timer interrupt issues a call to a user-supplied routine through interrupt 0x1C at every interrupt.

Application programs should avoid using the Timer Tick as much as possible, since many applications are already hooked into it and it could become overloaded.

Standard Keyboard Interface: Interrupt 09h

This interface is for both the internal keypad and an external keyboard.

Matrix scan codes are converted to PC-compatible scan codes before Interrupt 09h is issued. Standard scan codes are read from port 60h. If the scan code is not a request for an internal function or a control key press, both the character code and scan code (two bytes) are placed in a 16-word circular buffer at the position pointed to by the keyboard tail buffer pointer. The tail pointer is then incremented by two and allowed to wrap around the buffer.

The entire keyboard interface is described later in this publication in the *Standard Keyboard Interface* paragraph, on page 6-35.

Display Services: Interrupt 10h

A standard VGA BIOS is provided. The software interface to the display hardware is through function calls to the BIOS interrupt 10h. The function number is placed in the AH register. Other registers are loaded with the values required by the individual function and then the call to the interrupt is made. Each individual function is described in detail in the following pages.

Equipment Determination: Interrupt 11h

This function returns the equipment information stored in the word located at 40:10h in the BIOS data area. The organization of the word is defined as follows:

Bits	Description
15–14	Number of printer channels
13	Internal modem installed
12	joystick installed
11–9	Number of RS-232 channels
8	(Not used)
7–6	Number of diskette drives (if bit 0 = 1) 0 = One drive 1 = Two drives
5–4	Initial video mode 0 = Not used 1 = 40x25 color 2 = 80x25 color 3 = 80x25 B/W
3	(Not used)
2	Pointing device installed
1	Math coprocessor installed
0	Diskette available for boot

On Entry:
None

On Return:
AX = Equipment information

Memory Size Determination: Interrupt 12h

This software interrupt returns the contents of location 40:13h in the BIOS data area in the AX register. Register AX contains the amount of RAM available to the operating system – up to 640K – as determined by the POST. The value returned in register AX indicates the number of 1KB blocks available and does not include the interrupt vector memory, BIOS data memory, or extended BIOS data memory areas.

Certain memory-size information is stored in CMOS RAM located in the real-time clock. The organization of this data is as follows:

Offset	Size	Description
15h	byte	Base memory in 1KB multiples, low byte
16h	byte	Base Memory in 1KB multiples, high byte
17h	byte	Extended Memory in 1KB multiples, low byte
18h	byte	Extended Memory in 1KB multiples, high byte

Disk Services: Interrupt 13h

Disk Services perform I/O to RAM, flash, and PC Card devices. The partitioning of RAM and flash for mass storage and the assignment of drive letters are determined at system configuration. The following is a list of the Error Codes:

Code (Hex)	Description	Code (Hex)	Description
00	No error	0D	Invalid number of sectors on format
01	Invalid function or parameter	0E	Control data address mark detected
03	Write-protect error	0F	DMA arbitration level out of range
04	Sector not found	10	Uncorrectable ECC or CRC error
05	Reset failed	20	General controller failure
06	Diskette change line active	40	Seek operation failed
07	Drive parameter activity failed	80	Timeout
08	DMA overrun operation	AA	Drive not ready or not selected
09	Data boundary error	BB	Undefined error occurred
0A	Bad-sector flag detected	CC	Write fault on selected drive
0B	Bad cylinder detected	E0	Status error/error register = 0
0C	Media type not found	FF	Sense operation failed

Serial Communications Services: Interrupt 14h

The transmit and receive functions use a timeout value from a table at location 40:7C. The default timeout value is 1, which represents about 1 second.

The extended initialize function allows you to set baud rates beyond those available on a PC. Since these functions are polled (not interrupt-driven), the faster baud rates are of little utility.

System Services: Interrupt 15h

These services consist of the following topics:

- Keyboard intercept
- Device access
- Program termination
- Event wait interval
- System request key
- Moving memory and reading memory size
- Switching to protected mode
- Indicating interrupt complete
- Returning system configuration parameters
- Returning extended BIOS data area segment

Keyboard Services: Interrupt 16h

Access to the keyboard BIOS services is through INT 16h. The AH register is loaded with the function number to be performed. Other registers are loaded as required by the individual functions. Values are returned in individual registers as defined by the functions.

System Reboot: Interrupt 19h

Application programs can issue Interrupt 19h to force the computer to reboot. The computer is forced through the initial power-up sequence.

NOTE: *Do not use INT 19h to reboot, as it causes a system failure on the 6100 Computer.*

This computer uses ROM DOS and does not reload disk DOS when an interrupt 19h occurs. Refer to the *Interrupt Cross-Reference* table, page 6-28, for INT 15h, System Reset, function AX = 5380h, subfunction BH = 1Dh.

Real-Time Clock: Interrupt 70h

This interrupt handler controls the periodic and alarm interrupt functions from the RTC. The periodic function, when activated, occurs 1024 times per second. The Dword counter is decremented by 976 at each interrupt. When the location becomes equal to or less than zero, bit seven of the designated location is set. The alarm function of the RTC provides an interrupt at the specified time and software Interrupt 4Ah is issued.

Locating 6100 BIOS Interrupts

The interrupts supported for the 6100 Computer are listed in the *Interrupt Cross-Reference* tale, located under the *Interrupt Cross-Ref (by number)* tab. Even though they are matrixed into the entire set of interrupts, each individual interrupt supported by the 6100 BIOS can be identified by the notation, **BIOS**, immediately under the interrupt heading. The categories of interrupt services are listed in the center column of Table 6-2.

Individual interrupt definitions can be located:

- Looking up almost any noun or verb relating to a specific interrupt in the *Interrupt Index*, with the exception of *Read, Set,* and *Get*.
- Using the *Interrupt Cross-Reference* table; or
- Thumbing through the definitions, searching for the notation, *BIOS*.

Table 6-2
6100 BIOS Interrupt Summary

INT#	Supported Interfaces
08h	System Timer Interrupt
09h	Keyboard Interface
10h	Display Services
11h	Equipment Determination
12h	Memory Size Determination
13h	Disk Services
14h	Serial Communications Services
15h	System Services
16h	Keyboard Services
19h	System Reboot
1Ah	Timer and Real-Time Clock Services
70h	Real-Time Clock Interrupt

NORAND Proprietary System Interfaces

There are some proprietary system interface extensions reserved for use by Intermec Technologies Corporation. These are in the INT 15, AX=5380, series.

Unsupported PC, 4000 Series BIOS Functions

Nonmaskable Interrupt (NMI) 02h

This interrupt is not used by the 6100 BIOS.

Print Screen Interrupt 05h

The Print Screen interrupt is not supported by the 6100 BIOS.

4000 Series Video BIOS Functions: Interrupts 12h & 14h

4000 Series video functions are not available in the BIOS. This means that the 4000 display processing features listed below are not available. The 6100 screen rotation and font mapping utilities are provided to emulate the 4000 Series video functions for porting applications from the 4000 Series to the computer.

- Cursor Fixed and Cursor Chase modes are not supported by the VGA BIOS.
- NorLib and PL/N cursor on and off support are not supported.
- Inverse-mode character mapping is not supported (characters above 127 are mapped to inverse video characters 0–127 by turning off the high bit).

Also, the following 4000 Functions are not supported by the 6100 BIOS:

- Function AH = 12h, Subfunction BL = 0FAh: Set Physical Display Size
- Function AH = 12h, Subfunction BL = 0FBh: Return Pointer to Current Display Parameters
- Function AH = 12h, Subfunction BL = 0FCh, AL = 0DDh: Return Invert Mode Status
- Function AH = 12h, Subfunction BL = 0FCh, AL = 0DEh: Enable Invert Mode

- Function AH = 12h, Subfunction BL = 0FCh, AL = 0DFh: Disable Invert Mode
- Function AH = 12h, Subfunction BL = 0FCh, AL = 0E0h: Set Chase Mode
- Function AH = 12h, Subfunction BL = 0FCh, AL = 0E1h: Set Fixed Mode
- Function AH = 12h, Subfunction BL = 0FDh: Return Physical Display Size
- Function AH = 12h, Subfunction BL = 0FEh: Read/Write Contrast (supported by 4000API.EXE)
- Function AH = 12h, Subfunction BL = FFh: Backlight Off/On (supported by 4000API.EXE)
- Function AH = 14h, Subfunction AL = 00h: Load User Font
- Function AH = 14h, Subfunction AL = 01h Load System Default Font

Support for porting 4000 Series applications to the 6100 Computer is provided by VROTATE.EXE and FONTSEL.EXE for most of the above services. For detailed information on supported video functions, refer to *4000 Series Screen Emulation*, starting on page 6-24, or the *Display Services* Interrupt definition, page 6-36.

4000 Series Disk BIOS Services: Interrupt 13h

These 4000 BIOS Interrupt 13h functions are not supported by the 6100 BIOS:

- Function 05h: Format Disk (reserved)
- Function 0Ah: Read Long Sectors (reserved for diagnostics)
- Function 0Bh: Write Long Sectors (reserved for diagnostics)

Refer to *Disk Services* Interrupt, starting on page 6-36 in this section, for information on supported disk services.

4000 Series Port Control BIOS Services: Interrupt 14h

These 4000 BIOS Interrupt 14h functions are unsupported by the 6100 BIOS:

- Function AH = 05h: Extended Port Control
- Function AH = 0FFh: Port Power On/Off

Support is provided via 4000API.EXE for porting 4000 Series applications to the 6100 Computer. Refer to the *Serial Communications Services* Interrupt, starting on page 6-24 in this section, for supported serial communications services.

4000 Series Multitasking BIOS Services: Interrupt 15h

The 4000 Series BIOS multitasking services, informally referred to as INT 15, are **not** provided by the 6100 BIOS. This includes the following list of functions:

- Function AX = 0100h: Create a task
- Function AX = 0101h: Delete a task
- Function AX = 0102h: Return information about a task
- Function AX = 0103h: Disable Task Switching
- Function AX = 0104h: Enable Task Switching
- Function AX = 0105h: Reset Multitasker
- Function AX = 0106h: Pend on Mailbox with Optional Timeout
- Function AX = 0107h: Accept Message from Mailbox (no pend)
- Function AX = 0108h: Post Message to Mailbox
- Function AX = 0109h: Pend on Queue with Optional Timeout
- Function AX = 010Ah: Accept Message from Queue (no pend)
- Function AX = 010Bh: Post Message to Queue

- Function AX = 010Ch: Delay Current Task
- Function AX = 010Dh: Initialize a Queue Structure as Empty
- Function AX = 010Eh: Enable Time Slicing
- Function AX = 010Fh: Disable Time Slicing
- Function AX = 0110h: Set Task Identifier

4000 Series Printer BIOS Functions: Interrupt 17h

PC BIOS Interrupt 17h printer interfaces are **not** supported, as no parallel port is available on the 6100 Computer. These PC BIOS functions are not available:

- Function AH = 0 Print character
- Function AH = 1 Initialize printer port
- Function AH = 2 Read status

4000 Series Programming Interfaces: 4000API.EXE

Overview

4000API.EXE provides the capability of porting 4000 Series applications to the PEN*KEY 6000 Series environment by providing programming interfaces that these applications expect. These programming interfaces greatly reduce the amount of work required to port an application to the 6100 Computer.

Please note that 4000API.EXE provides proprietary functions that are not industry standards. Use of certain 4000API.EXE functions limit the ability of the application to function on standard platforms. If you desire portability and standardization of applications, consider modifying old programs and developing new programs to conform to industry standards. These PEN*KEY 6000 Series platforms are PC-compatible. When you want application portability to include PCs, use PC standards instead of the functions listed here.

Installation

To load 4000API.EXE for use with a C++ application, use this command:

```
4000API.EXE /C3 /PC /10 /14 /16
```

To load 4000API.EXE for use with a PL/N application, use these commands:

if an internal modem is used: `4000API.EXE /C3 /16 /10 /PC`

if an internal modem is not used: `4000API.EXE /16 /10 /PC`

Command Line Switches

/C3 Do not redirect COM3 to COM1. Do not ignore modem control signals. Existing PL/N applications frequently used COM3 to communicate with COM1, only without modem control signals being used. Typically, you should supply this switch, unless your application is a PL/N application that was originally designed to run on a 4000 Series Computer.

/PC Change the return value returned by interrupt 15h AX=01FBh. The default value is “SOFT BIOS V3.00”. When this switch is supplied, the value returned is “PC BIOS V3.00”.

/10 Do not trap interrupt 10h (the video BIOS interrupt). There are a couple of 4000-compatible video BIOS extensions enabled by default and disabled via this switch.

/14 Do not trap interrupt 14h (the COM port BIOS interrupt). By default, 4000API supplies 4000-compatible BIOS extensions for communicating to COM ports. By supplying this switch, the COM port BIOS supplied in ROM BIOS is used instead of the 4000API services. Typically, you should supply this switch, unless your application is a PL/N application that was originally to run on a 4000 Series Computer.

/16 Do not trap interrupt 16h (the keyboard BIOS interrupt). By default, 4000API emulates the 4000 keyboard, complete with the ability to remap keys. When this switch is supplied, PC-compatible keycodes are returned instead of 4000-style keycodes.

Supported Programming Interfaces

Some of the 4000 Series programming interface functions are supported for the 6100 Computer. Refer to the *Cross-Reference by Interrupt Numbers* on page 6-28, for a listing of the supported functions.

INT 10h: Display Services

These services consist of interrupts supporting the display screen, including topics such as: V25 Comparator port, screen refresh, contrast, and backlight.

INT 14h: Serial Communications Services

Software interrupt 14h is embedded into 4000API.EXE and allows access to the 4000 Series style serial communications channels. Three logical ports are mapped onto two physical ports. Port 0 and port 2 refer to the RS-232 ports. Port 2 acts like port 0 except that the modem control signals (RTS and CTS) are ignored. For example, Function 01h sends the character regardless of the state of those signals. Port 0 can act like port 2 with Function 05h.

A major enhancement that these communication functions provide is the ability to ignore modem control signals when sending and receiving characters. This is a non-PC standard, but allows the use of cables that do not have DTR/CSR and RTS/CTS loop-backs or three-wire-only devices (TXD, RXD, and GND). This feature is programmable via the extended port control functions.

Port 1 refers to the RS-485 port. This port does not use the modem control signals. If you try to read the signals for port 1, they always indicate “active”.

To extend battery life, shut off communications port power when the port is not in use. The power is turned on whenever you access a port. You must explicitly turn off the power with Function FFh.

The transmit and receive functions use a timeout value from a table at location 40:7C. The default timeout is 1, which represents about 1 second.

The extended initialize function allows you to set baud rates beyond those available on a PC. Since these functions are polled (not interrupt-driven), the faster baud rates are of little utility unless interrupt-driven functions are used in place of the BIOS functions to send and receive a character.

A major difference between the 4000 Series and PEN*KEY 6000 Series Computers lies in UART capability. The 6000 Series uses PC-compatible UARTs. The 4000 Series does not use the PC-compatible 8250 UARTs. Because of this difference, the following conditions are expected by 4000 code. These conditions do not exist in the PEN*KEY 6000 Series Computer. This may affect portability of code where these bits are incorrectly masked and tested; modifications to existing 4000 Series code may be required for proper functioning.

Not all status bit values of the PC-compatible UARTs are supported.

Break conditions are not generated or detected.

Sticky parity is not supported.

The transmit shift register empty and break detect signals are not available.

The transmit shift register empty and transmit holding register bits are both set when the UART is ready.

The carrier detect, ring indicator, delta carrier detect, trailing-edge ring indicator, delta data set ready, and delta clear-to-send signals are not available.

INT 15h: Multitasking Services (Description)

Subfunction 1, of Interrupt 15h, is traditionally reserved for cassette interface control. The 4000 Series BIOS replaced it with a set of multitasking services informally referred to as INT15. 4000API.EXE provides this same set of services for 4000 Series multitasking applications. These services are nonstandard, but 4000API.EXE may be executed on any PC to provide these services.

Tasks and Scheduling

Int15 manipulates two linked lists of task contexts, the ready list and the pended list. The pended list contains tasks that are waiting for some event. A task moves from this list to the ready list when its timeout expires or when the event on which it is waiting occurs.

The ready list is a queue of tasks scheduled for execution. Tasks enter at the tail of the queue and are activated from the head. All tasks have the same priority and are executed round-robin.

The currently executing task is not on either list. It moves to the pended list if it issues “wait any,” pend on mailbox, pend on queue, or delay with timeout commands. The current task moves to the ready list if its time slice expires (preemptive only) or if it voluntarily gives up control by issuing a delay with 0 timeout.

Int15 defaults to cooperative multitasking. That means that every task must occasionally give up control of the processor to let other tasks execute.

By enabling time-slicing, int15 becomes a preemptive multitasker. Each task in the ready list is allocated a 55-millisecond time slice. If the currently executing task exhausts its time slice before relinquishing control of the processor, it is forcibly swapped out and placed on the tail of the ready list.

Timeouts

Functions that require timeout values as input expect the register pair DXCX to contain the number of milliseconds before timeout. The timer resolution is the same as the system timer ticks: 55 milliseconds. A task that requests a one millisecond time out is guaranteed to wait at least one millisecond and possibly up to 55 milliseconds.

Resource Arbitration and Task Communication

Most of the BIOS device handling services are not reentrant. Tasks cannot share the same device (the display, for example) without an arbitrator that ensures that both tasks do not try to use the BIOS at the same time.

This resource arbitration can be handled via the intertask communication provided by mailboxes and queues.

Mailboxes

A mailbox is a 4-byte variable used for message-passing between tasks. A mailbox is empty if it contains 0; otherwise, it contains a message. A task can perform three mailbox operations: post, pend, and accept.

Posting is the action of placing a 4-byte message into a mailbox. Posting to a mailbox that is not empty overlays the message that was in the mailbox.

When a task pends on an empty mailbox, the task is moved to the pended list until another task posts a message to the mailbox. When a task pends on a mailbox with a message in it, the pend request returns without pending.

A task can accept from a mailbox if it wants to retrieve a message from the box without pending. An accept from an empty box returns zero.

Queues

A queue is an 8-byte variable for messages passing between tasks. A queue must be initialized to empty by calling the queue init function. A task can perform three queue operations: post, pend, and accept.

Posting is the action of adding a new message node to a queue. A message node consists of an 8-byte field (manipulated by INT 15) followed by a message field. The length and structure of the message field is up to the user. Queues differ from mailboxes in that posting to a nonempty queue just adds a new node to the tail of the queue.

When a task pends on an empty queue, the task is moved to the pended list until another task posts to the queue. When a task pends on a nonempty queue, the pend request returns immediately without pending.

A task can accept from a queue if it wants to retrieve a message from the queue without pending. An accept from an empty queue returns zero.

INT 15h: Intermec Miscellaneous System Services

These services consist of the following topics:

- Accumulating and computing the CRC on specified blocks of data
- Receiving, sending, and exchanging network packets
- Adjusting the processor speed
- Beeping the buzzer
- Returning a pointer to the BIOS version

INT 15h: PC-Like Miscellaneous System Services

These services consist of the following topics:

- Requesting system shutdown
- Translating keyboard scan codes
- Pending until a key is pressed on the keyboard

INT 16h: Keyboard Services

These services consist of the following topics:

- Set Typematic Rates
  - Turn the key repeat on and off
  - Setting the key repeat
- Turn the keyclick on and off
- Swapping the keyboard translate tables

Locating 4000API.EXE Interrupts

Several 4000API interrupts are supported for the 6100 Computer. The supported interrupts are listed in the *Interrupt Cross-Reference* table, located under the *Interrupt Cross-Ref (by number)* tab. Even though they are matrixed into the entire set of interrupts, each individual interrupt supported by 4000API.EXE can be identified by the notation, **4000API**, just under the interrupt heading.

The categories of interrupt services are listed in the center column of Table 6-3.

Individual interrupt definitions can be located:

- Using Table 6-3 (by keyword) listed in the right column, and searching for them in the Interrupt Index. Most nouns and verbs can locate an interrupt definition using the *Interrupt Index*, with the exception of *Set* and *Get*.
- Using the *Interrupt Cross-Reference* table

or just thumbing through the definitions, searching for the notation, **4000API**.

Table 6-3
Programming Interrupt Summary

INT#	Supported Interfaces	Keywords
10h	Video Functions	Backlight, Contrast, Comparator, Port, Refresh, Screen, V25
14h	Serial Communications	Character, Communications, Control, Extended, Initialize, Port, Power, Receive, Send, Serial, Status
15h	Multitasking Services	Accept, Create, Current, Delay, Delete, Disable, Empty, Enable, Identifier, Information, Initialize, Mailbox, Message, Multitask, Optional, Pend, Post, Reset, Queue, Return, Switch, Task, Timeout, Time-slicing
15h	Norand Miscellaneous System Services	Accumulate, Adjust, Beep, BIOS, Block, Buzzer, Compute, CRC, Data, Exchange, Network, Packet, Pointer, Processor, Receive, Return, Send, Speed, Version
15h	PC-Like Miscellaneous System Services	Battery, Keyboard, Low, Normal, Pend, Request, Scan code, Shutdown, System, Translate
16h	Keyboard Functions	Default, Key, Keyboard, Keyclick, Number, Rate, Repeat, Return, Swap, Tables, Timers, Translate, Turn on/off, Typematic

Unimplemented 4000 Series BIOS APIs

These 4000 Series Interrupt 16h Keyboard BIOS interfaces are not implemented in 4000API.EXE

- Function AH = 0FCh: Set and Read Keyboard Mask Value
- Function AH = 0FDh: Set Debounce Timer Value

These 4000 Series Interrupt 15h functions are not implemented in 4000API.EXE.

- Function AX = 01ECh: Huge Memory Transfer
- Function AX = 01EDh: Return ASIC Register
- Function AX = 01EEh: Return Silicon Serial Number
- Function AX = 01F1h: Turn On RAM Card Controller
- Function AX = 01F2h: Elapsed Timer Ticks
- Function AX = 01F3h: Turn Off RAM Card Controller

- Function AX = 01F6h: Battery Service
- Function AX = 01F7h: Set Low-Battery Interrupt Message
- Function AX = 01FCh, DL = 1: Set Length Of Available Memory
- Function AX = 01FCh, DL = 2: Allocate BIOS memory
- Function AX = 4100h: Wait On Any Event
- Function AH = 85h: System Request Has Been Pressed
- Function AX = 9102h: Post Keyboard
- System Resume Vector Interrupt 6Ch

4000 Series Screen Emulation: VROTATE.EXE, FONTSEL.EXE

VROTATE.EXE and FONTSEL.EXE are 4000 Series screen emulation programs for the 6100 Computer, in DOS text mode. These programs port applications, written for the 4000 Series platform to the 6100 Computer without modifying the display logic. VROTATE is a TSR program that provides screen rotation and FONTSEL provides the font characteristics. VROTATE and FONTSEL provide 4000 Series screen features, such as:

- Character mapping
- Cursor fixed mode
- Variable font sizes (20, 21, or 16 characters across the screen)

VROTATE and FONTSEL.EXE work together providing several font selections, as described below.

These programs replace the standard VGA BIOS interface available on the 6100 Computer with a 4000 Series compatible video BIOS interface. Since they replace the standard VGA BIOS, standard VGA calls are not available once VROTATE and FONTSEL are loaded. The VROTATE utility provides a rotate INT 10 interface that rotates the INT 10 calls into a portrait mode. It filters out video BIOS calls.

VROTATE and FONTSEL work only with the 320x240 pixel display of the 6100 Computer. In the absence of command line switches, the default mode for VROTATE and FONTSEL are as follows:

- Cursor chase video mode
- 8x16 bitmap font
- Inverse mode on (characters above 127 are mapped to inverse video characters 0–127 when the high bit is turned off)
- Cursor turned off

VROTATE.EXE, Parameters and Command Line Switches

The format for the VROTATE command line is:

```
VROTATE [x y [width height]] [-Repaint_scroll] [-Nowrap] [-PLN] [Disable]
```

where the brackets [] indicate optional parameters and the descriptions of the parameters are as follows:

Parameters	Description
x y	Coordinates of the upper-left corner of the display
width height	Window size and rotated position of the display, under which the INT 10 calls are limited or emulated
-Repaint_scroll	Causes the rotated BIOS to repaint the screen on all scroll operations, which is much slower, but very compatible
-Nowrap	Causes the screen to truncate at the right margin, instead of wrapping to the next line
-PLN	Applies hacks to support the PLN error screen
-Disable	Disables the VROTATE driver, which remains in memory, and can be reenabled by running VROTATE again

PL/N Options

The following is an additional parameter to be used for PL/N applications.

Parameters	Description
-PLN	To be used for PL/N applications.

Norand Enhanced Video BIOS Functions

The VROTATE utility implements the Norand Enhanced Video BIOS calls. The general Enhanced Call Formats are as follows:

On Entry:

AH = 7Fh Norand enhanced operations
AL = Subfunction

On Return:

AH = Error code
AL = 7Fh If enhanced function is supported

Subfunctions:

AL = 00h Get version information
AL = 13h Absolute write string
AL = 20h Repaint screen
AL = 21h Load or select font
AL = 28h Get window size
AL = 29h Set window size
AL = 2Ah Physical write image
AL = 30h Disable shadow buffer updates
AL = 32h Disable rotated video
AL = 33h Enable rotated video

For details of these interfaces, refer to the *Norand Enhanced Video BIOS* interrupts, starting on page 6-56 in this section.

FONTSEL.EXE, Parameters and Command Line Switches

This utility selects and loads the fonts in VROTATE. The command line is:

```
FONTSEL <font_number>
```

or

```
FONTSEL <font_number>=<[path\]font_file_name>
```

where:

<font_number> is one of the following:

0	Sets up the display for 8x8 pixels per character. Results in a 30-column, 20-line display.
1	Sets up the display for 8x16 pixels per character. Results in a 30-column, 20-line display.
2	Sets up a user-selectable display.
path\	Consists of the DOS path where the font file is located.
font_file_name	The name of a file that contains the font to be loaded.

▶ **NOTE:** *The current default font (and maximum font size) is 12x24.*

The first format is for selecting the current active font. The second format is for replacing a font in VROTATE, where **font_number** is the font to be replaced. If you replace the currently used font, also select the font again to cause the new character size to be used. If not trashed, it shows up on the display.

The format of a font file is:

```
Version       dw   1
Char_width    dw   ?
Char_height   dw   ?
Font_data     db   ? dup(?)
```

The 8x8 and 8x16 fonts can only be replaced with an 8x8 and 8x16 font. The user font can be replaced by any size font that meets the following requirements:

- Width must be a multiple of 4.
- Height must be a multiple of 8.
- Total size of the font data cannot exceed 9216 bytes, which is the size of the 12x24 font.

BMP Conversion Utility: BMPUTIL

This is a Windows utility for converting BMP files to a format used by the Nor- and Enhanced Video BIOS. This utility opens a standard Windows BMP file and generate the image data in a format useful for an application to send to the Physical Write Image BIOS function. The generated formats are C, Assembler, and binary.

Locating 4000 Series Video Interrupts

Several 4000 Series Video interrupts are supported for the 6100 Computer. The supported interrupts are listed in the *Interrupt Cross-Reference* table, located under the *Interrupt Cross-Ref (by number)* tab. Even though they are matrixed into the entire set of interrupts, each individual interrupt supported by the video drivers, VROTATE.EXE and FONTSEL.EXE can be identified by the notation, **VROTATE**, immediately under the interrupt heading. There are one or more reference tokens for each of the interrupt definitions, to accommodate interrupts supported by one or more applications.

The categories of interrupt services are listed in the center column of the following table. Individual interrupt definitions can be located:

- Using Table 6-4 (by keyword) listed in the right column, and searching for them in the *Interrupt Index*. Most nouns and verbs can locate an interrupt definition using the *Interrupt Index*, with the exception of *Set* and *Get*.
- Using the *Interrupt Cross-Reference* table

or just thumbing through the definitions, searching for the notation, **VROTATE**.

Table 6-4
Interrupts Supported by VROTATE.EXE

INT#	Supported Interfaces	Keywords
10h	Alternate Settings Video Functions	Alternate, Current Display, Parameter, Physical, Pointer, Refresh, Return, Screen, Settings, Size
10h	Set Norand Specific Display Modes	Chase, Fixed, Invert, Mode
10h	Norand (enhanced) Video BIOS	Absolute, Buffer, Font, Image, Information, Load, Physical, Repaint, Rotated, Select, Shadow, String, Size, Text, Updates, Version, Video, Window
10h	Display Services	Active, Attribute, Character, Cursor, Mode, Page, Position, Return, Scroll, State, Teletype, Type, Video
10h	Programmable Font Support	Default, Font, Load, System, User

Unimplemented 4000 Series Video Functions

The following functions remain unimplemented within the facilities for screen rotation and font mapping:

- Function AX = 04h: Read Light Pen Position and Mode
- Function AH = 12h, Subfunction 20h: Select Alternate Print Screen
- Function AH = 12h, Subfunction 35h: Switch Active Display
- Function AX = 12E2h: Set 16-line display parameter packet
- Function AX = 12E3h: Set Graphics Mode
- Function AH = 12h, Subfunction BL = 0FDh: Return Physical Display Size
- Function AH = 0FEh: Get Video Buffer Address
- Function AX = 0FFh: Refresh the Screen

ATA BIOS: ATABIOS.SYS

Overview

ATABIOS.SYS is a DOS block device driver for removable hard disks (such as PC Card ATA devices), using card and socket services. It identifies and configures cards with a device ID of 0Dh and a function code of 4 (FIXED_DISK_CARD) and takes over the INT 13 interface for doing the physical I/O to the device. It is used with NORATA.SYS.

Usage

ATABIOS.SYS comes with several command line switches. They are indicated by a preceding slash or dash and are case-insensitive (such as –i and /I both indicate the same switch). Some switches are followed by data (text or numeric), in which the date immediately follows the switch (no space between).

Data Format

- T is the indicator for a text character.
- H is the indicator for a hexidecimal digit.
- D is the indicator for a decimal digit.

EXAMPLE: A switch shown as NTTTTTTTT indicates that the “N” switch is followed by (up to) 8 characters (such as –nMODEM1). All switches are optional, and if omitted does NOT default to the switch or the value indicated.

Data Definitions

- R Retry command after a suspend
- DH Specify device type (default D)
- FH Specify function code (default 4)
- IHHH Specify I/O address (default – first available valid configuration)

Command Line Switches

There are no switches for ATABIOS.SYS.

Cross-Reference by Interrupt Numbers

This topic consists of the entire combined list of interrupts supported for the 6100 Computer. Each of the table entries in the cross-reference table points to the page number for the associated interrupt title and its description.

The following cross-reference consists of the entire combined list of interrupts supported for the 6100 Computer, by interrupt number first, then by the subfunction registers.

Table 6-5
Interrupt Cross-Reference

INT #	Interrupt Functions	Subfunctions	Page
08h	System Timer		6-13

Table 6-5 (Continued)
Interrupt Cross-Reference

INT #	Interrupt Functions	Subfunctions				Page
09h	Standard Keyboard Interface					6-35
10h	Display Services (Continued)	AH=	AL=	BH=	BL=	
	Set Display Mode	00h				6-36
	Set Cursor Type	01h				6-37
	Read V25 Comparator Port	01h	F9h			6-37
	Set Cursor Position	02h				6-38
	Read Cursor Position and Mode	03h				6-38
	Set Active Display Page	05h				6-38
	Scroll Active Page Up	06h				6-39
	Scroll Active Page Down	07h				6-39
	Read Attribute and Character at Cursor Position	08h				6-39
	Write Attribute and Character at Cursor Position	09h				6-40
	Write Character Only at Cursor Position	0Ah				6-40
	Set Color Palette	0Bh				6-41
	Write Graphics Dot	0Ch				6-41
	Read Graphics Dot	0Dh				6-41
	Teletype Character Write	0Eh				6-42
	Return Current Video State	0Fh				6-42
	Set Palette Register	10h	00h			6-42
	Set Border Color	10h	01h			6-42
	Set Palette and Border	10h	02h			6-43
	Toggle Blink and Intensity Bit	10h	03h			6-43
	Get Palette Register	10h	07h			6-43
	Get Border Color	10h	08h			6-43
	Get Palette and Border	10h	09h			6-44
	Set Color Register	10h	10h			6-44
	Set Block of Color Registers	10h	12h			6-44
	Set Color Page State	10h	13h			6-45
	Get Color Register	10h	15h			6-45
	Get Block of Color Registers	10h	17h			6-45
	Set PEL Mask	10h	18h			6-45
	Get PEL Mask	10h	19h			6-46
	Get Color Page State	10h	1Ah			6-46
	Set Gray Scale Values	10h	1Bh			6-46
	Load User Font	11h	00h/10h			6-47
	Load ROM 8x14 Fonts	11h	01h/11h			6-47
	Load ROM 8x8 Fonts	11h	02h/12h			6-47
	Set Block Specifier	11h	03h			6-48
	Load ROM 8x14 Font	11h	04h/14h			6-48
	Set INT 1Fh Font Pointer	11h	20h			6-48
	Set INT 43h for User's Font	11h	21h			6-49
	Set INT 43h for ROM 8x14 Font	11h	22h			6-49

Table 6-5 (Continued)
Interrupt Cross-Reference

INT #	Interrupt Functions	Subfunctions				Page
10h	Display Services (Continued)	AH=	AL=	BH=	BL=	
	Set INT 43h for ROM 8x8 Font	11h	23h			6-49
	Set INT 43h for ROM 8x16 Font	11h	24h			6-50
	Get Font Information	11h	30h			6-50
	Get Video Configuration Information	12h			10h	6-51
	Set Scan Lines	12h			30h	6-51
	Enable/Disable Default Palette Loading	12h			31h	6-52
	Enable/Disable Video	12h			32h	6-52
	Enable/Disable Gray Scale Summing	12h			33h	6-52
	Enable/Disable Cursor Emulation	12h			34h	6-52
	Enable/Disable Screen Refresh	12h			36h	6-53
	Set Physical Display Size	12h			FAh	6-53
	Return Pointer to Current Display Parameters	12h			FBh	6-53
	Return Physical Display Size	12h			FDh	6-53
	Read or Write Contrast	12h			FEh	6-54
	Backlight Off or On	12h			FFh	6-54
	Return Invert Mode	12h	DDh		FCh	6-54
	Enable Invert Mode	12h	DEh		FCh	6-54
	Disable Invert Mode	12h	DFh		FCh	6-55
	Set Chase Mode	12h	E0h		FCh	6-55
	Set Fixed Mode	12h	E1h		FCh	6-55
	Load User Font	14h	00h		FCh	6-56
	Load System Default Font	14h	01h		00h	6-56
	Get Version Information	7Fh	00h			6-56
	Absolute Write String	7Fh	13h			6-57
	Repaint Text Window	7Fh	20h			6-57
	Load or Select Font	7Fh	21h			6-57
	Get Window Size	7Fh	28h			6-58
	Physical Write Image	7Fh	2Ah			6-58
	Disable Shadow Buffer Updates	7Fh	30h			6-59
	Disable Rotated Video	7Fh	32h			6-59
	Enable Rotated Video	7Fh	33h			6-59
11h	Equipment Determination					6-14
12h	Memory Size Determination					6-15
13h	Disk Services	AH=	AL=	BH=	BL=	
	Reset Disk System	00h				6-59
	Read Status of Last Operation	01h				6-60
	Read Sectors into Memory	02h				6-60
	Write Sectors from Memory	03h				6-60
	Verify Sectors	04h				6-61

Table 6-5 (Continued)
Interrupt Cross-Reference

INT #	Interrupt Functions	Subfunctions				Page
13h	Disk Services (Continued)	AH=	AL=	BH=	BL=	
	Read Drive Parameters	08h				6-61
	Get Disk Type	15h				6-61
	Detect Disk Change	16h				6-62
	Set Media Type	18h				6-62
	Disable RAM Drive	DAh				6-62
	Enable RAM Drive	EAh				6-63
	Enable Checksum	ECh				6-63
14h	Serial Communications Services	AH=	AL=	BH=	BL=	
	Initialize Communications Port	00h				6-64
	Send a Character	01h				6-65
	Receive a Character	02h				6-65
	Get Port Status	03h				6-65
	Extended Initialize	04h				6-66
	Extended Port Control	05h				6-67
	Port Power On or Off	FFh				6-67
15h	System Services	AH=	AL=	BH=	BL=	
	Create a Task	01h	00h			6-68
	Delete a Task	01h	01h			6-68
	Return Information about a Task	01h	02h			6-69
	Disable Task Switching	01h	03h			6-69
	Enable Task Switching	01h	04h			6-69
	Reset Multitasker	01h	05h			6-69
	Pend on Mailbox with Optional Timeout	01h	06h			6-70
	Accept Message from Mailbox (No Pend)	01h	07h			6-70
	Post Message to Mailbox	01h	08h			6-70
	Pend on Queue with Optional Timeout	01h	09h			6-70
	Accept Message from Queue (No Pend)	01h	0Ah			6-71
	Post Message to Queue	01h	0Bh			6-71
	Delay Current Task	01h	0Ch			6-71
	Initialize a Queue Structure as Empty	01h	0Dh			6-71
	Enable Time-Slicing	01h	0Eh			6-71
	Disable Time-Slicing	01h	0Fh			6-72
	Set Task Identifier	01h	10h			6-72
	Accumulate CRC 16h	01h	F4h			6-72
	Exchange Network Packets	01h	F5h			6-73
	Adjust CX for Processor Speed	01h	F8h			6-73
	Beep the Buzzer	01h	FAh			6-74
	Return Pointer to BIOS Version	01h	FBh			6-74
	Receive a Network Packet	01h	FDh			6-74
	Send a Network Packet	01h	FEh			6-75
	Compute CRC 16 on Block of Data	01h	FFh			6-75
	Request System Shutdown, Normal	42h	00h			6-75

Table 6-5 (Continued)
Interrupt Cross-Reference

INT #	Interrupt Functions	Subfunctions				Page
15h	System Services (Continued)	AH	AL	BH	BL	
	Request System Shutdown, Low Battery	42h	01h			6-75
	Translate Keyboard Scan Code	4Fh				6-76
	Keyboard Intercept	4Fh				6-76
	APM Installation Check	53h	00h	00h	00h	6-77
	APM Real Mode Interface Connect	53h	01h	00h	00h	6-77
	APM Interface Disconnect	53h	04h	00h	00h	6-78
	CPU Idle	53h	05h			6-78
	CPU Busy	53h	06h			6-79
	Set Power State	53h	07h			6-79
	Enable/Disable Power Management	53h	08h	00h	01h	6-80
	Get Power Status	53h	0Ah	00h	01h	6-80
	Get PM Event	53h	0Bh			6-81
	Get Power State	53h	0Ch			6-81
	Enable/Disable Device Power Management	53h	0Dh			6-82
	System Reset	53h	80h	1Dh		6-82
	Device Open	80h				6-83
	Device Close	81h				6-83
	Program Termination	82h				6-83
	Set Event Wait Interval	83h	00h			6-84
	Cancel Event Wait Interval	83h	01h			6-84
	System Request Key	85h				6-84
	Wait	86h				6-84
	Move Block	87h				6-85
	Read Extended Memory Size	88h				6-85
	Switch to Protected Mode	89h				6-85
	Device Busy	90h				6-86
	Pend On Keyboard	90h	02h			6-86
	Interrupt Complete	91h				6-86
	Return System Configuration Parameters Address	C0h				6-87
	Return Extended BIOS Data Area Segment	C1h				6-87
16h	Keyboard Services	AH=	AL=	BH=	BL=	
	Read Next ASCII Character	00h				6-92
	Set Zero Flag if Buffer Empty	01h				6-92
	Read Shift Status	02h				6-92
	Set Typematic Rates					
	Turn Off Key Repeat	03h	04h			6-93
	Set Key Repeat Timers	03h	05h			6-93
	Turn On Key Repeat	03h	06h			6-93
	Turn Keyclick Off or On	04h				6-93
	Put Key into Buffer as if from Keyboard	05h				6-93
	Read Next Extended ASCII Character	10h				6-94
	Set Zero Flag if Extended Key Buffer Empty	11h				6-94
	Read Extended Shift Status	12h				6-95

Table 6-5 (Continued)
Interrupt Cross-Reference

INT #	Interrupt Functions	Subfunctions				Page
16h	Keyboard Services (Continued)	AH	AL	BH	BL	
	Swap Keyboard Translate Tables	FEh				6-95
	Return Number of Keys on Default Keyboard	FFh				6-96
19h	System Reboot					6-16
1Ah	Timer and Real-Time Clock Services	AH=	AL=	BH=	BL=	
	Read System Timer Ticks	00h				6-97
	Set System Timer Ticks	01h				6-97
	Read the Real-Time Clock Time	02h				6-97
	Set the Real-Time Clock Time	03h				6-98
	Read the Real-Time Clock Date	04h				6-98
	Set the Real-Time Clock Date	05h				6-98
	Set the Real-Time Clock Alarm	06h				6-99
	Reset the Real-Time Clock Alarm	07h				6-99
	Read the Real-Time Clock Alarm	09h				6-99
33h	Standard Mouse Interface	AH=	AL=	BH=	BL=	
	Mouse Reset and Status	00h	00h			6-100
	Show Cursor	00h	01h			6-100
	Hide Cursor	00h	02h			6-100
	Get Button Status and Mouse Position	00h	03h			6-100
	Set Cursor Position	00h	04h			6-101
	Get Button Press Information	00h	05h			6-101
	Get Button Release Information	00h	06h			6-101
	Set Minimum and Maximum x Cursor Position	00h	07h			6-101
	Set Minimum and Maximum y Cursor Position	00h	08h			6-102
	Set Graphics Cursor Block	00h	09h			6-102
	Set Text Cursor	00h	0Ah			6-102
	Read Motion Counters	00h	0Bh			6-102
	Set Interrupt Subroutine Call Mask and Address	00h	0Ch			6-102
	Light Pen Emulation Mode On	00h	0Dh			6-103
	Light Pen Emulation Mode Off	00h	0Eh			6-103
	Set Mickey to Pixel Ratio	00h	0Fh			6-103
	Conditional Off	00h	10h			6-103
	Set Double-Speed Threshold	00h	13h			6-103
	Swap Interrupt Subroutines	00h	14h			6-104
	Get Status Block Size	00h	15h			6-104
	Save Driver Status	00h	16h			6-104
	Restore Driver Status	00h	17h			6-104
	Set Alternate Subroutine Call Mask and Address	00h	18h			6-104
	Get User Alternate Interrupt Address	00h	19h			6-105
	Set Mouse Sensitivity	00h	1Ah			6-105
	Get Mouse Sensitivity	00h	1Bh			6-105

Table 6-5 (Continued)
Interrupt Cross-Reference

INT #	Interrupt Functions	Subfunctions				Page
33h	Standard Mouse Interface (Continued)	AH=	AL=	BH=	BL=	
	Set Mouse Interrupt Rate	00h	1Ch			6-105
	Set Display Page Number	00h	1Dh			6-105
	Get Display Page Number	00h	1Eh			6-106
	Disable Mouse Driver	00h	1Fh			6-106
	Enable Mouse Driver	00h	20h			6-106
	Software Reset	00h	21h			6-106
	Set Language for Messages	00h	22h			6-106
	Get Language Number	00h	23h			6-107
	Get Driver Version, Mouse Type, and IRQ Number	00h	24h			6-107
70h	Real-Time Clock Interrupt					6-16

Interrupt Definitions

These interrupts are organized by interrupt number, first, then by subfunction, as shown in the *Interrupt Cross-Reference* table, starting on page 6-28.

For the entire list of interrupts (organized alphabetically by interrupt name), refer to the *Interrupt Index* at the end of this publication.

A reference token is included at the beginning of each interrupt definition to assist you in determining which application (or applications) support that particular interrupt. This reference token is a name that represents the application handling the interrupt (such as 4000API.EXE, VROTATE.EXE, ELANAPM.EXE, or BIOS) for the interrupt function being defined.

In some cases, there may be more than one token. If these tokens are together at the top of the interrupt definition, then the same definition applies for both. In some cases, there may be two tokens, but each token has its own definition. For example, if there is a BIOS token and a 4000API token with a different description and a description for each token, then choose the applicable definition, according to that application that is handling the interrupt.

All text between that token and the next interrupt definition (or to the next token) applies to that interrupt for the application (or for BIOS).

- BIOS **BIOS**
- VROTATE.EXE **VROTATE**
- 4000API.EXE **4000API**
- ELANAPM.EXE **ELANAPM**
- 61MOUSE.COM **MOUSE**

Standard Keyboard Interface: INT 09h

Matrix scan codes are converted to PC-compatible scan codes before Interrupt 09h is issued. Standard scan codes are read from port 60h. If the scan code is not a request for an internal function or a control key press, both the character code and scan code (two bytes) are placed in a 16-word circular buffer at the position pointed to by the keyboard tail buffer pointer. The tail pointer is then incremented by two and allowed to wrap around the buffer.

INT 09h supports the following internal functions:

Function	Description
Ctrl-Alt-Del	System Reset. Detection of a Control-Alt-Delete sequence causes the handler to set the reset flag to 1234h (do not do a memory test) and jump to the POST.
Ctrl-Scroll Lock	Break. The Ctl-Break sequence issues an INT 1Bh and returns the value zero.
Shift-PrtSc	Print Screen. Shift-Print Screen issues an INT 05h.
Ctrl-Num Lock	Pause. The Pause key causes the handler to issue INT 15, Function 41h to wait for a valid ASCII keystroke.

The keyboard handler issues an INT 15h, Function 91h with AL = 2 on return from interrupt to indicate a keystroke is available for keys that return values.

The keyboard handler issues an INT 15h, Function 4Fh after reading the scan code from port 60, for the system to replace or absorb the scan code. On return from INT 15, the code is processed if the carry flag is set. Otherwise, the key is not processed.

The SYSREQ key issues an INT 15, Function 85h to inform the system of a make or break operation of the SYSREQ key.

Control, Alt , Shift keys, Num Lock, Scroll Lock, Pause, and Caps Lock key presses update the keyboard mode and type flags. No character code is returned by the keyboard BIOS for these key presses. Applications that want to know the status of these flags may inquire the keyboard BIOS through Interrupt 16h, Function 02h.

The Scan Code tables that list the character codes for AT-compatible keyboards, refer to the *Keyboard Services* Interrupt definitions (INT 16h), page 6-87.

Display Services: INT 10h

INT 10h, Function AH = 00h: Set Display Mode

BIOS

On Entry:

AH = 00h
AL = Display mode
0 = 40x25 B/W text
1 = 40x25 color text
2 = 80x25 B/W text
3 = 80x25 color text
4 = 320x200 color graphics
5 = 320x200 B/W graphics
6 = 640x200 color graphics
13 = 320x300 16-color graphics
14 = 640x200 16-color graphics
15 = 640x350 2-color graphics
16 = 640x350 16-color graphics
17 = 640x480 2-color graphics
18 = 640x480 16-color graphics
19 = 320x300 256-color graphics

On Return:

None

- Start address of the REGEN buffer data item is initialized to zero.
- crt_mode is set to the selected mode.
- crt_cols is set to the number of columns for the selected mode.
- page_len is set to 1000h for 80x25 modes, and 800h for 40x25 modes.
- page_offset is set to zero for all modes.
- cursor_loc array is initialized to zero for all modes.
- curstype is set to 0x607 for all modes.
- curpage is set to 0 for all modes.
- If bit 7 of AL is set, the display buffer is not cleared.
- Default mode is 3 (80 x 25 text color); set by the BIOS at power-up.

VROTATE

All requested input modes are saved in memory but no action is taken to place the video controller in a new mode. The screen is cleared.

On Entry:

AH = 00h
AL = Display mode (ignored)

On Return:

None

INT 10h, Function AH = 01h: Set Cursor Type

BIOS

The cursor is not displayed in graphics modes and the selected cursor type applies to all video pages. *curstype* is set to the value in CX. The cursor can be displayed on up to eight scan rows, numbered 0 (top) to 7 (bottom). The default setting is CH = 6, CL = 7. Setting bit 5 of CH causes the cursor to disappear.

On Entry:

AH = 01h
CH = Value 0–7 = Start scan row in character box.
CL = Value 0–7 = Ending scan row in character box.

On Return:

None

VROTATE

This function is ignored. The cursor shape does not change, so CH and CL are ignored with respect to cursor shape. The cursor top and bottom lines are saved to memory. The function does control whether the cursor is on or off for the currently selected video page. If CH is equal to 0XE0, the cursor is turned off. This checks for PL/N emulation of cursor on and cursor off. If this test fails and if bit 5 of CH is set, the cursor is turned off. Otherwise, the cursor is turned on.

On Entry:

AH = 01h
CH = Top cursor scan line
CL = Bottom cursor scan line

On Return:

None

INT 10h, Function AH = 01F9h: Read V25 Comparator Port

4000API

On Entry:

AH = 01h
AL = F9h
BH = Mask used with output of PORTT
BL = Trip voltage threshold to compare with

On Return:

BL = PORTT value
- Zero = Result of comparison
- Set = Current value below input trip-voltage threshold
- Reset = Current value above input trip-voltage threshold

INT 10h, Function AH = 02h: Set Cursor Position

BIOS, VROTATE

Sets the cursor position for the display page represented by BH. There is no error checking of cursor position to determine whether it is outside of the page buffer. Applies to all video modes, though not visible in graphic modes. A page need not be active for the function to perform properly. The cursor may be move off the screen, causing it to disappear. Position (0, 0) is the upper left corner of the display.

cursor_loc[page] is set to the value in DH and DL. The page number must be 0 for graphics modes. There are eight pages available for 40x25 text modes, numbered 0 7, and four pages for 80x25 text modes, numbered 0 4.

On Entry:

AH = 02h
BH = Display page number (zero based)
DH = Row
DL = Column

On Return:

None

INT 10h, Function AH = 03h: Read Cursor Position and Mode

BIOS, VROTATE

Returns the cursor position from the value in cursor_loc[page] for the page represented by BH in register DX. The page selected need not be active. The display page value is equal to zero for graphic modes.

On Entry:

AH = 03h
BH = Display page number

On Return:

DH = Cursor row
DL = Cursor column
CX = Cursor type
    CH = Start scan row in character box
    CL = Ending scan row in character box

INT 10h, Function AH = 05h: Set Active Display Page

BIOS, VROTATE

Makes the page represented by AL the active display page. All page numbers are zero-based. Page zero is the default page number for all video modes and starts at the beginning of the display memory. Pages 0–7 may be selected for modes 0–7 and 0x0D. Pages 0–3 may be selected for mode 0x0E. Pages 0 and 1 may be selected for modes 0x0F and 0x10. The cursor position and display information are maintained for pages during switches.

▶ **NOTE:** *For VROTATE.EXE, only pages 0 and 1 are supported.*

On Entry:

AH = 05h
AL = Page number (zero based)

On Return:

None

INT 10h, Function AH = 06h: Scroll Active Page Up

BIOS, VROTATE

Allows active display page scroll up the number of lines indicated by AL. If AL = 0, the entire window is blanked. Information outside the scroll window indicated by the values in CH, CL, DH, and DL is not altered. Works in all display modes. For 320x200 graphics modes, the position that determines the window area is based on a 40x25 display. For 640x200 and 640x480 graphic modes, the position used is based on a 80x25 display.

On Entry:

AH = 06h
AL = Number of lines to scroll
BH = Attribute for blank lines
CX = Upper-left corner
  CH = Row
  CL = Column
DX = Lower-right corner
  DH = Row
  DL = Column

On Return:

None

INT 10h, Function AH = 07h: Scroll Active Page Down

BIOS, VROTATE

Allows the active display page to scroll down the number of lines indicated by AL. If AL = 0, the entire window is blanked. Information outside the scroll window indicated by the values in CH, CL, DH, and DL is not altered. Works in all display modes. For 320x200 graphics modes, the position used to determine the window area is based on a 40x25 display. For 640x200 and 640x480 graphic modes, the position used is based on a 80x25 display.

On Entry:

AH = 07h
AL = Number of lines to scroll
BH = Attribute for blank lines
CX = Upper-left corner
  CH = Row
  CL = Column
DX = Lower-right corner
  DH = Row
  DL = Column

On Return:

None

INT 10h, Function AH = 08h: Read Attribute & Character at Cursor Position

BIOS, VROTATE

Reads character and attribute at current cursor position for selected page. Works for all video modes. The page can be inactive for this to operate correctly.

On Entry:

AH = 08h
BH = Display page number

On Return:

AH = Attribute of character
AL = Value of character at the cursor position

INT 10h, Function AH = 09h: Write Attribute & Character at Cursor Position

BIOS, VROTATE

Writes the character and attribute at the current cursor position, for the selected page. The character is written the number of times indicated by the value in CX. If the number of repeats exceed the end of a line, characters wrap to the next line. In graphics modes, wrapping does not take place. All ASCII values from 0–256 produces a display, including control characters. Control characters do not produce control functions but do cause special characters to display. Control characters do not alter the cursor position. The cursor is never advanced by the function. Pages that are not active can be written to.

In graphic modes, the bit map for ASCII values 128–255 are stored in a table pointed at by location 0:10Ch in the interrupt table location (Interrupt 43h). This value is initialized to an internal table maintained by the BIOS. This value may be altered to point to a user-defined table. The BIOS maintains a separate bit map table for ASCII values 0–127. The value in BL is the foreground color of the character being written, and if bit seven of BL is set, the color value is XOR'ed with the current contents of the screen position.

On Entry:

AH = 09h
AL = Value of the character
BH = Display page number
BL = Character attribute (or color in graphics mode)
CX = Count of characters to write

On Return:

None

INT 10h, Function AH = 0Ah: Write Character Only at Cursor Position

BIOS, VROTATE

Performs the same as Function 9, except that the attribute (of the previous character at the current cursor location) remains unchanged.

On Entry:

AH = 0Ah
AL = Value of the character
BH = Display page
CX = Count of characters to write

On Return:

None

INT 10h, Function AH = 0Bh: Set Color Palette

BIOS

BL is effective only for text modes (when BH = 0). In this instance, the border color is set to the value represented in BL. For 320x200 graphic modes, the background color (when BH = 0) is set to the color represented by the value in BL. The value for BL (when BH = 0) should range from 0–31, where values 16–31 represent the intense version of the colors 0–15.

When BH = 1, the color palette is selected for 320x200 graphic modes. In this instance, BL = 0 selects the palette. Two palettes (0–1) are available. If BL = 0, the palette is green (1), red (2), yellow (3). If BL = 1, the palette is cyan (1), magenta (2), white (3).

On Entry:

AH = 0Bh
BH = "0" = Sets the background and border colors for graphics modes or the border color for text modes.
"1" = Selects the palette in 320x200 color graphics mode.
BL = Color value to use with the color ID (value in BH).

On Return:

None

INT 10h, Function AH = 0Ch: Write Graphics Dot

BIOS

Writes a specified pixel to video memory at the location represented by DX and CX. The value in AL is the color value for graphic modes. If bit 7 of AL is set, the color value is XOR'ed with the current screen location contents. The range of pixel values depends on the current video mode.

On Entry:

AH = 0Ch
AL = Dot value
BH = Page number
CX = Dot column number:
0–320 = modes 4–5
0–640 = mode 6
DX = Dot row number (0–200)

On Return:

None

INT 10h, Function AH = 0Dh: Read Graphics Dot

BIOS

Reads the value of the picture element at the location specified by CX and DX. The value returned in AL depends on the current video mode.

On Entry:

AH = 0Dh
BH = Page number
CX = Dot column number:
0–320 = modes 4–5
0–640 = mode 6
DX = Dot row number (0–200)

On Return:

AL = Color value

INT 10h, Function AH = 0Eh: Teletype Character Write

BIOS, VROTATE

Writes characters to the display as if it was a serial terminal. It functions in all display modes. A carriage return homes the cursor on the current line. A line feed advances the cursor to the next line. Backspace backs up the cursor, but only to the beginning of the current line. Bell beeps the buzzer. All other characters are displayable characters. The cursor is always advanced after a character is written. If the end of a line is reached, the cursor is advanced to the next line. If the position of the last line of the display is written to, the display is scrolled up by one line.

On Entry:

AH = 0Eh
AL = Character to write
BH = Page
BL = Foreground color in graphics mode

On Return:

None

INT 10h, Function AH = 0Fh: Return Current Video State

BIOS, VROTATE

Returns information stored in locations maintained by Functions 0 and 5.

On Entry:

AH = 0Fh

On Return:

AH = Number of screen columns
AL = Mode currently set
BH = Active display page

INT 10h, Function AX = 1000h: Set Palette Register

BIOS

Sets the selected palette register to the color value in BH.

On Entry:

AH = 10h
AL = 00h
BH = Color value
BL = Palette register (00–0Fh)

On Return:

None

INT 10h, Function AX = 1001h: Set Border Color

BIOS

Sets the color of the screen border to the color value in BH.

On Entry:

AH = 10h
AL = 01h
BH = Color value

On Return:

None

INT 10h, Function AX = 1002h: Set Palette and Border

BIOS

Sets all palette registers and border color in one call. The color list is 17 bytes long. The first 16 bytes represent the color values loaded into the palette registers 0–15. The last byte is the border color value. The default palette is:

Pixel Value	Color	Pixel Value	Color
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Brown	14	Yellow
7	White	15	Intense White
8	Gray		

On Entry:

AH = 10h
AL = 02h
ES:DX = Pointer to color list

On Return:

None

INT 10h, Function AX = 1003h: Toggle Blink and Intensity Bit

BIOS

On Entry:

AH = 10h
AL = 03h
BH = Enable:
  00h = Intensity
  01h = Blink

On Return:

None

INT 10h, Function AX = 1007h: Get Palette Register

BIOS

Returns the color associated with the specified palette register.

On Entry:

AH = 10h
AL = 07h
BL = Palette register

On Return:

BH = Color

INT 10h, Function AX = 1008h: Get Border Color

BIOS

On Entry:

AH = 10h
AL = 08h

On Return:

BH = Border color

INT 10h, Function AX = 1009h: Get Palette and Border

BIOS

On Entry:

AH = 10h
AL = 09h
ES:BX = Pointer to 17-byte buffer

On Return:

ES:BX = Pointer to buffer containing the palette values in bytes 0–15 and the border color in byte 16

INT 10h, Function AX = 1010h: Set Color Register

BIOS

For gray-scale summing, the weighted gray-scale value is calculated and stored into all three components of the color register.

On Entry:

AH = 10h
AL = 10h
BX = Color register
CH = Green value
CL = Blue value
DH = Red value

On Return:

None

INT 10h, Function AX = 1012h: Set Block of Color Registers

BIOS

The table consists of 3-byte entries, one per color to be programmed. The bytes of an individual entry are red, green and blue values for the associated color register. For gray-scale summing, the weighted gray-scale value is calculated and stored into all three components of the color register.

On Entry:

AH = 10h
AL = 12h
BX = First color value
CX = Number of color registers
ES:DX = Pointer to color table

On Return:

None

INT 10h, Function AX = 1013h: Set Color Page State

BIOS

Selects the paging mode for the color registers or selects an individual page of color registers.

On Entry:

AH = 10h
AL = 13h
BL = 00h = Set the paging mode
01h = Select a color register page
BH = Page to select
00h = 4 pages of 64 registers
01h = 16 pages of 16 registers

On Return:

None

INT 10h, Function AX = 1015h: Get Color Register

BIOS

Returns the contents of a color register as its red, green, and blue components.

On Entry:

AH = 10h
AL = 15h
BX = Color register

On Return:

CH = Green value
CL = Blue value
DH = Red value

INT 10h, Function AX = 1017h: Get Block of Color Registers

BIOS

Allows the red, green, and blue components associated with each of a set of color registers read in one operation. The table consists of a series of 3-byte entries. The bytes of an individual entry are red, green, and blue values for the associated color register.

On Entry:

AH = 10h
AL = 17h
BX = First color register
CX = Number of color registers
ES:DX = Pointer to buffer to receive the color list

On Return:

ES:DX = Pointer to buffer updated with color information

INT 10h, Function AX = 1018h: Set PEL Mask

BIOS

On Entry:

AH = 10h
AL = 18h
BL = PEL Mask to write

On Return:

None

INT 10h, Function AX = 1019h: Get PEL Mask

BIOS

On Entry:
AH = 10h
AL = 19h

On Return:
BX = PEL Mask Value

INT 10h, Function AX = 101Ah: Get Color Page State

BIOS

On Entry:
AH = 10h
AL = 1Ah

On Return:
BH = Color page
BL = Paging mode:
00h = 4 pages of 64 registers
01h = 16 pages of 16 registers

INT 10h, Function AX = 101Bh: Set Gray-Scale Values

BIOS

For each color register, the weighted sum of its red, green and blue values is calculated according to the formula:

```
gray_sum = 30% red + 59% green + 11% blue.
```

The gray_sum is then written back into all three components of the color register. The original values are lost.

On Entry:
AH = 10h
AL = 1Bh
BX = First color register
CX = Number of color registers

On Return:
None

INT 10h, Function AX = 1100h/1110h: Load User Font

BIOS

Loads the user font table into the specified block of character-generator RAM. Provides the font selection in text display modes.

If AL = 10h, page 0 must be active. The points, rows, and length of the refresh buffer are recalculated and the controller is reprogrammed with the maximum scan line (points – 1), cursor start (points –2) , cursor end (points – 1), vertical display end ((rows * points) –1), and underline location (points – 1, mode 7 only). To avoid unpredictable behavior, call this function immediately after mode set.

On Entry:

AH = 11h
AL = 00h or 10h
BH = Number of points (bytes) per character
BL = Block
CX = Number of characters defined by table
DX = First character code in table
ES:BP = Pointer to font table

On Return:

None

INT 10h, Function AX = 1101h/1111h: Load ROM 8x14 Fonts

BIOS

Loads the default ROM BIOS 8x14 font table into the specified block of the character-generator RAM. Provides the font selection in text display modes.

If AL = 11h, page 0 must be active. The points, rows, and length of the refresh buffer are recalculated and the controller is reprogrammed with the maximum scan line (points – 1), cursor start (points –2), cursor end (points – 1), vertical display end ((rows * points) –1), and underline location (points – 1, mode 7 only). To avoid unpredictable behavior, call this function immediately after mode set.

On Entry:

AH = 11h
AL = 01h or 11h
BL = Block

On Return:

None

INT 10h, Function AX = 1102h/1112h: Load ROM 8x8 Fonts

BIOS

Loads the default ROM BIOS 8x8 font table into the specified block of the character-generator RAM. Provides the font selection in text display modes.

If AL = 12h, page 0 must be active. The points, rows, and length of the refresh buffer are recalculated and the controller is reprogrammed with the maximum scan line (points – 1), cursor start (points –2), cursor end (points – 1), vertical display end ((rows * points) –1), and underline location (points – 1, mode 7 only). To avoid unpredictable behavior, call this function immediately after mode set.

On Entry:

AH = 11h
AL = 02h or 12h
BL = Block

On Return:

None

INT 10h, Function AX = 1103h: Set Block Specifier

BIOS

Determines the character blocks selected by bit 3 of the character attribute bytes in text display modes.

When using a 256-character set, both fields of BL should select the same character block. In these cases, bit 3 of the attribute byte selects the foreground intensity. With 512-character sets, the fields in BL select the blocks holding each half of the character set, so that bit 3 of the attribute byte selects either the upper or lower half of the character set.

On Entry:

AH = 11h
AL = 03h
BL = Character generator select code:
Bits 0, 1, 4 = character block selected for bit 3 of attribute byte = 0.
Bits 2, 3, 5 = character block selected for bit 3 of attribute byte = 1.
Bits 6, 7 are not used.

On Return:

None

INT 10h, Function AX = 1104h/1114h: Load ROM 8x14 Fonts

BIOS

Loads the default ROM BIOS 8x14 font table into the specified block of the character-generator RAM. This function provides the font selection in text display modes.

If AL = 14h, page 0 must be active. The points, rows, and length of the refresh buffer are recalculated and the controller is reprogrammed with the maximum scan line (points –1), cursor start (points –2) , cursor end (points –1), vertical display end ((rows * points) –1), and underline location (points –1, mode 7 only). To avoid unpredictable behavior, call this function immediately after mode set.

On Entry:

AH = 11h
AL = 04h or 14h
BL = Block

On Return:

None

INT 10h, Function AX = 1120h: Set INT 1Fh Font Pointer

BIOS

Sets the INT 1Fh pointer to point to the user table input in ES:BP. This table generates character codes 80h–FFh in graphics modes 4, 5 and 6.

On Entry:

AH = 11h
AL = 20h
ES:BP = Pointer to font table

On Return:

None

INT 10h, Function AX = 1121h: Set INT 43h for User's Font

BIOS

Sets the INT 43h vector to point to the user font table input in ES:BP and updates the video ROM BIOS data area. The video controller is not reprogrammed. Provides font selection in graphic display modes.

On Entry:

AH = 11h
AL = 21h
BL = 00h = Character rows specified by register DL
01h = 14 rows
02h = 25 rows
03h = 43 rows
CX = Bytes per character
DL = Character rows per screen
ES:BP = Pointer to font table

On Return:

None

INT 10h, Function AX = 1122h: Set INT 43h for ROM 8x14 Font

BIOS

Sets the INT 43h vector to point to the default ROM BIOS 8x14 font and updates the video ROM BIOS data area. The video controller is not reprogrammed. Provides font selection in graphic display modes.

On Entry:

AH = 11h
AL = 22h
BL = 00h = Character rows specified by register DL
01h = 14 rows
02h = 25 rows
03h = 43 rows
CX = Bytes per character
DL = Character rows per screen

On Return:

None

INT 10h, Function AX = 1123h: Set INT 43h for ROM 8x8 Font

BIOS

Sets the INT 43h vector to point to the default ROM BIOS 8x8 font and updates the video ROM BIOS data area. The video controller is not reprogrammed. Provides font selection in graphic display modes.

On Entry:

AH = 11h
AL = 23h
BL = 00h = Character rows specified by register DL
01h = 14 rows
02h = 25 rows
03h = 43 rows
CX = Bytes per character
DL = Character rows per screen

On Return:

None

INT 10h, Function AX = 1124h: Set INT 43h for ROM 8x16 Font

BIOS

Sets the INT 43h vector to point to the default ROM BIOS 8x16 font and updates the video ROM BIOS data area. Provides font selection in graphic display modes. The video controller is not reprogrammed.

On Entry:

AH = 11h
AL = 24h
BL = 00h = Character rows specified by register DL
01h = 14 rows
02h = 25 rows
03h = 43 rows
CX = Bytes per character
DL = Character rows per screen

On Return:

None

INT 10h, Function AX = 1130h: Get Font Information

BIOS

On Entry:

AH = 11h
AL = 30h
BH = Font code:
0 = Current INT 1Fh contents
1 = Current INT 43h contents
2 = ROM 8x14 font
3 = ROM 8x8 font (codes 0–127)
4 = ROM 8x8 font (codes 128–255)
5 = ROM alternate 9x14 font
6 = ROM 8x16 font
7 = ROM alternate 9x16 font.

On Return:

CX = Bytes per character
DL = Character rows on screen
ES:BP = Pointer to font table

Video, Alternate Settings: (AH=12h) Interrupt 10h

INT 10h, Function AH = 12h, Subfunction 10h: Get Video Configuration Info

BIOS

Returns configuration information for the video system. Feature bits are set from the input status register 0 in response to an output on the following specified feature control register bits. Bits 0–3 in CL indicate the state of VGA dip switches 1–4, respectively.

Feature Bits (CH)	Feature Control Output Bit	Input Status Bit
0	0	5
1	0	6
2	1	5
3	1	6
4–7	Not used	---

On Entry:

AH = 12h
BL = 10h

On Return:

BH = Display type:
- 0 = Color
- 1 = Monochrome

BL = Memory installed on VGA board:
- 0 = 64k
- 1 = 128k
- 2 = 192k
- 3 = 256k

CH = Feature bits
CL = Switch settings

INT 10h, Function AH = 12h, Subfunction 30h: Set Scan Lines

BIOS

Selects the number of scan lines for text modes. The selected value takes effect the next time the set display mode function is performed.

On Entry:

AH = 12h
BL = 30h
AL = 0 = 200 scan lines
- 1 = 350 scan lines
- 2 = 400 scan lines

On Return:

AL = 12h = VGA active
- 00h = VGA not active

INT 10h, Function AH = 12h, Subfunction 31h: Enable/Disable Default Palette Loading

BIOS

On Entry:
AH = 12h
BL = 31h
AL = 00h = Enable default palette loading
01h = Disable default palette loading

On Return:
AL = 12h = Function supported

INT 10h, Function AH = 12h, Subfunction 32h: Enable/Disable Video

BIOS

Disables or enables CPU access to the video adapter's I/O ports and refresh buffer. The display is turned off or on.

- **On Entry:**
  AH = 12h
  BL = 32h
  AL = 0 = Enable video
  1 = Disable video
- **On Return:**
  AL = 12h = Function supported

INT 10h, Function AH = 12h, Subfunction 33h: Enable/Disable Gray Scale Summing

BIOS

Enables or disables gray-scale summing.

On Entry:
AH = 12h
BL = 33h
AL = 0 = Enable gray scale summing
1 = Disable gray scale summing

On Return:
AL = 12h = Function supported

INT 10h, Function AH = 12h, Subfunction 34h: Enable/Disable Cursor Emulation

BIOS

When enabled, the BIOS remaps the cursor starting and ending lines for the current character cell dimensions.

On Entry:
AH = 12h
BL = 34h
AL = 0 = Enable cursor emulation
1 = Disable cursor emulation

On Return:
AL = 12h = Function supported

INT 10h, Function AH = 12h, Subfunction 36h: Enable/Disable Screen Refresh

BIOS, 4000API

This video refresh control interrupt specifies whether the video contents should display on the screen. Disabling the video refresh blanks the screen. When video refresh is disabled, the entire screen displays the color specified by the DAC color register 00h.

On Entry:

AH = 12h
BL = 36h
AL = 0 = Enable screen refresh
1 = Disable screen refresh

On Return:

AL = 12h = Function supported

INT 10h, Function AH = 12h, BL = FAh: Set Physical Display Size

VROTATE

Supported for the 16-line display only.

On Entry:

AH = 12h
BL = 0FAh
CH = Number of rows
CL = Number of columns

On Return:

ES:DI = Points at the 16-line display parameter packet (this value is reserved for use by Intermec Technologies Corporation).

INT 10h, Function AH = 12h, BL = FBh: Return Pointer to Current Display Parameters

VROTATE

Returns a pointer to the 16-line display parameter packet and font information for the 16-line display only.

On Entry:

AH = 12h
BL = 0FBh

On Return:

ES:DI = Points at the 16-line display parameter packet
CX = Maximum number of programmable characters (if DX = 0, then DS:SI and BH have no meaning)
DS:SI = Points at current font table
BH = Number of bytes per character on font table

INT 10h, Function AH = 12h, BL = FDh: Return Physical Display Size

VROTATE

On Entry:

AH = 12h
BL = 0FDh

On Return:

AH = Number of rows
AL = Number of columns

INT 10h, Function AH = 12h, BL = FEh: Read or Write Contrast

VROTATE

Returns or changes the current contrast setting. Range is 0 (darkest) to 15.

On Entry:

AH = 12h
BL = 0FEh
AL = Request type:
0 = Read request
1 = Write request
BH = Contrast level, if write request

On Return:

BH = Contrast level, if read request

INT 10h, Function AH = 12h, BL = FFh: Backlight Off or On

4000API

Issued by the keyboard interrupt handler when the backlight key is pressed. If the light is active, it is deactivated. If inactive, activated.

On Entry:

AH = 12h
BL = 0FFh
AL = Request type:
0 = Activate backlight
1 = Turn light off
2 = Toggle the backlight

On Return:

None

Norand-Specific Display Modes: Interrupt 10h

INT 10h, Function AX = 12DDh, Subfunction BL = FCh: Return Invert Mode

VROTATE

On Entry:

AH = 12h
AL = 0DDh
BL = 0FCh

On Return:

AL = Nonzero = Invert mode set

INT 10h, Function AX = 12DEh, Subfunction BL = FCh: Enable Invert Mode

VROTATE

Enables character mapping of character codes above 128 by subtracting 128 from the character code. The character is then displayed in inverse video, mapped as character codes 0 through 127.

On Entry:

AH = 12h
AL = 0DEh
BL = 0FCH

On Return:

None

INT 10h, Function AH = 12DFh, Subfunction BL = FCh: Disable Invert Mode

VROTATE

On Entry:
AH = 12h
AL = 0DFh
BL = 0FCh

On Return:
None

INT 10h, Function AH = 12E0h, Subfunction BL = FCh: Set Chase Mode

VROTATE

Forces the physical screen window to "chase" the cursor around the screen.

On Entry:
AH = 12h
AL = 0E0h
BL = 0FCh

On Return:
None

INT 10h, Function AH = 12E1h, Subfunction BL = FCh: Set Fixed Mode

VROTATE

Forces the physical screen window to remain fixed over the virtual screen segment whose upper left corner is specified by the DX value.

On Entry:
AH = 12h
AL = 0E1h
BL = 0FCh
DH = Row window coordinate
DL = Column window coordinate

On Return:
None

Programmable Font Support: Interrupt 10h

INT 10h, Function AX = 1400h: Load User Font

VROTATE

The font file must be in the following format. The largest bitmapped font that can be supported is 8 KB.

xsize	byte	Number of font pixels in x direction
ysize	byte	Number of font pixels in y direction
font[(xsize+7/8),(ysize+7/8)]	byte	The actual bitmap font

On Entry:

AH = 14h
AL = 00h
BL = Request type
Bit 000h Load main font
Bit 0FFh Load entire 16-line display font
DX = First character value to be programmed
CX = Number of characters to be programmed
BH = Number of bytes per character (8 or 16)
ES:DI = Pointer to font bit map

On Return:

None

INT 10h, Function AX = 1401h: Load System Default Font

VROTATE

On Entry:

AH = 14h
AL = 01h
BL = 00h: Load main font

On Return:

None

Norand Enhanced Video BIOS: Interrupt 10h

INT 10h, Function AX = 7F00h: Get Version Information

VROTATE

Returns version information and validates if Norand Enhancements are installed.

On Entry:

AH = 7Fh
AL = 00h
ES:DI = Communications port number
CX = Size of target version structure

On Return:

AH = 0
AL = 7Fh
ES:DI = Filled with the following structure:

```
{
unsigned int  InfoSize;
Char Key[14]; // which should be “NORAND_VIDEO”
unsigned int Version;
unsigned int Revision;
}
```

INT 10h, Function AX = 7F13h: Absolute Write String

VROTATE

Performs absolute screen writes. Allows the system application to write strings anywhere on the screen. Overrides the emulation window.

On Entry:
AH = 7Fh
AL = 13h
BL = Attribute (Not yet implemented)
CX = Character count
DX = Start cursor position
ES:BP = String to write

On Return:
AH = 00h
AL = 7Fh

INT 10h, Function AX = 7F20h: Repaint Text Window

VROTATE

Repaints the displayed portion of the emulated text window.

On Entry:
AH = 7Fh
AL = 20h

On Return:
AH = 00h
AL = 7Fh

INT 10h, Function AX = 7F21h: Load or Select Font

VROTATE

Performs absolute screen writes. Allows the system application to write strings anywhere on the screen. Overrides the emulation window.

▶ **NOTE:** *If the currently-used font is changed, reselect that font to use the new character size.*

On Entry:
AH = 7Fh
AL = 21h
Select Font:
BH = 0 (Font select option)
BL = Font number, 0 = 8x8, 1 = 8x16, 2 = User defined
Load Font:
BH = 1 (Font loaded option)
BL = Font number
CH = Character height (must be a multiple of 8)
CL = Character width (must be a multiple of 4)
ES:DI = Font table (must be in rotated format)

On Return:
AL = 7Fh
AH = 0 = Successful operation
1 = Invalid font
2 = Bad load font format

INT 10h, Function AX = 7F28h: Get Window Size

VROTATE

Gets information about the current window size and position where the standard video BIOS calls are restricted. The restricted area is the only area that standard BIOS calls can modify.

Default window size may vary. Mileage may vary.

▶ **NOTE:** *For a rotated system, the calls are based on the rotated window,i.e. 0,0 for the rotated 0,0.*

On Entry:

AH = 7Fh
AL = 28h

On Return:

AX = 007Fh
CX = Window width
DX = Window height
SI = Window X position
DI = Window Y position

INT 10h, Function AX = 7F2Ah: Physical Write Image

VROTATE

Bolts graphical data anywhere to the physical screen. Call goes around the windowing to allow the application to place graphics outside the emulated video BIOS screen.

▶ **NOTE:** *Pass data with each scan line starting on a byte boundary.*

On Entry:

AH = 7Fh
AL = 2Ah
ES:DI = Pointer to image data block
Image data block format:

```
Type     dw 0           ; must be zero
  X,Y    dw ?           ; x,y on screen to place block
  Width  dw ?           ; width of image
  Height dw ?           ; height of image
  BPP    db ?           ; bits per pixel
  Image  db ? dup( ? ) ; image packed to match display
```

On Return:

AH = Error code
  0 = Successful
  1 = BPP not supported
AL = 7Fh

There is a windows utility (currently called “BMUTIL”) that takes a standard Windows BMP file that is in line art resolution (1 BPP) and generates a “C” include file with the data for the width, height, and image information. The assumption is that this can compile into a “C” application.

INT 10h, Function AX = 7F30h: Disable Shadow Buffer Updates

VROTATE

Disables the updates to the text shadows buffer. The shadow buffer repaints and does some scroll operation, along with normal video BIOS calls to get the character and screen attributes. The effect is disabling the shadow buffer is that a repaint or other operation is based upon old screen data.

On Entry:
AH = 7Fh
AL = 30h

On Return:
AH = Error code (0 = successful)
AL = 7Fh

INT 10h, Function AX = 7F32h: Disable Rotated Video

VROTATE

Disables the rotated text driver to manage the screen.

On Entry:
AH = 7Fh
AL = 32h

On Return:
AH = Error code (0 = successful)
AL = 7Fh

INT 10h, Function AX = 7F33h: Enable Rotated Video

VROTATE

Enables the rotated text driver to manage the screen.

On Entry:
AH = 7Fh
AL = 33h

On Return:
AH = Error code (0 = successful)
AL = 7Fh

Disk Services: Interrupt 13h

INT 13h, Function AH = 00h: Reset Disk System

BIOS

Unsupported and always returns the carry clear and AH = 0.

On Entry:
AH = 00h
DL = Drive number

On Return:
Carry = Set (if AH is nonzero)
AH = Disk status

INT 13h, Function AH = 01h: Read Status of Last Operation

BIOS

On Entry:

AH = 01h
DL = Drive number (if bit 7 = 0, this indicates diskette)

On Return:

Carry = Set (if AH is nonzero)
AH = Disk status

INT 13h, Function 02h: Read Sectors into Memory

BIOS

The requested sectors are transferred into the buffer.

On Entry:

AH = 02h
AL = Number of sectors
DH = Track number
DL = Drive number
CH = Cylinder number
CL = Sector number
ES:BX = Buffer address

On Return:

Carry = Set (if AH is nonzero)
AH = Disk status
AL = Number of sectors actually transferred

INT 13h, Function AH = 03h: Write Sectors from Memory

BIOS

The requested sectors are transferred from the buffer to the disk.

On Entry:

AH = 03h
AL = Number of sectors
DH = Track number
DL = Drive number
CH = Cylinder number
CL = Sector number
ES:BX = Buffer address

On Return:

Carry = Set (if AH is nonzero)
AH = Disk status
AL = Number of sectors actually transferred

INT 13h, Function AH = 04h: Verify Sectors

BIOS

The sector block checks for the requested sectors are verified.

On Entry:

AH = 04h
AL = Number of sectors
DH = Track number
DL = Drive number
CH = Cylinder number
CL = Sector number
ES:BX = Buffer address

On Return:

Carry = Set (if AH is nonzero)
AH = Disk status
AL = Number of sectors actually transferred

INT 13h, Function AH = 08h: Read Drive Parameters

BIOS

A fixed format is all that is supported for any drive. This function reports the native format for any given drive.

On Entry:

AH = 08h
DL = Drive number

On Return:

AH = Error code if carry set = 1
Carry = Set of invalid drive
AX = 0 if no error
BX = 5 if no error
CX = 7F08h = 128 cylinders, 8 sectors per track
DH = 0 = One head per cylinder
DL = 3 = Number of floppy drives

INT 13h, Function AH = 15h: Get Disk Type

BIOS

Returns AH = 01 (floppy with NO change-line support) as long as the drive number is valid.

On Entry:

AH = 15h
DL = Drive number

On Return:

Carry = Set if invalid drive (error code = 1 if carry set)
AH = Type code:
- 00h = No such drive
- 01h = Floppy without change-line support
- 02h = Floppy with change-line support
- 03h = Hard disk

CX:DX = Number of 512-byte sectors

INT 13h, Function AH = 16h: Detect Disk Change

BIOS

Verifies that the specified PC Card drive is available. This procedure returns disk change for PC Cards if they are present. It returns *not ready* if the PC Card is not present. Call Function 15h first to see whether the drive supports a change line.

On Entry:

AH = 16h
DL = Drive number

On Return:

AH = Type code:
- 00h = No such drive
- 01h = Floppy without change-line support
- 02h = Floppy with change-line support
- 03h = Hard disk

CX:DX = Number of 512-byte sectors
Carry = Set if invalid drive or media changed (Error code = 1 if carry set)
Clear if no media change
AH = Drive status
- 00h = No media change
- 01h = Invalid drive
- 06h = Media changed or change line not supported
- 80h = Card is not present

INT 13h, Function AH = 18h: Set Media Type

BIOS

On Entry:

AH = 18h
DL = Drive number
CH = Lower 8 bits of highest cylinder number
CL = Sectors per track (bits 0–5) top 2 bits of highest cylinder number (bits 6, 7)

On Return:

CX:DX = Number of 512-byte sectors
Carry = 0 (if no error)
AH = 0
1 (if error)
AH = Type code:
- 00h = No such drive
- 01h = Floppy without change-line support
- 02h = Floppy with change-line support
- 03h = Hard disk

INT 13h, Function AH = DAh: Disable RAM Drive

BIOS

Reprograms the memory controller to disable access to the RAM associated with the RAM drive.

On Entry:

AH = 0DAh
DL = 0

On Return:

AH =0

INT 13h, Function AH = EAh: Enable RAM Drive

BIOS

Reprograms the memory controller to enable access to the RAM associated with the RAM drive.

On Entry:

AH = 0EAh
DL = 0

On Return:

AH = 0

INT 13h, Function AH = ECh: Enable Checksum

BIOS

Copies the following RAM drive signature into the reserved portion of the RAM drive logical boot sector.

```
_RamDriveSignature label byte
  db “Norand RamDrive”,0
  db 0,0, 0Fh, 0FFh,0FFh,0FFh
  db 0F0h, 3Ch,99h,0C3h, 55h,0AAh
```

On Entry:

AH = 0ECh
ES:SI = Pointer to global descriptor table (GDT):

offset
00h = Uninitialized, null descriptor
08h = Uninitialized, is made into GDT descriptor
10h = Uninitialized, descriptor for source of move
18h = Uninitialized, descriptor for destination of move
20h = Uninitialized, used by BIOS
28h = Uninitialized, is made into SS descriptor 0

On Return:

Carry = 0 (if no error)
1 (if error)
AH = Return code
00h = Normal return
01h = Parity error
02h = Interrupt error
03h = Address line 20 gating failed

Serial Communications Services: Interrupt 14h

INT 14h, Function AH = 00h: Initialize Communications Port

BIOS, 4000API

Programs the Baud Rate Register and Byte Format Register for the selected port and return values of the Line Status Register and Modem Status Register.

On Entry:

AH = 00h
DX = Communications port number
AL = Communications parameters:

Bits 7, 6, 5: Baud Rate
000 = 110 bps
001 = 150 bps
010 = 300 bps
011 = 600 bps
100 = 1200 bps
101 = 2400 bps
110 = 4800 bps
111 = 9600 bps

Bits 4,3: Parity
00 = NO parity
10 = ODD parity
11 = EVEN parity

Bit 2: Stop Bits
0 = 1 stop bit
1 = 2 stop bits

Bits 1, 0: Word Length
10 = 7-bit word length
11 = 8-bit word length

On Return:

AH = ***Line status register:***
7 = Timeout (if set, other bits are meaningless)
6 = Transmit shift register empty
5 = Transmit holding register empty
4 = Unused (break detected)
3 = Framing error
2 = Parity error
1 = Overrun error
0 = Data ready

AL = ***Modem status register:***
7 = Carrier detect (CD)
6 = Ring Indicator (RI)
5 = Data Set Ready (DSR)
4 = Clear To Send (CTS)
3 = Delta CD
2 = Delta trailing edge RI
1 = Delta DSR
0 = Delta CTS

INT 14h, Function AH = 01h: Send a Character

BIOS, 4000API

Transmits the character in register AL via the communications port specified by DX. DTR and RTS are raised, and DSR and CTS are expected within a timeout period specified by an array at 0:7C. Once DSR and CTS are received, the character to be sent is placed in the transmit buffer register. The value of the Line Status Register is returned in AH.

On Entry:

AH = 01h
DX = Communications port number
AL = Character to be sent

On Return:

AH = Line status as for Function 00h

DTR and RTS are raised and DSR and CTS are expected within a timeout period specified by an array at 40:7C.

INT 14h, Function AH = 02h: Receive a Character

BIOS, 4000API

Receives a character in register AL via the communications port specified by DX. DTR is raised, RTS is dropped, and DSR is expected within a timeout period specified by an array at 40:7C. When DSR is received, a character is expected within the receive buffer within the timeout period. If a timeout occurs, the timeout bit is set in AH. If a character is received, it is returned in AL. The value of the Line Status Register is returned in AH.

On Entry:

AH = 02h
DX = Communications port number

On Return:

AH = Line status as for Function 00h
AL = Character received

DTR and RTS are raised and DSR and CTS are expected within a timeout period specified by an array at 40:7C.

INT 14h, Function AH = 03h: Get Port Status

BIOS, 4000API

Returns the current line and modem status for the communications port specified by DX. The value of the Line Status Register is returned in AH. The value of the Modem Status Register is returned in AL.

On Entry:

AH = 03h
DX = Communications port number

On Return:

AH = Line status (same as Function 00h)
AL = Modem status (same as Function 00h)

INT 14h, Function AH = 04h: Extended Initialize

BIOS, 4000API

Programs the Baud Rate Register and Byte Format Register for the selected port and return values of the Line Status Register and Modem Status Register.

On Entry:

AH = 04h
DX = Communications port number
BH =Parity setting:
- 0 = None
- 1 = Odd
- 2 = Even

BL = Stop bits:
- 0 = One
- 1 = Two

CH =Word length:
- 2 = 7 bits
- 3 = 8 bits

CL = Baud rate:
- 0 = 110
- 1 = 150
- 2 = 300
- 3 = 600
- 4 = 1200
- 5 = 2400
- 6 = 4800
- 7 = 9600
- 8 = 19200
- 9 = 38400
- 10 = 57600
- 11 = 115200
- 12 = 125000
- 13 = 250000
- 14 = 500000

On Return:

AH = Line status (same as Function 00h)
AL = Modem status (same as Function 00h)

This is a nonstandard PC function. It may not be available in every PC BIOS. Use caution where portability is a concern.

INT 14h, Function AH = 05h: Extended Port Control

4000API

On Entry:

AH = 05h
AL = 0 = Read modem control register
1 = Write modem control register
BL = Modem control register value (if write):
7 = Ignore modem control signals, COM1 only
6–2 = Reserved
1 = RTS
0 = Data terminal only
DX = Communications port number

On Return:

AH = Line status (same as Function 00h)
AL = Modem status (same as Function 00h)
BL = Modem control register value (if read)

This is a nonstandard PC function. It may not be available in every PC BIOS. Use caution where portability is a concern.

INT 14h, Function AH = FFh: Port Power Off or On

4000API

Turns off the communications port driver power.

On Entry:

AH = 0FFh
DX = Communications port number
AL = Turn communications port power
0 = Off
1 = On

On Return:

None

This is a nonstandard PC function. It may not be available in every PC BIOS. Use caution where portability is a concern.

System Services: Interrupt 15h

INT 15h, Function AX = 0100h: Create a Task

4000API

Creates a task and inserts the new task into the ready list.

On Entry:
AH = 01h
AL = 00h
ES:BX = Context pointer

On Return:
CX = New task identifier

The context pointer (ES:BX) points at a structure that looks like the following Code Fragment:

```
context struc
   dival   dw   ?   ;di register contents for new task
   sival   dw   ?   ;si register contents for new task
   bpval   dw   ?   ;bp register contents for new task
           dw   ?   ;does not care
   bxval   dw   ?   ;bx register contents for new task
   dxval   dw   ?   ;dx register contents for new task
   cxval   dw   ?   ;cx register contents for new task
           dw   ?   ;does not care
   esval   dw   ?   ;es register contents for new task
   dsval   dw   ?   ;ds register contents for new task
   ipval   dw   ?   ;ip register contents for new task
   csval   dw   ?   ;cs register contents for new task
           dw   ?   ;does not care
context ends
```

csval:ipval specifies the code segment and instruction pointer where the child task begins execution; the parent task MUST provide them. The child task always inherits the flags register from the parent.

Create returns a task ID that is computed with the following equation (ES:BX is the value passed to the create task function):

```
Task_id = ((es << 4) + bx) >> 4;
```

When a newly created task is activated, SS (stack segment) is set to the passed value of ES (extra segment); SP (stack pointer) is set to the passed value of BX. The task context is then popped off the stack into the processor registers and an IRET is executed to pass control to the task. Thus, the new tasks SS:SP point at the “base” of the context passed during the create command.

INT 15h, Function AX = 0101h: Delete a Task

4000API

On Entry:
AH = 01h
AL = 01h
CX = Task identifier (0 means delete current task)

On Return:
CX = Task identifier (0 if task not found)
ES:BX = Released TCB address

INT 15h, Function AX = 0102h: Return Information About a Task

4000API

Provided for diagnostic purposes. Because TCBs may move on every task switch, lock task switching prior to making this call and unlock after the information has been extracted. It is possible for the running task to obtain its task ID by calling with CX = 0.

It is not possible to obtain the TCB address for the current task. The TCB exists on the task stack only when the task is ready or pending (not running).

On Entry:
AH = 01h
AL = 02h
CX = Task ID (0 for currently active task)

On Return:
CX = Task identifier (0 if task not found)
ES:BX = TCB address or pointer to multitasking data area for active task.
SI = (Modified)
DI = (Modified)

INT 15h, Function AX = 0103h: Disable Task Switching

4000API

On Entry:
AH = 01h
AL = 03h

On Return:
None

INT 15h, Function AX = 0104h: Enable Task Switching

4000API

On Entry:
AH = 01h
AL = 04h

On Return:
None

INT 15h, Function AX = 0105h: Reset Multitasker

4000API

Purges the ready list and the pend list. Resets current task identifier to zero. Take this drastic measure only when resetting all hardware and firmware.

On Entry:
AH = 01h
AL = 05h

On Return:
None

INT 15h, Function AX = 0106h: Pend on Mailbox with Optional Timeout

4000API

On Entry:
AH = 01h
AL = 06h
ES:BX = Mailbox pointer
DX:CX = Timeout value in milliseconds; 0 means no timeout

On Return:
AX = (Undefined)
DX:CX = Message, if available; 0 otherwise

▶ **NOTE:** *INTERRUPT ROUTINES MUST NOT ISSUE THIS CALL.*

INT 15h, Function AX = 0107h: Accept Message from Mailbox (No Pend)

4000API

On Entry:
AH = 01h
AL = 07h
ES:BX = Mailbox pointer

On Return:
AX = (Undefined)
DX:CX = Message, if available; 0 otherwise

INT 15h, Function AX = 0108h: Post Message to Mailbox

4000API

On Entry:
AH = 01h
AL = 08h
ES:BX = Mailbox pointer
DX:CX = Message

On Return:
DX:CX = Zero if mailbox was empty, operation successful;
previous message if mailbox overrun

INT 15h, Function AX = 0109h: Pend on Queue with Optional Timeout

4000API

On Entry:
AH = 01h
AL = 09h
ES:BX = Pointer to queue
DX:CX = Timeout value in milliseconds; 0 means no timeout

On Return:
AX = (Undefined)
DX:CX = Pointer to queue message node if no timeout; 0 if timeout error

▶ **NOTE:** *INTERRUPT ROUTINES MUST NOT ISSUE THIS CALL.*

INT 15h, Function AX = 010Ah: Accept Message from Queue (No Pend)

4000API

On Entry:
AH = 01h
AL = 0Ah
ES:BX = Pointer to queue

On Return:
AX = (Undefined)
DX:CX = Pointer to queue message node if queue not empty;
0 if queue empty

INT 15h, Function AX = 010Bh: Post Message to Queue

4000API

On Entry:
AH = 01h
AL = 0Bh
ES:BX = Pointer to queue
DX:CX = Pointer to queue message node

On Return:
AX = (Undefined)

INT 15h, Function AX = 010Ch: Delay Current Task

4000API

On Entry:
AH = 01h
AL = 0Ch
DX:CX = Timeout value in milliseconds; 0 forces a task switch

On Return:
AX = (Undefined)

▶ **NOTE:** *INTERRUPT ROUTINES MUST NOT ISSUE THIS CALL.*

INT 15h, Function AX = 010Dh: Initialize a Queue Structure as Empty

4000API

On Entry:
AH = 01h
AL = 0Dh
ES:BX = Pointing at queue.

On Return:
None

INT 15h, Function AX = 010Eh: Enable Time-Slicing

4000API

On Entry:
AH = 01h
AL = 0Eh

On Return:
None

INT 15h, Function AX = 010Fh: Disable Time-Slicing

4000API

On Entry:

AH = 01h
AL = 0Fh

On Return:

None

INT 15h, Function AX = 0110h: Set Task Identifier

4000API

On Entry:

AH = 01h
AL = 10h
CX = New task ID for current task

On Return:

None

INT 15h, Function AX = 01F4h: Accumulate CRC 16h

4000API

The CRC is computed for the block using the seed passed in BX. The generator polynomial is CRC–16 = X**16 + X**15 + X**2 + X.

On Entry:

AH = 01h
AL = 0F4h
DS:SI = Points at the data block
CX = Contains the block length in bytes
BX = Contains the CRC seed

Output

DS:SI = Points past the end of the block
CX = Zero
AX = Contains the CRC

INT 15h, Function AX = 01F5h: Exchange Network Packets

4000API

Allows access to the BIOS Media ACcess (MAC) layer for sending and then immediately receiving a network packet. Set the COM channel for the proper communication rate, parity, and bits before making this call.

On Entry:

AH = 01h
AL = 0F5h
ES:BX = Pointer to cbuf to receive
DS:DI = Pointer to cbuf to send
DL = COM port (0 or 1)
CX = Timeout value in milliseconds for send and receive

On Return:

AX = (Undefined)
DX = (Undefined)
CX = +Length or –Error
–21 = Packet too long for this address
–41 = CRC error in packet for this address
–A1 = Timeout, transmit operation did not complete
–B1 = Timeout, receive macro service has stalled
–C1 = Timeout, transmit complete, no receive activity
–FF = Timeout, receive activity

Code fragment:

```
cbuf    struc
        dd   ?  ; BIOS scratch area
        db   ?  ; unused by BIOS
        db   ?  ; BIOS scratch area
len     dw   ?  ; length of buffer data
        dw   ?  ; BIOS scratch area
adr     db   ?  ; network address
        dw   ?  ; BIOS scratch area
ctl     db   ?  ; frame control field
buffer  db   ?  ; dup(?); frame i-field buffer
        dw   ?  ; BIOS scratch area
cbuf    ends
```

INT 15h, Function AX = 01F8h: Adjust CX for Processor Speed

4000API

Programs that use software timing loops should use this function to adjust their loop counts to match the processor speed. The input value of CX is assumed to be the loop count for 4-Mhz operation (8-MHz crystal speed).

On Entry:

AH = 01h
AL = 0F8h
CX = Count value for 4-MHz operation

On Return:

CX = Adjusted count value for current processor speed
Zero = 0 = No counter underflow error
1 = Counter value underflowed
Carry = 0 = No counter overflow error
1 = Counter value overflowed

INT 15h, Function AX = 01FAh: Beep the Buzzer

4000API

On Entry:

AH = 01h
AL = 0FAh
CX = Beep duration in milliseconds (Approximate)

On Return:

None

INT 15h, Function AX = 01FBh: Return Pointer to BIOS Version

4000API

On Entry:

AH = 01h
AL = 0FBh

On Return:

ES:BX = Pointer to null-terminated ASCII string that contains the BIOS version number. (For example, "SOFT BIOS 3.00". However, if the /PC switch is used on 4000API, "PC BIOS 3.00" is returned.)

INT 15h, Function AX = 01FDh: Receive a Network Packet

4000API

Allows access to the BIOS MAC layer for receiving a network packet. Set the COM channel for the proper communication rate, parity, and bits before making this call.

On Entry:

AH = 01h
AL = 0FDh
CX = Timeout value in milliseconds
DX = COM port (0 or 1)
ES:BX = Pointer to cbuf (cbuf as in Function 1F5h)
`es:[bx].len` must indicate number bytes available in the buffer.

On Return:

DX = (Undefined)
AX = (Undefined)
CX = +length or –error (errors defined as Function 1F5h)
`es:[bx].adr` = Contains the destination address for this packet (all addresses are received).
`es:[bx].ctl` = Contains the packet control character.
`es:[bx].buffer` = Contains additional packet information, if appropriate.

INT 15h, Function AX = 01FEh: Send a Network Packet

4000API

Allows access to the BIOS MAC layer for sending a network packet. Set the COM channel for the proper communication rate, parity, and bits before making this call.

On Entry:

AH = 01h
AL = 0FEh
DX = COM port (0 or 1)
ES:BX = Pointer to cbuf (cbuf as in Function 1F5h)
`es:[bx].len` = Length of the data to be sent.
`es:[bx].adr` = The network address of this node.
`es:[bx].ctl` = The packet control character.
`es:[bx].buffer` = Additional packet information, if appropriate.

On Return:

AX = (Undefined)
CX = (Undefined)

INT 15h, Function AX = 01FFh: Compute CRC 16 on Block of Data

4000API

Generator polynomial is CRC–16 = X**16 + X**15 + X**2 + X.

On Entry:

AX = 01FFh
DS:SI = Points at the data block
CX = Contains the block length in bytes

On Return:

DS:SI = Points past the end of the block
CX = Zero
AX = CRC

INT 15h, Function AX = 4200h: Request System Shutdown, Normal

4000API

Wakes the unit from sleep with the ON/OFF key. Puts the processor to sleep to save power. It is issued by the BIOS whenever the system is idle. The display is turned off.

On Entry:

AH = 42h
AL = 00h

On Return:

None

INT 15h, Function AX = 4201h: Request System Shutdown, Low Battery

4000API

On Entry:

AH = 42h
AL = 01h

On Return:

None

▶ **NOTE:** *Applications you write should NOT allow the ON/OFF key to wake the unit from sleep. Force the display off, and turn off all peripherals. This function is issued by the BIOS in case of a critical power failure.*

INT 15h, Function AH = 4Fh: Translate Keyboard Scan Code

4000API

Invoked by the keyboard interrupt (INT 09h) handler to allow a user routine to translate the raw keyboard scan code. These scan codes are not PC-compatible. Refer to the topic on keyboard scan codes in the *Standard Keyboard Interface* paragraph on page 6-35 in this section.

On Entry:

AH = 4Fh
AL = Original scan code
Carry = Set

On Return:

AL = Original or modified scan code
Carry = Set means to post the key value
Reset forces the keyboard interrupt routine to ignore the key

INT 15h, Function AH = 4Fh: Keyboard Intercept

BIOS

Called by INT 09h ISR each time a key is pressed. It creates alternate keyboard layouts. Scan codes may be substituted or discarded by the intercept routine. The default BIOS routine returns the input scan code unmodified and carry set. Applications can call INT 15h, Function C0h to determine whether this feature is supported by the BIOS.

On Entry:

AH = 4Fh
AL = Scan code

On Return:

AL = Scan code
Carry = 0 = Scan code processed, does not buffer scan code
1 = Scan code not processed, does buffer the scan code

INT 15h, Function AX = 5300h: APM Installation Check

ELANAPM

Allows the APM driver (caller) to determine whether the system's BIOS supports the APM functionality and, if so, which version of the specification it supports. The APM version number returned from this call is the highest level of APM supported by the APM BIOS.

On Entry:

AH = 53h
AL = 00h
BX = 0000h

On Return:

Carry = 0 = APM is supported by BIOS
AH = 1 – APM major version number (in BCD format)
AL = 1 – APM minor version number (in BCD format)
BH = ASCII "P" character
BL = ASCII "M" character
CX = APM flags:

Bit		
	0 = 1	16-bit protected mode interface supported
	1 = 1	32-bit protected mode interface supported
	2 = 1	CPU Idle call slows processor clock speed
	2 = 0	CPU Idle call stops the clock
	3 = 1	APM BIOS Power Management disabled
	4 = 1	APM BIOS Power Management disengaged
Other bits		reserved

Carry = 1 if unsuccessful:
AH = Error code

09h Unrecognized device ID
86h APM not present

INT 15h, Function AX = 5301h: APM Real Mode Interface Connect

ELANAPM

Provides compatibility with the Microsoft APM 1.01 specification. Use of the call is not required for any of the APM functions to succeed, other than the Disconnect function.

On Entry:

AH = 53h
AL = 01h
BX = 0000h

On Return:

Carry = 0 if successful:
1 if unsuccessful:
AH = Error code:

02h Real mode interface connection already established
05h 16-bit protected mode interface already established
07h 32-bit protected mode interface already established
09h Unrecognized device ID

INT 15h, Function AX = 5304h: APM Interface Disconnect

ELANAPM

Provides compatibility with the Microsoft APM 1.01 specification. Use of the call does not prevent any of the APM functions from succeeding or failing, except for the Connect function. Use of the function has no effect upon the power management configuration.

On Entry:

AH = 53h
AL = 04h
BX = 0000h

On Return:

Carry = 0 if successful:
1 if unsuccessful:
AH = Error code:
03h Interface not connected
09h Unrecognized device ID

INT 15h, Function AX = 5305h: CPU Idle

ELANAPM

An APM driver or power-aware application can use this call to reduce the power consumed by the system. This call causes the system to halt the CPU and reduce the clock frequency of the system until an interrupt occurs. Any interrupt routine that generates I/O activity to one of the defined activity monitors or issues an APM busy call causes the IDLE state to exit and return control to the driver or power-aware application.

On Entry:

AH = 53h
AL = 05h

On Return:

Carry = 0 if successful:
1 if unsuccessful:
AH = Error code:
03h Interface not connected
0Bh Unrecognized device ID

INT 15h, Function AX = 5306h: CPU Busy

ELANAPM

Informs the APM BIOS that system is now busy. Busy calls prevent the system from entering Idle or Suspend states and cause the timers associated with these states to be reset.

System APM drivers or power-aware applications should call Busy during processing that does not generate I/O activity to one of the defined activity monitors to prevent slowdown of processing that occurs if the system enters Idle or Suspend during normal application processing.

On Entry:

AH = 53h
AL = 06h

On Return:

Carry = 0 if successful:
1 if unsuccessful:
AH = Error code:
03h Interface not connected
0Bh Unrecognized device ID

INT 15h, Function AX = 5307h: Set Power State

ELANAPM

Sets system or device specified in power device ID into the requested power state.

On Entry:

AH = 53h
AL = 07h
BX = Power device ID:
0001h = All devices power managed by the APM BIOS
01XXh = Display
02XXh = PC Card controller
04XXh = RS-232
05XXh = Network adapters
06XXh = PC Card Slots
0EXXh = Intermec defined devices where XXh = unit number (0 based) unit number of FFh means all devices in this class
CX = Power state:
0000h = Ready
0001h = Standby
0002h = Suspend
0003h = OFF
0004h = Last request processing notification
0005h = Last request rejected
** Supported only for Power Device ID 0001h

On Return:

Carry = 0 if successful:
1 if unsuccessful:
AH = Error code:
01h = Power management functionality disabled
09h = Unrecognized device ID
0Ah = Parameter value out of range
0Bh = Interface not engaged
60h = Unable to enter requested state

INT 15h, Function AX = 5308h: Enable/Disable Power Management

ELANAPM

Enables or disables automatic System Idle and System Suspend functions.

On Entry:

AH = 53h
AL = 08h
BX = 0001h
CX = Function code:
0000h Disable power management
0001h Enable power management

On Return:

Carry = 0 if successful:
1 if unsuccessful:
AH = Error code:
01h Power management functionality disabled
03h Interface not connected
09h Unrecognized device ID
0Ah Parameter value out or range (Function code)

INT 15h, Function AX = 530Ah: Get Power Status

ELANAPM

Returns the current power status of the system.

On Entry:

AH = 53h
AL = 0Ah
BX = 0001h = APM BIOS

On Return:

Carry = 0 if successful:
BH = AC line status
00h Off-line
01h On-line
FFh (undefined)
All other values reserved
BL = Battery status
00h High
01h Low
02h Critical
03h Charging
FFh (undefined)
All other values reserved
CH = Battery flag:
Bit
0 = 1 High
1 = 1 Low
2 = 1 Critical
3 = 1 Charging
7 = 1 No system battery or powerfail has occurred
All other bits reserved
FFh = (undefined)
All other values reserved

CL = Remaining battery life – percentage of charge
0–100 = Percentage of full charge
FFh = (undefined)
All other values reserved
DX = Remaining battery life
15 = Time units:
0 = seconds
1 = minutes
14–0 = Number of seconds or minutes:
0–7FFFh Valid number of seconds
0–7FFEh Valid number of minutes
FFFFh = (undefined)
Carry = 1 if unsuccessful:
AH = Error code (09h Unrecognized device ID)

INT 15h, Function AX = 530Bh: Get PM Event

ELANAPM

Gets PM Event returns the next pending PM event or indicates whether no PM events are pending. Call this function until there are no more pending PM events.

On Entry:
AH = 53h
AL = 0Bh

On Return:
Carry = 0 if successful:
BX = PM event code
1 if unsuccessful:
AH = Error code: (80h No power management events pending)

INT 15h, Function AX = 530Ch: Get Power State

ELANAPM

Returns the device power state only when a specific device ID is used.

On Entry:
AH = 53h
AL = 0Ch
BX = Power device ID
0001h = All devices power managed by the APM BIOS
01XXh = Display
02XXh = PC Card controller
04XXh = RS-232
05XXh = Network adapters
06XXh = PC Card slots
0EXXh = Intermec defined devices where XXh = unit number
(0 based) unit number of FFh means all devices in this class

On Return:
Carry = 0 if successful:
CX = Power state:
0000h Ready
0001h Standby
0002h Suspend
0003h OFF
1 if unsuccessful:
AH = Error code: (09h Unrecognized device ID)

INT 15h, Function AX = 530Dh: Enable/Disable Device Power Management

ELANAPM

Enables or disables APM BIOS automatic power management for a specified device. When disabled, the APM BIOS does not automatically power-manage the device. The only automatic power management that the BIOS does with devices is during Suspend. If power management of a device is disabled, during Suspend the power state of the device is not altered from the state set before Suspend was entered.

On Entry:

AH = 53h
AL = 0Dh
BX = Power device ID
- 0001h = All devices power managed by the APM BIOS
- 01XXh = Display
- 02XXh = PC Card controller
- 04XXh = RS-232
- 05XXh = Network adapters
- 06XXh = PC Card slots
- 0EXXh = Norand defined devices where XXh = unit number (0 based)

CX = Function code
- 0000h Disable
- 0001h Enable

On Return:

Carry = 0 if successful:
1 if unsuccessful:
AH = Error code: (09h Unrecognized device ID)

INT 15h, Function AX = 5380h, Subfunction BH = 1Dh: System Reset

BIOS

Performs a system reset. RESET.EXE provides this function as well.

On Entry:

AH = 53h
AL = 80h
BH = 1Dh

On Return:

None (system resets)

The following assembly code could be used in an application to produce the same results as RESET.EXE:

```
hwdreset  proc near
  mov     bh, 1Dh     ; cold reset
  mov     ax, 5380h
  int       15h
  cli
  hlt
hwreset     endp
```

INT 15h, Function AH = 80h: Device Open

BIOS

Default BIOS returns with AH = 0 and carry flag clear.

On Entry:

AH = 80h
BX = Device ID
CX = Process ID

On Return:

Carry = 0 if successful:
AH = 0
1 if unsuccessful:
AH = Status

INT 15h, Function AH = 81h: Device Close

BIOS

Default BIOS returns with AH = 0 and carry flag clear.

On Entry:

AH = 81h
BX = Device ID
CX = Process ID

On Return:

Carry = 0 if successful:
AH = 0
1 if unsuccessful:
AH = Status

INT 15h, Function AH = 82h: Program Termination

BIOS

Default BIOS returns with AH = 0 and carry flag clear.

On Entry:

AH = 82h
BX = Process ID

On Return:

Carry = 0 if successful:
AH = 0
1 if unsuccessful:
AH = Status

INT 15h, Function AX = 8300h: Set Event Wait Interval

BIOS

Programs the real-time clock to generate an interrupt after the specified amount of time and bit 7 of the byte pointed at by ES:BX is set. The duration of the wait interval is a multiple of 976 microseconds.

On Entry:

AH = 83h
AL = 00h
CX = High byte of interval in microseconds
DX = Low byte of interval in microseconds
ES:BX = Pointer to byte in calling program's memory that has bit 7 set when the interval expires

On Return:

Carry = 0 if successful:
AH = 0
1 if unsuccessful:
AH = Status

INT 15h, Function AX = 8301h: Cancel Event Wait Interval

BIOS

On Entry:

AH = 83h
AL = 01h

On Return:

None

INT 15h, Function AH = 85h: System Request Key

BIOS

Called by INT 09h ISR when the SYS REQ key is pressed. The default BIOS returns AH = 0 and carry clear.

On Entry:

AH = 85h
AL = 00 = Key make
01 = Key break

On Return:

Carry = 0 if successful:
AH = 0
1 if unsuccessful:
AH = Status

INT 15h, Function AH = 86h: Wait

BIOS

Suspends the calling program for the specified time period. The values in CX and DX are in microseconds but are rounded down to the nearest multiple of 976 microseconds.

On Entry:

AH = 86h
CX = High byte of interval in microseconds
DX = Low byte of interval in microseconds

On Return:

Carry = 0 if successful
1 if unsuccessful

INT 15h, Function AH = 87h: Move Block

BIOS

Copies a block of memory from anywhere in the system address space to anywhere else in the system address space. Memory space not generally available to real-mode programs is accessible through this function.

On Entry:

AH = 87h
CX = Number of 16-bit words to move
ES:SI = Pointer to 48-byte GDT table allocated by calling program

On Return:

AH = 00 = Successful move
01 = RAM parity error occurred
02 = Other exception occurred
03 = Gate address line 20 failed
Carry = 0 = No error
1 = Error

When called, two of the descriptors in the GDT table specify the source and destination addresses of the block to be moved. The BIOS builds the other four descriptors. The calling program supplies the segment limits, base address of the data blocks, and the access rights for each location.

INT 15h, Function AH = 88h: Read Extended Memory Size

BIOS

Reads the amount of memory above 1 megabytes from the CMOS RAM locations 30h and 31h.

On Entry:

AH = 88h

On Return:

AX = Number of contiguous 1 KB blocks of extended memory

INT 15h, Function AH = 89h: Switch to Protected Mode

BIOS

GDT describes the memory management environment in effect on return to the calling program. All table descriptors are initialized by the calling program except the BIOS code segment descriptor, with an access rights byte set to 9Bh. The address is set to F0000h and the segment limit is set to 0FFFFh. The DS, ES, and SS descriptors are loaded with 0018h, 0020h and 0028h, respectively. The descriptors built at these GDT offsets describe the segments that these registers reference after the BIOS returns to the calling program.

The program entering protected mode must construct its own IDT. This IDT must not overlap the BIOS real-mode table and must handle all interrupts while the program is in protected mode.

On Entry:

AH = 89h
BH = Interrupt number for IRQ 0
BL = Interrupt number for IRQ 8
ES:SI = Pointer to GDT built by application

On Return:

Carry = 0 = No error
AH = 00 = Successful move
1 = Error
AH = FFh = Unsuccessful

INT 15h, Function AX = 90h: Device Busy

BIOS

Invoked by BIOS. Default handler returns with AH set to zero and carry flag clear.

On Entry:

AH = 90h
AL = Device type:
- 00h = Fixed disk
- 01h = Diskette
- 02h = Keyboard
- 03h = Pointing device
- 80h = Network
- FCh = Fixed disk reset
- FDh = Diskette driver motor start
- FDh = Printer

ES:BX = Points to request block (if AL = 80h–FFh)

On Return:

Carry = 0 = No wait performed – driver must complete own wait
1 = Wait performed (I/O complete or timeout)

INT 15h, Function AX = 9002h: Pend On Keyboard

BIOS, 4000API

Invoked from the BIOS to indicate that it wants to pend until a key is pressed.

On Entry:

AX = 9002h

On Return:

None

▶ **NOTE:** *INTERRUPT ROUTINES MUST NOT ISSUE THIS CALL.*

INT 15h, Function AH = 91h: Interrupt Complete

BIOS

On Entry:

AH = 91h
AL = Device type
- 00h = Fixed disk
- 01h = Diskette
- 02h = Keyboard
- 03h = Pointing device
- 80h = Network
- FCh = Fixed disk reset
- FDh = Diskette driver motor start
- FEh = Printer

ES:BX = Points to request block if AL = 80h–FFh

On Return:

Carry = 0 = No wait performed Driver completes own wait
1 = Wait was performed (I/O complete or timeout)

INT 15h, Function AH = C0h: Return System Configuration Parameters Addr

BIOS

On Entry:

AH = C0h

On Return:

AH = 00h = Successful
86h = System model could not be determined
Carry = 0 = No error
1 = Error
ES:BX = Address of system configuration table

INT 15h, Function AH = C1h: Return Extended BIOS Data Area Segment

BIOS

On Entry:

AH = C1h

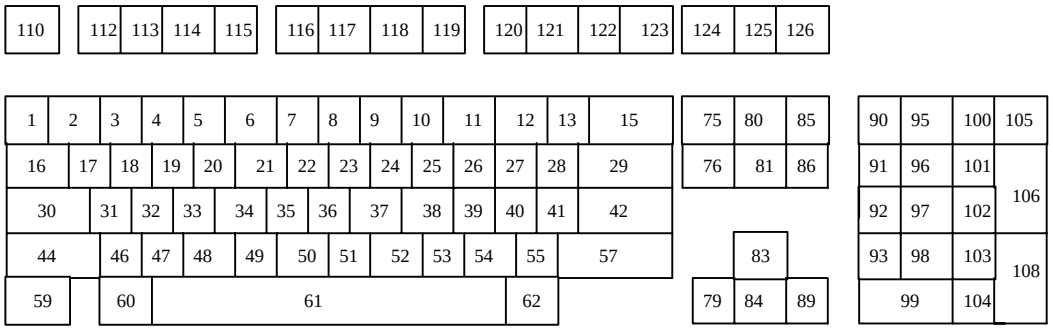
On Return:

AH = 00h = Successful
86h = No EBDA
Carry = 0 = No error
1 = Error (no EBDA)
ES = Segment address of EBDA

Keyboard Services: Interrupt 16h

Scan Codes

The following tables list the character codes for AT-compatible keyboards. The column in each of these tables labeled "Scan Codes In Hex" lists the codes offered to INT 09h by the keyboard controller through Port 60h. The first table shows the character codes returned in AX through the INT 16h standard functions 00h/01h. The second table shows the values returned in AX through the INT 16h extended functions 10h/11h. Scan codes containing the hidden key notation are used for the extended keys on the 101-style keyboards. Hidden key notation is defined as scan codes offered to INT 09h in the form E0-xx. Hidden key notation is not used for the 83/84-style keyboards. The key number system for the 101-key keyboard is defined in the following chart:



The states of the keys are defined as follows:

Pressing Caps Lock shifts alphabetical keys. All other keys defined to return normal values return the normal value indicated in the chart. Pressing a shift key while in the Caps Lock state causes alphabetical keys to return nonshifted values. All other keys defined to return shifted values return their shifted value. Pressing Caps lock a second time reverses the action.

Pressing Scroll Lock by itself causes no action other than the recording of the state in the shift flags. Applications must query the state of the flag to take advantage of Scroll Lock.

Pressing Num Lock causes the numeric keypad keys to return numeric values. Pressing a shift key while in the Num Lock state causes these same keys to return the normal values listed in the chart. Pressing Num Lock a second time reverses the action.

Holding the Alt down in combination with the numeric keypad keys lets you enter any ASCII code from 0 to 255 into the system. Register AH returns zero and register AL returns the entered code when the Alt key is released. If more than three digits are entered, the value returned is module-256.

Combination of Alt, Ctrl, and shift key presses are given the following priority: Alt, Ctrl, Shift. The only valid combination is Ctrl-Alt used for system reset.

Duplicate keys on the 101-style keyboard return the same 2-byte character code as their equivalent 83/84-style keyboard value. These duplicates states are recorded in the shift flags. Applications must query the shift flags to distinguish these duplicate keys.

Character Codes Returned by INT 16h, Functions 00h/01h

When a 101-style keyboard is used:

- Return all standard 83/84 style keyboard character codes, as is.
- Mask duplicate keys to the same 2-byte code as 83/84 style counterparts.
- Kill any character codes not compatible with the 83/84 key keyboard.

Table 6-6
Character Codes Returned by INT 16h, Functions 00h/01h

101 Key No.	U.S. Keyboard Legend	Scan Codes (Hex)	Unshifted (AH/AL)	Shifted (AH/AL)	Control (AH/AL)	Alt (AH/AL)
1	‘ ~	29	29/60	29/7e		
2	1 !	02	02/31	02/21		78/00
3	2 @	03	03/32	03/40	03/00	79/00
4	3 #	04	04/33	04/23		7A/00
5	4 $	05	05/34	05/24		7B/00
6	5 %	06	06/35	06/25		7C/00
7	6 ^	07	07/36	07/5E	07/1E	7D/00
8	7 &	08	08/37	08/26		7E/00
9	8 *	09	09/38	09/2A		7F/00
10	9 (	0A	0A/39	0A/28		80/00
11	0)	0B	0B/30	0B/29		81/00
12	– _	0C	0C/2D	0C/5F	0C/1F	82/00

Table 6-6 (Continued)
Character Codes Returned by INT 16h, Functions 00h/01h

101 Key No.	U.S. Keyboard Legend	Scan Codes (Hex)	Unshifted (AH/AL)	Shifted (AH/AL)	Control (AH/AL)	Alt (AH/AL)
13	= +	0D	0D/3D	0D/2B		83/00
15	Backspace	0E	0E/08	0E/08	0E/7F	
16	Tab	0F	0F/09	0F/00		
17	Q	10	10/71	10/51	10/11	10/00
18	W	11	11/77	11/57	11/17	11/00
19	E	12	12/65	12/45	12/05	12/00
20	R	13	13/72	13/52	13/12	13/00
21	T	14	14/74	14/54	14/14	14/00
22	Y	15	15/79	15/59	15/19	15/00
23	U	16	16/75	16/55	16/15	16/00
24	I	17	17/69	17/49	17/09	17/00
25	O	18	18/6F	18/4F	18/0F	18/00
26	P	19	19/70	19/50	19/10	19/00
27	[{	1A	1A/5B	1A/7B	1A/1B	
28	] }	1B	1B/5D	1B/7D	1B/1D	
29	\	2B	2B/5C	2B/7C	2B/1C	
30	Caps Lock	3A				
31	A	1E	1E/61	1E/41	1E/01	1E/00
32	S	1F	1F/73	1F/53	1F/13	1F/00
33	D	20	20/64	20/44	20/04	20/00
34	F	21	21/66	21/46	21/06	21/00
35	G	22	22/67	22/47	22/07	22/00
36	H	23	23/68	23/48	23/08	23/00
37	J	24	24/6A	24/4A	24/0A	24/00
38	K	25	25/6B	25/4B	25/0B	25/00
39	L	26	26/6C	26/4C	26/0C	26/00
40	;:	27	27/3B	27/3A		
41	”’	28	28/27	28/22		
42	(102 Key)	2B				
43	Enter	1C	1C/0D	1C/0D	1C/0A	
44	L Shift	2A				
45	(102 Key)	56				

Table 6-6 (Continued)
Character Codes Returned by INT 16h, Functions 00h/01h

101 Key No.	U.S. Keyboard Legend	Scan Codes (Hex)	Unshifted (AH/AL)	Shifted (AH/AL)	Control (AH/AL)	Alt (AH/AL)
46	Z	2C	2C/7A	2C/5A	2C/1A	2C/00
47	X	2D	2D/78	2D/58	2D/18	2D/00
48	C	2E	2E/63	2E/43	2E/03	2E/00
49	V	2F	2F/76	2F/56	2F/16	2F/00
50	B	30	30/62	30/42	30/02	30/00
51	N	31	31/6E	31/4E	31/0E	31/00
52	M	32	32/6D	32/4D	32/0D	32/00
53	,<	33	33/2C	33/3C		
54	.>	34	34/2E	34/3E		
55	/?	35	35/2F	35/3F		
57	R Shift	36				
58	L Ctrl	1D				
60	L Alt	38				
61	Space	39	39/20	39/20	39/20	39/20
62	R Alt	E0–38				
64	R Ctrl	E0–1D				
75	Insert	E0–52	52/00	52/00		
76	Delete	E0–53	53/00	53/00		
79	Left	E0–4B	4B/00	4B/00	73/00	
80	Home	E0–47	47/00	47/00	77/00	
81	End	E0–4F	4F/00	4F/00	75/00	
83	Up	E0–48	48/00	48/00		
84	Down	E0–50	50/00	50/00		
85	Page Up	E0–48	49/00	49/00	84/00	
86	Page Down	E0–51	51/00	51/00	76/00	
89	Right	E0–4D	4D/00	4D/00	74/00	
90	Num Lock	45				
91	7 Home	47	47/00	47/37	77/00	
92	4 Left	4B	4B/00	4B/34	73/00	
93	1 End	4F	4F/00	4F/31	75/00	
95	/	E0–35	35/2F	35/2F		
96	8 Up	48	48/00	48/38		

Table 6-6 (Continued)
Character Codes Returned by INT 16h, Functions 00h/01h

101 Key No.	U.S. Keyboard Legend	Scan Codes (Hex)	Unshifted (AH/AL)	Shifted (AH/AL)	Control (AH/AL)	Alt (AH/AL)
97	5	4C	4C/00	4C/35		
98	2 Down	50	50/00	50/32		
99	0 Ins	52	52/00	52/30		
100	*	37	37/2A	37/2A		
101	9 PgUp	49	49/00	49/39	84/00	
102	6 Right	4D	4D/00	4D/36	74/00	
103	3 PgDn	51	51/00	51/33	76/00	
104	. Del	53	53/00	53/2E		
105	–	4A	4A/2D	4A/2D		
106	+	4E	4E/2B	4E/2B		
108	Enter (pad)	E0–1C	1C/0D	1C/0D	IC/0A	
110	Esc	01	01/1B	01/1B	01/1B	
112	F1	3B	3B/00	54/00	5E/00	68/00
113	F2	3C	3C/00	55/00	5F/00	69/00
114	F3	3D	3D/00	56/00	60/00	6A/00
115	F4	3E	3E/00	57/00	61/00	6B/00
116	F5	3F	3F/00	58/00	62/00	6C/00
117	F6	40	40/00	59/00	63/00	6D/00
118	F7	41	41/00	5A/00	64/00	6E/00
119	F8	42	42/00	5B/00	65/00	6F/00
120	F9	43	43/00	5C/00	66/00	70/00
121	F10	44	44/00	5D/00	67/00	71/00
122	F11	57				
123	F12	58				
125	Scroll Lock	46				
126	Pause					

INT 16h, Function AH = 00h: Read Next ASCII Character

BIOS

If no key is in the buffer, pend until a key is pressed. This function must issue INT 15h, Function 90h (Device Busy) with AL set to 2 (Keyboard) to inform the OS that no key is available and that another task may be run. This function acts as an 83/84-style keyboard filter, masking duplicates and removing undefined 101 keys. Control is returned only when a key is available, and the key is removed from the keyboard buffer. Send a command to the keyboard to update the LED settings, to ensure that the keyboard LEDs match current flag settings.

This could create some PL/N compatibility issues since the scan code returned in the AH register in the 4000 BIOS is zero when AL is a valid ASCII character.

On Entry:
AH = 00h

On Return:
AX = Character code

INT 16h, Function AH = 01h: Set Zero Flag if Key Buffer Empty

BIOS

The key is not removed from the buffer. If the keyboard buffer does not contain a key, the zero flag is set. This function acts as an 83/84-style keyboard filter, masking duplicates and removing undefined 101 keys. Send a command to the keyboard to update the LED settings, to ensure that the keyboard LEDs match the current flag settings.

This could create some PL/N compatibility issues since the scan code returned in the AH register in the 4000 BIOS is zero when AL is a valid ASCII character.

On Entry:
AH = 01h

On Return:
Zero = Set flag if the keyboard buffer is empty
AX = Character code

INT 16h, Function AH = 02h: Read Shift Status

BIOS

On Entry:
AH = 02h

On Return:
AL = Shift status
- 7 = Insert state
- 6 = Caps Lock state
- 5 = Num Lock state
- 4 = Scroll Lock state
- 3 = Alt + Shift
- 2 = Ctl + Shift
- 1 = Left shift
- 0 = Right shift

INT 16h, Function AX = 0304h: Set Typematic Rates; Turn Off Key Repeat

4000API

On Entry:
AH = 03h
AL = 04h

On Return:
None

INT 16h, Function AX = 0305h: Set Typematic Rates; Set Key Repeat Timers

4000API

The default first repeat delay is 14, or about 0.77 second. The default repeat time is 3, or about 0.17 second

On Entry:
AH = 03h
AL = 05h
BH = First repeat delay in system timer ticks
BL = Repeat time in system timer ticks

On Return:
None

INT 16h, Function AX = 0306h: Set Typematic Rates; Turn On Key Repeat

4000API

On Entry:
AH = 03h
AL = 06h

On Return:
None

INT 16h, Function AH = 04h: Turn Keyclick Off or On

4000API

On Entry:
AH = 04h
AL = Click state
Bit values:
0 Turn key click off
1 Turn key click on

On Return:
None

INT 16h, Function AH = 05h: Put Key into Buffer as if from Keyboard

BIOS

On Entry:
AH = 05h
CH = Scan code
CL = Character

On Return:
AL = 0 if successful:
1 if buffer is full

INT 16h, Function AH = 10h: Read Next Extended ASCII Character

BIOS

If there is no key in the buffer, pend until a key is pressed. This function must issue INT 15h, Function 90h (Device Busy) with AL set to 2 (Keyboard) to inform the operating system that no key is available and that another task may be started. This function does not filter character codes for 83/84-style keyboard compatibility. Control is returned only when a key is available and the key is removed from the keyboard buffer. A command must be sent to the keyboard to update the LED settings, to ensure that the keyboard LEDs match the current flag settings.

On Entry:
AH = 10h

On Return:
AX = Character code

▶ **NOTE:** *There could be some PL/N compatibility issues, since the scan code returned in the AH register in the 4000 BIOS is zero when AL is a valid ASCII character – such as some of the ALT-character combinations. See the chart showing the character codes returned by INT 16h, in the* ***Standard Keyboard Interface*** *topic, on page 6-35.*

INT 16h, Function AH = 11h: Set Zero Flag if Extended Key Buffer Empty

BIOS

The key is not removed from the buffer. If the keyboard buffer does not contain a key, the zero flag is set. This function does not filter character codes for the 83/84-style keyboards. A command must be sent to the keyboard to update the LED settings, to ensure that the keyboard LEDs match the current flag settings. This could create some PL/N compatibility issues since the scan code returned in the AH register in the 4000 BIOS is zero when AL is a valid ASCII character.

On Entry:
AH = 11h

On Return:
Zero = Set flag if the keyboard buffer is empty
AX = Character code

INT 16h, Function AH = 12h: Read Extended Shift Status

BIOS

On Entry:

AH = 12h

On Return:

AH = Shift status
- 7 = Sys Req pressed
- 6 = Caps Lock state
- 5 = Num Lock state
- 4 = Scroll Lock state
- 3 = Right Ctl state
- 2 = Left Ctl state
- 1 = Left Alt state
- 0 = Right Alt state

AL = Shift status
- 7 = Insert state
- 6 = Caps Lock state
- 5 = Num Lock state
- 4 = Scroll Lock state
- 3 = Alt + Shift
- 2 = Ctl + Shift
- 1 = Left shift
- 0 = Right shift

INT 16h, Function AH = FEh: Swap Keyboard Translate Tables

4000API

On Entry:

AH = 0FEh

ES:BX = Address of new keyboard table as defined in the following table.
ES:BX = 0:0 causes the default table to be restored.

On Return:

None

If ES:BX = 0, install the default keyboard table.

This is a nonstandard function. Using this function affects the portability of the application to other PC-compatible platforms.

40-key translate table:

```
;unshifted 40 key table

db   'A',   'B',   'C',    'D',      'E',   'F',   'G'
db   'H',   'I',   'J',    'K',      'L',   'M',   'N'
db   'O',   'P',   'Q',    'R',      'S',   'T',   'U'
db   'V',   'W',   'X',    CLEAR,    '7',   '8',   '9'
db   -1,    -1,    -1,     EXIT,     '4',   '5',   '6'
db   -1,    -1,    -1,     LSHFT,    '1',   '2',   '3'
db   -1,    -1,    -1,     ESCAPE,   '0',   -1,    -1
```

```
;shifted 40 key table

db   TUP,   TDWN,   TLEFT,   TRIGHT,   '+',   '=',    '$'
db   '@',   '-',    '#',     '*',     '%',   '/',    '&'
db   ';',   ',',    ':',     '?',     '.',   TDEL,   LITE
db   ' ',   'Y',    'Z',     CLEAR,   '7',   '8',    '9'
db   -1,    -1,     -1,      EXIT,    '4',   '5',    '6'
db   -1,    -1,     -1,      LSHFT,   '1',   '2',    '3'
db   -1,    -1,     -1,      ESCAPE,  '0',   -1,     -1
```

25-key translate table:

```
;unshifted 25 key table

db   TUP,   TDWN,     TLEFT,   TRIGHT
db   -1,    '-',      '.',     EXIT
db   -1,    LSHFT,    TDEL,    CLEAR
db   -1,    '7',      '8',     '9'
db   -1,    '1',      '2',     '3'
db   -1,    ESCAPE,   ',0',    CR

;shifted 25 key table

db    TUP,   TDWN,     TLEFT,    LITE
db    -1,    '-',      '.',      EXIT
db    -1,    LSHFT,    TDEL,     CLEAR
db    -1,    '7',      '8',      '9'
db    -1,    '4',      '5',      '6'
db    -1,    '1',      '2',      '3'
db    -1,    ESCAPE,   '0',      CR
```

INT 16h, Function AH = FFh: Return Number of Keys on Default Keyboard

4000API

On Entry:

AH = 0FFh

On Return:

AL = Number of keys

This is a nonstandard function. Using this function affects the portability of the application to other PC-compatible platforms.

Timer and Real-Time Clock Services: Interrupt 1Ah

INT 1Ah, Function AH = 00h: Read System Timer Ticks

BIOS

Retrieves the values from the BIOS data areas 40:6Eh (high word) and 40:6Ch (low word). The value returned is the total number of clock ticks since midnight of system timer channel 0. Clock ticks occur 18.2 times per second and are processed by INT 08h. Interrupts are disabled while the timer tick is read to prevent update during access to the memory locations. The timer overflow flag, returned in AL, is reset to zero when this function is executed.

On Entry: AH = 00h

On Return:

CX = High word of system timer tick count
DX = Low word of system timer tick count
AL = Timer overflow flag
0 if day has not rolled over since last read
1 if timer value has exceeded 24 hours

INT 1Ah, Function AH = 01h: Set System Timer Ticks

BIOS

Sets the BIOS data values located at 40:6Ch and 40:6Eh to the data contained in CX and DX, respectively. It thereby sets the system timer tick count. Execution of this function clears the timer overflow flag at location 40:70h.

On Entry:

AH = 01h
CX = High byte of the system timer tick count
DX = Low byte of the system timer tick count

On Return:

None

INT 1Ah, Function AH = 02h: Read the Real-Time Clock Time

BIOS

Reads the hour, minute, second, and Daylight Savings option data from the CMOS RAM area of the MC 146818. Must verify that the update-in-progress bit is zero and disable interrupts before accessing the CMOS data.

On Entry:

AH =02h

On Return:

CH = Hours in BCD (0–23)
CL = Minutes in BCD (0–59)
DH = Seconds in BCD (0–59)
DL = Daylight Savings Time option
0 = No Daylight Savings Time
1 = Daylight Savings Time

INT 1Ah, Function AH = 03h: Set the Real-Time Clock Time

BIOS

Writes the hour, minute, second, and Daylight Savings option data to the CMOS RAM area of the MC 146818. It must wait for the update-in-progress bit to become zero and disable interrupts before accessing the CMOS data. Then, it must return with the carry flag set, if there was an error; otherwise, the carry flag must be cleared.

On Entry:

AH = 03h
CH = Hours in BCD (0–23)
CL = Minutes in BCD (0–59)
DH = Seconds in BCD (0–59)
DL = Daylight Savings Time option
0 = No Daylight Savings Time
1 = Daylight Savings Time

On Return:

None

INT 1Ah, Function AH = 04h: Read the Real-Time Clock Date

BIOS

Reads the century, year, month, and day from the CMOS RAM area of the MC 146818. It must wait for the update-in-progress bit to become zero and disable interrupts before accessing the CMOS data. Then, it must return with the carry flag set, if there was an error; otherwise, the carry flag must be cleared.

On Entry:

AH = 04h

On Return:

CH = Century in BCD (19 or 20)
CL = Year in BCD (0–99)
DH = Month in BCD (1–12)
DL = Day in BCD (1–31)

INT 1Ah, Function AH = 05h: Set the Real-Time Clock Date

BIOS

Writes the century, year, month, and day data to the CMOS data area of the MC 146818. It must wait for the update-in-progress bit to become zero and disable interrupts before accessing the CMOS data. Then, it must return with the carry flag set, if there was an error; otherwise, the carry flag must be cleared.

On Entry:

AH = 05h
CH = Century in BCD (19 or 20)
CL = Year in BCD (0–99)
DH = Month in BCD (1–12)
DL = Day in BCD (1–31)

On Return:

None

INT 1Ah, Function AH = 06h: Set the Real-Time Clock Alarm

BIOS

If the alarm is currently enabled, the function is not performed. This function writes the alarm hour, alarm minute, and alarm second data to the CMOS RAM area of the MC 146818 and sets the AIE bit in Register B.

This function must wait for the update-in-progress bit to become zero and disable interrupts before accessing CMOS data. Once activated, the alarm expires once every 24 hours until deactivated. When the alarm expires, INT 4Ah is issued.

If the calling program needs control passed to it, it must place the address of the interrupt handler in the vector for INT 4Ah. The user routine must save and restore all registers and preserve the state of the machine.

If the high two bits of the hour are set, an alarm expires every hour. If the high two bits of the minutes are set, the alarm expires every minute. If the high two bits of the seconds are set, the alarm expires every second.

The function must return with the carry flag set, if there was an error; otherwise, the carry flag must be cleared.

On Entry:

AH = 06h
CH = Hours in BCD (0–23)
CL = Minutes in BCD (0–59)
DH = Seconds in BCD (0–59)

On Return:

None

INT 1Ah, Function Ah = 07h: Reset the Real-Time Clock Alarm

BIOS

Clears any pending alarm specified by INT 1Ah, Function 06h by resetting the AIE bit in Register B.

On Entry:

AH = 07h

On Return:

None

INT 1Ah, Function AH = 09h: Read the Real-Time Clock Alarm

BIOS

Reads the alarm hour, alarm minute, and alarm second data and the AIE bit in Register B from the CMOS RAM area of the MC 146818. This function must wait for the update-in-progress bit to become zero and disable interrupts before accessing the CMOS data.

The function must return with the carry flag set, if there was an error; otherwise, the carry flag must be cleared.

On Entry:

AH = 09h

On Return:

CH = Hours in BCD (0–23)
CL = Minutes in BCD (0–59)
DH = Seconds in BCD (0–59)
DL = Alarm status
  0 = Alarm not enabled
  1 = Alarm enabled

Standard Mouse Interface: INT 33h

The following is a list of the common mouse functions and the corresponding actions. To pass commands to the driver, put the command into register AX and perform an interrupt call to INT 33h.

INT 33h, Function AX = 0000h: Mouse Reset and Status

MOUSE

Resets the mouse hardware and software; and returns the status.

On Entry:
AH = 00h
AL = 00h

On Return:
AX = 1 Successful
0 Failed
BX = Number of buttons

INT 33h, Function AX = 0001h: Show Cursor

MOUSE

No action taken.

On Entry:
AH = 00h
AL = 01h

On Return:
None

INT 33h, Function AX = 0002h: Hide Cursor

MOUSE

No action taken.

On Entry:
AH = 00h
AL = 02h

On Return:
None

INT 33h, Function AX = 0003h: Get Button Status and Mouse Position

MOUSE

Gets the current button status and reports the position of the mouse cursor.

On Entry:
AH = 00h
AL = 03h

On Return:
BX = 0 Pen is down
1 Button is down
CX = “X” coordinate of mouse cursor
DX = “Y” coordinate of mouse cursor

INT 33h, Function AX = 0004h: Set Cursor Position

MOUSE

Sets the position of the mouse cursor.

On Entry:

AH = 00h
AL = 04h
CX = "X" coordinate of mouse cursor
DX = "Y" coordinate of mouse cursor

On Return:

None

INT 33h, Function AX = 0005h: Get Button Press Information

MOUSE

Gets the current status information relating to pressing of the pen or button.

On Entry:

AH = 00h
AL = 05h
BX = 0 For pen status information
1 For button status information

On Return:

AX = Pen or button status
BX = Number of presses since last call
CX = "X" coordinate at last press
DX = "Y" coordinate at last press

INT 33h, Function AX = 0006h: Get Button Release Information

MOUSE

Gets the current status information relating to releasing of the pen or button.

On Entry:

AH = 00h
AL = 06h
BX = 0 For pen status information
1 For button status information

On Return:

AX = Pen or button status
BX = Number of releases since last call
CX = "X" coordinate at last release
DX = "Y" coordinate at last release

INT 33h, Function AX = 0007h: Set Minimum & Maximum x Cursor Position

MOUSE

Sets both the minimum and the maximum x coordinate of the cursor position.

On Entry:

AH =00h
AL = 07h
CX = Minimum cursor position
DX = Maximum cursor position

On Return:

None

INT 33h, Function AX = 0008h: Set Minimum & Maximum y Cursor Position

MOUSE

Sets both the minimum and the maximum y coordinate of the cursor position.

On Entry:

AH = 00h
AL = 08h
CX = Minimum cursor position
DX = Maximum cursor position

On Return:

None

INT 33h, Function AX = 0009h: Set Graphics Cursor Block

MOUSE

On Entry:

AH = 00h
AL = 09h

On Return:

None

INT 33h, Function AX = 000Ah: Set Text Cursor

MOUSE

No action taken.

On Entry:

AH = 00h
AL = 0Ah

On Return:

None

INT 33h, Function AX = 000Bh: Read Motion Counters

MOUSE

Reads the motion counters.

On Entry:

AH = 00h
AL = 0Bh

On Return:

CX = “X” mickey count (delta x since last call)
DX = “Y” mickey count (delta y since last call)

INT 33h, Function AX = 000Ch: Set Interrupt Subroutine Call Mask and Addr

MOUSE

Sets the call mask and address for the subroutine.

On Entry:

AH = 00h
AL = 0Ch
CX = Call mask
ES:DX = Subroutine address

On Return:

None

INT 33h, Function AX = 000Dh: Light Pen Emulation Mode On

MOUSE

On Entry:
AH = 00h
AL = 0Dh

On Return:
None

INT 33h, Function AX = 000Eh: Light Pen Emulation Mode Off

MOUSE

No action taken.

On Entry:
AH = 00h
AL = 0Eh

On Return:
None

INT 33h, Function AX = 000Fh: Set Mickey to Pixel Ratio

MOUSE

On Entry:
AH = 00h
AL = 0Fh

On Return:
None

INT 33h, Function AX = 0010h: Conditional Off

MOUSE

No action taken.

On Entry:
AH = 00h
AL = 10h

On Return:
None

INT 33h, Function AX = 0013h: Set Double-Speed Threshold

MOUSE

No action taken.

On Entry:
AH = 00h
AL = 13h

On Return:
None

INT 33h, Function AX = 0014h: Swap Interrupt Subroutines

MOUSE

Swaps interrupt mask and subroutine address for previous mask and subroutine address.

On Entry:

AH = 00h
AL = 14h
BX:DX = Subroutine address
CX = Call mask

On Return:

BX:DX = Previous subroutine address
CX = Previous call mask

INT 33h, Function AX = 0015h: Get Status Block Size

MOUSE

Gets size of status block.

On Entry:

AH = 00h
AL = 15h

On Return:

BX = Size

INT 33h, Function AX = 0016h: Save Driver Status

MOUSE

Saves status of mouse driver.

On Entry:

AH = 00h
AL = 16h

On Return:

ES:[DX] = Copy of status block

INT 33h, Function AX = 0017h: Restore Driver Status

MOUSE

Restores status of mouse driver.

On Entry:

AH = 00h
AL = 17h
ES:[DX] = Copied to status block

On Return:

None

INT 33h, Function AX = 0018h: Set Alternate Subroutine Call Mask and Addr

MOUSE

No action taken.

On Entry:

AH = 00h
AL = 18h

On Return:

None

INT 33h, Function AX = 0019h: Get User Alternate Interrupt Address

MOUSE

On Entry:
AH = 00h
AL = 19h

On Return:
AX = 0
BX = 0
CX = 0
DX = 0

INT 33h, Function AX = 001Ah: Set Mouse Sensitivity

MOUSE

No action taken.

On Entry:
AH = 00h
AL = 1Ah

On Return:
None

INT 33h, Function AX = 001Bh: Get Mouse Sensitivity

MOUSE

Gets the mouse sensitivity.

On Entry:
AH = 00h
AL = 1Bh

On Return:
BX = 50
CX = 50
DX = 50

INT 33h, Function AX = 001Ch: Set Mouse Interrupt Rate

MOUSE

No action taken.

On Entry:
AH = 00h
AL = 1Ch

On Return:
None

INT 33h, Function AX = 001Dh: Set Display Page Number

MOUSE

No action taken.

On Entry:
AH = 00h
AL = 1Dh

On Return:
None

INT 33h, Function AX = 001Eh: Get Display Page Number

MOUSE

Gets current active page number.

On Entry:
AH = 00h
AL = 1Eh

On Return:
BX = 0

INT 33h, Function AX = 001Fh: Disable Mouse Driver

MOUSE

Disables the mouse driver.

On Entry:
AH = 00h
AL = 1Fh

On Return:
AX = –1

INT 33h, Function AX = 0020h: Enable Mouse Driver

MOUSE

Enables the mouse driver. No return values.

On Entry:
AH = 00h
AL = 20h

On Return:
None

INT 33h, Function AX = 0021h: Software Reset

MOUSE

Resets the mouse software values.

On Entry:
AH = 00h
AL = 21h

On Return:
AX = –1
BX = 2

INT 33h, Function AX = 0022h: Set Language for Messages

MOUSE

No action taken.

On Entry:
AH = 00h
AL = 22h

On Return:
None

INT 33h, Function AX = 0023h: Get Language Number

MOUSE

On Entry:
AH = 00h
AL = 23h

On Return:
BX = 0

INT 33h, Function AX = 0024h: Get Driver Version, Mouse Type, and IRQ No.

MOUSE

Gets values for driver version, mouse type, and IRQ number.

On Entry:
AH = 00h
AL = 24h

On Return:
BX = Version
CH = 0
CL = IRQ

▶ **NOTE:** *For indexes to all of the interrupt definitions within this section (organized alphabetically by topic), refer to the* ***Interrupt Index****, at the end of this publication.*

Section 7

Reference, System Information

Introduction

This section contains system reference information, intended for supporting application programming efforts on the PEN*KEY® 6100 Computer. This includes: some information about the DOS operating system, the boot process, system messages, error codes, COM ports, I/O maps, IRQ maps, and system variables.

Topic Summary

List of Tables

NOTE: *Because this is an open system, your system configuration could differ somewhat from that shown in the following tables; for instance, you may have a custom installation. If you want to determine your exact configuration, refer to the .INI files in flash memory or in the 6100 Tool Kit. The IRQ and COM port mapping, for example, is defined in CARDID.INI. To verify this information, enter "CARDINFO /v" from the DOS screen, comparing that with the definitions in the CARDID.INI file.*

ROM DOS 5

The ROM DOS 5 product shipped in PEN*KEY 6100 flash memory is produced by Microsoft Corporation. It provides a number of configuration options that you can selectively choose when designing a DOS-based system. The primary difference between ROM DOS 5 and MS-DOS 5 is that much of the ROM DOS code executes from the upper memory address space, using read-only memory. There are several ways in which to load the code. In the 6100 Computer, the code is copied from flash into write-protected system memory, which is referred to as *shadow RAM*.

Since DOS is not loaded from disk, the IO.SYS and MSDOS.SYS files required for booting from a disk are not required in a ROM DOS-based computer. The initialization code normally loaded from disk is loaded from flash memory instead. CONFIG.SYS is the first file read from a disk during the ROM DOS boot process. Unless the CONFIG.SYS file contains a SHELL statement, this default shell statement is used: `SHELL=D:\COMMAND.COM /P D:\`

All the CONFIG.SYS statements that MS-DOS 5 supports are supported in ROM DOS 5. This includes the *`DOS=HIGH`* and *`DOS=UMB`* statements provided by MS DOS 5 to load data and DOS programs into the corresponding memory areas. For `DOS=HIGH` to be useful, an XMS driver, such as HIMEM.SYS, must be loaded. To use `DOS=UMB`, an XMS manager that supports the upper memory functions must be loaded. A simple XMS driver named ELANUMP.SYS is available for using shadow RAM as upper memory.

ELANUMP.SYS is much smaller and simpler than EMM386.EXE. Using shadow RAM instead of XMS memory to provide upper memory, it provides more overall memory to DOS and Windows applications. However, since ELANUMP.SYS cannot support expanded memory, EMM386.EXE is still the only method for providing expanded (EMS) memory.

The standard MS-DOS 5 version of COMMAND.COM is provided in flash. However, one change has been made to COMMAND.COM to make it more useful on the 6100 Computer, as described later in this section. If you prefer to use the unmodified version of COMMAND.COM, you can get it from the Tool Kit. Use a SHELL statement in CONFIG.SYS to override the default shell statement:

```
SHELL=A:\COMMAND.COM A:\ /P
```

The CONFIG.SYS file is processed before a command processor is selected. This particular CONFIG.SYS does not put any unique restrictions on file content.

All the messages displayed by COMMAND.COM have been shortened or split into multiple lines so that they fit the 20-column screen on the 6100 Computer.

- Error messages are displayed as numbers.
- The critical error handler displays a menu that includes numeric responses so that the user can respond without using an alpha keyboard. For example, “Abort, Retry, Fail?” becomes:
  2) Abort,
  1) Retry,
  3) Fail?

  These numbers are not always in a natural order because each line is displayed independently, pending on the options available. If the numbers were rearranged to look correct oneway, they would look wrong for another.
- The Echo command is modified to allow “ECHO,” to display a blank line instead of “Cannot load program.” In a related change, comma (,), equals (=), and quote (“) now terminates a keyword comparison.
- A bug with the GOTO is eliminated so that it works correctly with labels of two or more characters.

Boot Process

These paragraphs describe various methods that boot the 6100 Computer. Also described are the possible options during the boot process, which is done by holding a key down during the boot phase or by reconfiguring a CMOS setting. Topics covered include cold booting, warm booting, and the master-mode boot cycle.

Cold Booting

A cold boot cycle is usually never performed by the customer, unless the handheld computer is shipped with no backup battery or main battery. A *cold boot* is defined as a boot cycle in which power is applied for the first time. A cold boot is normally done only at the factory, when the batteries are installed. The only time you would perform a cold boot, is after a situation where the main battery pack has drained, no backup battery was installed, and no external charge is applied. These paragraphs describe the basic steps that occur during a cold boot.

BIOS Code is Shadowed

As one of the first steps, the BIOS copies itself, the video BIOS, and ROM DOS 5 into the first megabyte of system memory and then write-protects that memory. This is known as *BIOS shadowing*. The RAM used in this process is called *shadow RAM*. Shadow RAM is different from conventional memory in that a hardware mechanism prevents the shadow RAM from being modified once code is copied there. BIOS code is copied into shadow RAM primarily to enhance performance. Running from system memory is faster than running from flash memory directly.

Power-On Self-Tests (POSTs) are Run

A series of tests is performed on the hardware to ensure the system is functioning properly. Mostly, these tests are performed without indication that they are running. If a test fails, an error message is displayed on the screen or a series of beeps is sent to the speaker. The beep codes are used only for those errors that might occur before the screen can be enabled. For a description of all the beep codes that may be generated, see the *Audible Error Codes* table on page 7-8.

Video BIOS is Enabled

Early in the POST phase, the video BIOS is given control to initialize the screen. From there, errors are reported by messages on the screen rather than by beeps.

Version Messages are Displayed

Once the screen is enabled, version numbers are displayed for the ROM BIOS and any other hardware components that have version designations that can be read by software.

Detection of Cold Boots Using the CMOS Signature

A portion of CMOS memory contains a value known as a *checksum signature;* and indicates whether CMOS memory, as a whole, is valid. At the time of a cold boot, none of the memory in the system is initialized, including the CMOS memory. Consequently, because the signature is not correct, the ROM BIOS can determine that a cold boot has occurred and initialize CMOS and the associated real-time clock hardware to reflect a standard set of power-on defaults. It also records in EEPROM that a cold boot has occurred. Since the standard power-on defaults do not include a RAM drive, no RAM drive is present after a cold boot.

In standard PCs, the CMOS RAM is usually the only memory maintained by a backup battery. In contrast, all memory in the PEN*KEY computer is maintained at all times, with a fully charged backup battery. This memory can be maintained for approximately 200 hours in the event of a main battery failure. The CMOS RAM is special only because the PC-compatible software expects the CMOS RAM to exist and to contain certain configuration values.

Invalid RamDrive Message

One of the first tests involving extended memory verifies that the RAM drive, if it exists, is not corrupted by a power failure. If the RAM drive signature is corrupted, the RAM drive is removed and the default boot drive is set to drive A. The *Invalid RamDrive* message is then displayed and a single beep is issued.

Testing XMS Memory Message

This message displays while the POST of extended memory is performed. The initials *XMS* refer to the e**X**tended **M**emory **S**pecification published by Lotus, Intel, Microsoft, and AST Research. The XMS specification describes a program interface for accessing extended memory, the high memory area (HMA), and upper memory. You hear a clicking sound while the extended memory test is under way. One click occurs for each 64K segment of memory tested. Immediately following the extended memory test, a screen display indicates the quantity of memory that was found and tested. Any memory allocated to a RAM drive is not included in either the testing or reporting of available XMS memory.

Flash Memory Size Report

The size of the flash memory is reported on each boot cycle. There is no test associated with this display, but there is the possibility that an incorrect size could be reported if a hardware problem existed.

BIOS Extensions are Scanned For and Installed

After most of the POSTs have ended, the ROM BIOS scan through system memory in the address range of E800:0 through F7FF:F, looking for BIOS extensions. On PC-compatible computers, a BIOS extension is any software or firmware that runs prior to the time that the operating system is loaded. BIOS extensions execute directly from write-protected memory, such as ROM or shadow RAM. A special signature precedes each BIOS extension. After that signature is validated, the corresponding BIOS extension is made active (or *posted*). Currently the 6100 Computer includes the following BIOS extensions: the VGA BIOS (VGABIOS.BIN) and ROM DOS 5 (ROMDOSLO.BIN).

The VGA BIOS is located and executed prior to this stage of the boot process since it resides in the address range of C000:0 through CFFF:F. ROM DOS 5 itself is located at F000:0. That leaves 32 kilobytes of shadow RAM between E800:0 and EFFF:F that serves the same function as an option ROM socket in a traditional PC environment. If a BIOS extension signature is found at address E800:0, these 32 kilobytes of shadow RAM is scanned for BIOS extensions and remains resident as write-protected RAM. If no signature is found, the address range is redirected back to the ISA bus for use by PC Card adapters.

ROM DOS 5 is Booted

When the DOS 5 option ROM is posted (as a BIOS extension), it replaces interrupt 19h with a vector that points back into the ROM DOS 5 code. It then returns to the BIOS without making any other system changes. After the BIOS has finished scanning for all BIOS extensions, it issues an interrupt 19h to boot

the system. At this point, ROM DOS 5 displays its startup message and takes control of the system. The bootstrap code written for ROM DOS 5 prepares an environment very similar to the environment used by the disk-based MS-DOS 5 product. The primary difference is that much of ROM DOS executes directly from shadow RAM instead of from conventional memory. All DOS variables and a significant amount of code are still located at the traditional address of 70:0h.

The sole function of the ROM DOS BIOS extension is to load and execute ROM DOS directly from memory instead of loading it from a disk drive. The ROM DOS BIOS extension is enhanced to add the ability to select any of the primary DOS drives as the default boot drive rather than being limited to A: or C:.

Drives A through D are Initialized

One of the first interactions between ROM DOS 5 and the ROM BIOS is the initialization of DOS drives. Using interrupt 13h, the ROM BIOS supports two drives, as follows:

- Drive 0 = A:, the first PC Card slot. With the 6100 Computer display toward you, drive A: is toward the rear of the unit.
- Drive 1 = B:, the second PC Card slot (also used for Master Mode booting)

Refer to *Figure 1-1, Location of Reset Button and PC Card Drives* on page 1-17, in the *Getting Started* section, for the location of the PC Card slots.

A DOS 5 embedded device driver uses Interrupt 15h, Function 87h (move extended memory), to access both the RAM drive (drive C:) and the flash drive data (drive D:). ROM DOS 5 obtains the starting address for each drive by reading values stored in the BIOS data area of system memory.

Boot Drives Supported

Drives A through F are supported as boot drives. SRAM cards are recognized as drives A and B. ATA cards are recognized as drives E and F.

CONFIG.SYS is Loaded and Processed

Once ROM DOS has initialized, it reads and processes the CONFIG.SYS file on the default drive. By now, the default drive is chosen, by means of the boot selection process, discussed in the following paragraphs. If CONFIG.SYS does not exist, no error or warning is displayed.

See *Appendix A, Sample Configuration Files,* for example CONFIG.SYS files.

COMMAND.COM is Processed

The default command shell for ROM DOS 5 is:

```
SHELL=D:\COMMAND.COM /P D:\
```

After COMMAND.COM is loaded by this shell statement, AUTOEXEC.BAT is processed.

Drives Supported for Use

Note that drives E through H are supported, where previous drive letters (A and B) can be assigned drive letters in the range of E through H.

Warm Booting (or Resetting)

A *warm boot* is only slightly different from a *cold boot*. Since the two forms of booting are similar, this publication may refer to *booting* without specifying *warm* or *cold*, except in cases where the form of boot is important for the understanding of the topic.

For *warm booting*, the 6100 Computer must be booted before and still have power applied to it from either batteries or a charger. To warm boot, press the reset switch. This reinitializes the hardware and software. POST procedures are re-executed and ROM DOS restarts.

See *Figure 1-1, Location of Reset Button and PC Card Drives* on page 1-17, in the *Getting Started* section, for the location of the reset button.

If a RAM drive is formatted, the data on that drive is still there, unchanged by the reset cycle. The rest of memory is reconstructed as part of the tests performed. For example, the extended memory (which was *not* allocated to the RAM drive) is reset when extended memory is tested. The amount of extended memory reported does not include memory allocated to the RAM drive. If all extended memory is allocated to the RAM drive, the clicking associated with the extended memory test is not heard and 0K of extended memory is reported.

Master Mode Boot Sequence

A Master Mode boot is the process of booting from a PC Card rebooting and reflashing with a functioning version of flash, to replace the information in flash memory (reflashing), for a PEN*KEY 6000 Series hand-held computer that does not have a functioning version.

A Master Mode boot cycle differs from a standard boot cycle in that the BIOS code copied to shadow RAM comes from a PC Card instead of from flash memory. In addition, drive B: is temporarily used as the default drive. In that way, CONFIG.SYS and AUTOEXEC.BAT are executed from the PC Card instead of any startup files in flash memory or on the RAM drive.

The Master Mode boot process also affects the drive B: support provided by the ROM BIOS. Since drive B: now refers to the same data as the ROM drive (D:), drive B: is write-protected by the BIOS.

Boot Drive Selection

The 6100 Computer always boots to DOS from ROM, (that is, it never loads IO.SYS and MSDOS.SYS from a boot drive). CONFIG.SYS and AUTOEXEC.BAT files are read from the default drive whenever the files are present. The default drive used to load these startup files is controlled by a CMOS variable. Two bits of this variable define a drive that could possibly be used as the default drive. One bit determines whether a Master Mode boot is performed. If the Master Mode boot bit is set, the other two bits are ignored and the default drive is B:. If the Master Mode bit is not set, the other two bits define a drive that could *possibly* be used as the default drive.

Before a default drive is selected, a drive-ready test is performed. If media is present and the drive is ready, it becomes the default drive. Note that this test applies only to drives A: and B:, since they are the only drives associated with removable media. When the default boot drive is not ready, the ROM drive is selected as the default drive. To obtain the ROM DOS boot-up menu, press and release the [Suspend/Resume] key after resetting the unit.

Use the “Start from” menu to override the CMOS setting by manually selecting the drive that is the default drive. This menu is available even during a Master Mode boot. The drive selected at the “Start from” menu is still tested to ensure that it is ready before that drive is selected as the default drive. If the selected drive is not ready, the flash drive (D:) is used instead of the selected drive. See *Setup for PC Development* in the *Getting Started* section for more information on the “Start from” menu.

System Messages

Table 7-1
System Messages

Display Message	Meaning	Response
PEN*KEY BIOS Version: {n.nn}	---	
Testing PC Card	---	None (status comment)
Master Mode Boot	PC Card is used in place of flash	
Testing XMS Memory..	Performs extended memory tests	
{nnnn} KB Extended	Reports number of extended memory KB tested	
{nnn} KB FLASH	Reports number of flash memory KB found	
8 KB Cache Ok	Reported on 486 systems	
Loading MS-DOS 5 ROM U.S. Patent #4955066 Norand Release {n.nn}	Reports loading MS-DOS from ROM and shows revision level	
Invalid CMOS	Memory in the 6100 Computer is cleared, including the RAM drive.	An application download is required unless the application was not stored on the RAM drive.
Invalid RamDrive	A power failure corrupted the RAM drive but left CMOS intact. This occurs when swapping batteries and backup battery is bad or missing.	An application download is required unless the application was not stored on the RAM drive.
Hot Int. Failure	Hot interrupt failure	Error: Call the Customer Support Center at 800-755-5505 (U.S.A. or Canada) or 425-356-1799.
IC1 Failure	---	
IC2 Failure	---	
Master DMA Failure	---	
Slave DMA Failure	---	
DMA Page Reg. Failure	DMA page register failure	
Timer0 Failure	---	
Timer2 Failure	---	
Timer Tick Failure	---	
Reset Code Failure	---	
Clock Failure	---	
PC CARD Failure	PC Card controller failure	
{nnnn} Option ROM Failure	Option ROM signature found for a block of shadow RAM that did not have a valid checksum	If building your own custom flash, you may need to reorganize the storage of the files in flash.
{nnnn} Keyboard Failure	Keyboard controller failure	Error: Call the Customer Support Center at 800-755-5505 (U.S.A. or Canada) or 425-356-1799.
{nnnn} Memory Failure	Memory test failure	
COM1	Serial Port 1 failure	
COM2	Serial Port 2 failure	
COM3	Serial Port 3 failure	
Memory Test Failure	Extended Memory test failure	
No FLASH Memory	Flash identification, sizing failed	

Audible Error Codes

There are two different kinds of beeping that can occur when the unit is rebooted. This is the result of a two-stage process that occurs during a reboot. The primary set of beep codes all have the same frequency and duration. The number of beeps generated is what identifies the code. The secondary set of beep codes consists of at least one long, one medium long, and one short beep. Table 7-2 shows these error codes.

Table 7-2
Audible Error Codes

Duration / No. of Beeps			
Long	**Medium**	**Short**	**Meaning**
			POST and Runtime Power Notification
0	0	1	The main battery is too low to start the system.
0	0	4	The PC Card controller fails power-on diagnostics.
0	0	5	The PC Card controller fails to respond.
			POST (only)
1	1	2	Unable to read clocks for peripherals.
1	1	4	The ROM BIOS checksum failed.
1	3	1	RAM Refresh is not active.
1	3	3	The first 64K of system RAM is not operating correctly.
3	4	3	The LCD cannot be enabled.
4	2	3	Cannot enable A20 using port 64h (see NOTE) (standard keyboard port).
4	2	4	Cannot enable A20 using port 92h (see NOTE) (IBM fast A20 port).
4	2	5	Cannot enable A20 using port 0EEh (see NOTE) (special fast A20 port).
4	4	2	Cannot copy BIOS into shadow RAM.

▶ **NOTE:** *A20 refers to the special PC/AT hardware that enables or disables address line number 20 on the system memory address bus. The A20 line can be disabled to maintain compatibility with 8088-based hardware platforms.*

Hardware Ports and Memory Maps

The following IRQ and I/O maps are valid for 6100 Computer.

Hardware Ports

The system supports the following hardware ports:

Table 7-3
Hardware Ports

COM Port	Address	IRQ	Description
COM1	3F8h	4	8-pin connector on bottom, accessed through the dock.
COM2	2F8h	3	Serial lid, MSR, or scanner pod. Pod can be radio, bar code scanner.
COM3	3E8h	14	IrDA port; also accessed through the 25-pin dock connector.
COM4	(see NOTE)	(see NOTE)	PC Card modem (if installed and supported by PC Card drivers).

▶ **NOTE:** *Addresses and IRQs allocated according to PC Card driver. The default Address = 2E8h and the default IRQ = 5, using NORMOD.SYS and CardSoft.*

Table 7-4
COM1 Connector Pin-Outs

9-pin Connector	25-pin Connector	Description
2	2	TX
3	3	RX
5	7	SG
7	4	RTS
8	5	CTS
4	20	DTR (see NOTE below)
6	6	DSR (see NOTE below)

▶ **NOTE:** *DTR is looped back to DSR, both at dock and at internal UART. This means there is no DTR or DSR connection from the UART to the modem.*

IRQ and Other Hardware Interrupts

The system supports the following hardware interrupts:

Table 7-5
Hardware Interrupts

Vector	Type	Usage
00h	processor	Divide error
01h	processor	Debug exceptions (e.g., single step)
02h	processor	Nonmaskable interrupt
03h	processor	Breakpoint
04h	processor	Overflow trap
05h	processor	Bounds exception
07h	processor	Invalid opcode
08h	IRQ0	Timer interrupt handler
	processor	Double exception
09h	IRQ1	Keyboard interrupt handler
	processor	Coprocessor segment overrun
0Ah	IRQ2	Slave (cascades IRQ8 – IRQ15)
	processor	Invalid task state segment
0Bh	IRQ3	Serial port 2 (COM2) (serial lid or scanner)
	processor	Segment not present
0Ch	IRQ4	Serial port 1 (COM1)
	processor	Stack fault
0Dh	IRQ5	COM4 (PC Card modem)
	processor	General protection error
0Eh	IRQ6	Reserved for PC Card – (card status change)
	processor	Page fault
0Fh	IRQ7	BATTERY DATA
10h	processor	Coprocessor error
70h	IRQ8	Real-time clock periodic interrupt
71h	IRQ9	Reserved for PC Card
75h	IRQ13	Reserved for PC Card
76h	IRQ14	COM3 (IrDA port)
SMI		System management interrupt

I/O Map

Table 7-6
I/O Address and Devices

I/O Address	Device
000 – 00F	8237 DMA controller #1
020 – 021	8259 interrupt controller #1
022 – 023	ELAN configuration registers
040 – 043	8253 system timer
060	Keyboard
061	Port B
064	Keyboard
070 – 071	Real-time clock and NMI mask
080	POST debug port
081 – 08F	DMA page registers
092	PS/2 port 92
0A0 – 0A1	8259 interrupt controller #2
0C0 – 0DF	8237 DMA controller #2
108 – 16F	Reserved for PC Card general use
170 – 177	Reserved (secondary hard disk controller)
178 – 1EF	Reserved for PC Card general use
1F0 – 1F7	Reserved for PC Card (primary ATA/IDE disk controller)
1F8 – 1FF	Reserved for PC Card general use
200 – 207	Touch screen clock
210 – 21f	Reserved for PC Card general use
220 – 224	Reserved for SST radio
225 –237	Reserved for PC Card general use
238 – 23F	Reserved (bus mouse)
240 – 277	Reserved for PC Card general use
278 – 27F	Reserved (Parallel Port; i.e., LPT1)
280 – 2E7	Reserved for PC Card general use
2E8 – 2EF	Reserved (PC Card / COM)
2F0 – 2F7	Reserved for PC Card general use
2F8 – 2FF	COM 2 (serial lid or scanner)
320 – 35F	Reserved for PC Card general use
378 – 37F	Reserved (PC Card/Parallel port)
380 – 38F	Reserved (PC Card/SDLC)
390 – 29F	Reserved for PC Card general use
3A0 – 3AF	Reserved (PC Card/SDLC)
3B0 – 3BF	Reserved (MDA/EGA/VGA)
3B0 – 3DF	VGA/CGA
3E8 – 3EF	COM3 (IrDA port)
3F6 – 3F7	Reserved for PC Card (primary ATA/IDE disk controller)
3F8 – 3FF	COM 1

BIOS/CMOS System Variables

ROM BIOS is system software at the lowest layer of a PC operating environment. ROM BIOS provides a hardware abstraction level in a system so that higher levels of software need not be concerned with certain hardware details. Occasionally, it is helpful to examine the ROM BIOS or at least look at the data that it uses to get its job done. The following are lists of data areas on which the ROM BIOS operates.

▶ **NOTE:** *A full description of each variable used by BIOS is beyond the scope of this guide.*

ROM BIOS Data Area

A PC-compatible ROM BIOS uses system variables stored, starting at location 40:0h. This address starts the first block of data immediately following the interrupt vector table. A *de facto* industry standard defines most of the variables on which a ROM BIOS must operate. In the following table, the entries unique to the NORAND® BIOS are ***italicized and bold***. The rest are defined in accordance with the industry standard.

Table 7-7
BIOS Data in System RAM

Addr	Size	Description
40:00	4 words	I/O address of up to four asynchronous communications adapters (COM1–COM4)
40:08	3 words	Reserved: I/O address, up to 3 printer adapters (LPT1 LPT3)
40:0E	word	Segment address of extended BIOS Data Area
40:10	word	Equipment status word (Returned by interrupt 11h)
40:12	byte	Reserved: POST status
40:13	word	Conventional memory size in kilobytes (returned by INT12h)
40:15	***word***	***Pass/Fail indicators for RAM POST test***
40:17	word	Keyboard shift flags
40:19	byte	Alt-keypad accumulator
40:1A	word	Pointer to next keycode in keyboard buffer
40:1C	word	Pointer to next available location to save a keycode in the keyboard buffer
40:1E	16 words	Circular keyboard buffer
40:3E	byte	Reserved: diskette recalibrate status
40:3F	byte	Reserved: diskette motor status
40:40	byte	Reserved: diskette master time-out control
40:41	byte	Diskette status return code—status of last RAM card access
40:42	7 bytes	Reserved: diskette controller status bytes
40:49	byte	Video mode setting
40:4A	word	Number of video columns in memory—not all columns are represented onscreen
40:4C	word	Current page size (in bytes)
40:4E	word	Current page address
40:50	8 words	Cursor position for each page
40:60	word	Cursor display mode (start/end line)
40:62	byte	Current display page
40:63	word	I/O address of display hardware
40:65	byte	Current mode select register
40:66	byte	Current palette value

Table 7-7 (Continued)
BIOS Data in System RAM

Addr	Size	Description
40:67	dword	Segmented address of video option ROM
40:6B	byte	Reserved: the last interrupt that occurred
40:6C	word	Time of day counter, low word
40:6E	word	Time of day counter, high word
40:70	byte	Timer overflow indicator
40:71	byte	System break flag
40:72	word	System reset flag
40:74	byte	Reserved: last hard disk status
40:75	byte	Number of fixed drives
40:76	byte	Reserved: fixed disk control byte
40:77	byte	Reserved: fixed disk port offset
40:78	4 bytes	Reserved: printer time-out table
40:7C	4 bytes	Serial time-out table
40:80	word	Offset to start of keyboard buffer
40:82	word	Offset to end of keyboard buffer
40:84	***byte***	***Number of rows on screen(24/25)***
40:85	***word***	***Character height (bytes/char)***
40:87	***byte***	***video control bit flags***
40:88	***byte***	***Video feature bit flags***
40:89	***byte***	***VGA control bit flags***
40:8A	***byte***	***Display combo code table index***
40:8B	byte	Reserved: diskette data rate info
40:8C	byte	Reserved: fixed disk status register
40:8D	byte	Reserved: fixed disk error register
40:8E	byte	Reserved: fixed disk interrupt flag
40:8F	byte	Reserved: diskette controller information
40:90	byte	Media type for drive 0
40:91	byte	Media type for drive 1
40:92	byte	Media type for drive 2
40:93	byte	Media type for drive 3
40:94	2 bytes	Reserved: current track for each drive
40:96	byte	Keyboard status byte
40:97	byte	Keyboard LED status byte
40:98	dword	Segmented pointer to user wait flag
40:9C	dword	Wait time-out data word
40:A0	byte	Wait active flag
40:A1	39 bytes	Reserved
40:C8	10 bytes	Reserved for real mode restart stack—***used by RESTART***
40:D2	word	RS-232 time-out counter—used by interrupt 14h
40:D4	***4 bytes***	***Reserved—used by POST***
40:D8	***20 words***	***Reserved—for NORAND BIOS***

CMOS Registers

Registers in CMOS RAM are provided as part of a real-time clock interface. These registers are accessed by writing a register number to port 70h, then either reading or writing port 71h to access the contents of the specified register. The first few registers are real-time clock registers. In the list that follows, Norand-specific register definitions are ***italicized and bold***. The remaining registers are defined by PC compatibility standards.

Table 7-8
CMOS Register Assignments

Register Name	Assignment
RTC_SECONDS	equ 0
RTC_SECONDS_ALARM	equ 1
RTC_MINUTES	equ 2
RTC_MINUTES_ALARM	equ 3
RTC_HOURS	equ 4
RTC_HOURS_ALARM	equ 5
RTC_WEEKDAY	equ 6
RTC_DAY	equ 7
RTC_MONTH	equ 8
RTC_YEAR	equ 9
RTC_REG_A	equ 10
RTC_REG_B	equ 11
RTC_REG_C	equ 12
RTC_REG_D	equ 13
RTC_CENTURY	equ 032h
CMOS_DIAGS	equ 0Eh
CMOS_SHUTDOWN	equ 0Fh
SD_NORMAL_RESET	equ 0
SD_REAL_MODE_ENTRY	equ 1
SD_BOOTSTRAP	equ 4
SD_EOI_JMP	equ 5
SD_FAR_JMP	equ 6
SD_BLOCK_MOVE1	equ 7
SD_MEM_TEST	equ 8
SD_BLOCK_MOVE2	equ 9
SD_FAR_JMP2	equ 10
SD_IRET	equ 11
SD_FAR_RET	equ 12
CMOS_DISKETTE1_TYPE	equ 10h
CMOS_DISKETTE2_TYPE	equ 11h
CMOS_FDISK_TYPE	equ 12h
CMOS_EQUIPMENT	equ 14h
CMOS_BASE_MEMLO	equ 15h ; conventional memory size
CMOS_BASE_MEMHI	equ 16h
CMOS_XTD_MEMLO	equ 17h ; extended memory size
CMOS_XTD_MEMHI	equ 18h
CMOS_FDISK1_TYPE	equ 19h

Table 7-8 (Continued)
CMOS Register Assignments

Register Name	Assignment
CMOS_FDISK2_TYPE	equ 1Ah
CMOS_CHKSUM_HI	equ 2Eh ; checksum for 10h–2Dh
CMOS_CHKSUM_LO	equ 2Fh
CMOS_POST_XTD_MEMLO	equ 30h ; posted extended memory size
CMOS_POST_XTD_MEMHI	equ 31h
CMOS_DOS_FLAGS	equ 3Fh ; ROM DOS boot flags
bMasterModeBoot	equ 01h ; MasterMode boot bit
bDefaultDrive	equ 06h ; 00=A:, 01=B:, 10=C:, 11=D:
CMOS_FLASH_INFO	equ 40h
FLASH_1_512K	equ 00h ; Bit values define flash size
FLASH_2_512K	equ 01h
FLASH_3_512K	equ 02h
FLASH_4_512K	equ 03h
bATMEL	equ 40h ; Set only for Atmel Flash type
bAMD	equ 80h ; Set only for AMD flash type

Section 8

Reference, Open Systems Publications

Introduction

This section contains reference information consisting of publications that are referenced from previous sections, or publications that may be good references for application development activity. This includes Application API, DOS 5.0 API, and Hardware Interface publications.

▶ **NOTE:** *All publications from Intermec Technologies Corporation should be ordered through your local Account Executive.*

Application API Publications

- ***APM BIOS Interface Specification 1.1***
  Intel Corporation
  Literature Distribution Center
  P.O. Box 7641
  Mt. Prospect, IL 60056-7641
  (800) 548-4725
  Publication Order Number: 241704-001
- ***Handwriter Recognition System for Windows User's Guide***
  Communication Intelligence Corp. (CIC)
  Can be ordered through your Account Executive
  P/N: 961-054-001
- ***Handwriter Recognition System for Windows Release Notes***
  Communication Intelligence Corp. (CIC)
  Can be ordered through your Account Executive
  P/N: 961-054-002
- ***PenDOS V2.21, Handwriter Recognition System User's Guide***
  Communication Intelligence Corp. (CIC)
  Can be ordered through your Account Executive
  P/N: 961–054–003
- ***PenDOS V2.21, Handwriter Recognition System Release Notes***
  Communication Intelligence Corp. (CIC)
  Can be ordered through your Account Executive.
  P/N: 961–054–004
- ***Handwriting Recognition System, HR-1200, User Manual***
  Synaptics
  Can be ordered through your Account Executive
  P/N: 961–054–007
- ***Microsoft Developers CD***
  A technical reference for developers; published quarterly on CD. To obtain information contact:
  Microsoft Developers Network
  One Microsoft Way
  Redmond, WA 98052–6399

- ***PCMCIA Standards Organization***
  Personal Computer Memory Card International Association
  1030 East Duane Avenue, Suite G
  Sunnyvale, CA 94086
  ISBN: 408-720-0107

- ***Ralf Brown's Interrupt List, release 39 (last updated 2/6/94)***
  This public domain online reference can be obtained in a number of ways. Below is a list of some of the access locations and methods available for obtaining this list:
  On the Internet, by standard anonymous FTP from FTP.CS.CMU.EDU [128.2.206.173].

  - Change directory to:

    `/cs.cmu.edu/user/ralf/pub`

  - Then get the files:

    `inter??a.zip through inter??d.zip.`

  You *must* change directory in a single command because of the way CMU's anonymous FTP works. Do not forget to set mode to "binary" or "type L 8"! ZIP unarchivers for MS-DOS and Unix are available in the "archivers" subdirectory.

  If connected to AFS, you can perform standard Unix/VMS/whatever directory listing and copy files from the above directory.

  On FIDOnet, from SoundingBoard BBS 1:129/26 1-412-621-4604 9600/14.4k HST/V32 as files INTERrrA.ZIP to INTERrrD.ZIP (rr stands for the release number), in file area #8. First-time callers may download.

  Alternative Distribution Points (the list is typically available within 24 hours of release):

  - Internet:
    On the SimTel Software Repository mirrors as files `inteRRRA.zip` to `inteRRRD.ZIP` in directory `/pub/msdos/info`, where *RR* stands for the release number. Note that you must use a binary transfer mode ("tenex" or "type L 8" for most people) to successfully FTP the files. The SimTel mirrors include

    ```
    oak.oakland.edu [141.210.10.117]
    wuarchive.wustl.edu [128.252.135.4]
    ftp.uu.net [137.39.1.9]
    nic.funet.fi [128.214.6.100]
    src.doc.ic.ac.uk [146.169.3.7]
    archie.au [139.130.4.6]
    ```

  - FIDO:
    SyncPoint BBS 1:261/1008 1-410-529-2584 File Requests.

  Additional Distribution Points:

  - BITnet:
    You may retrieve the copy on SimTel via the following automated mail servers:

    - In the US–
      `LISTSERV@RPITSVM` (alias VM.ITS.RPI.EDU)
      `LISTSERV@NDSUVM1` (alias VM1.NODAK.EDU)

- In Europe–

Austria	TRICKLE@AWIWUW11
Belgium	TRICKLE@BANUFS11
Denmark	TRICKLE@DKTC11
France	TRICKLE@FRMOP11
Germany	TRICKLE@DEARN
Italy	TRICKLE@IMIPOLI
Netherlands	TRICKLE@HEARN
Spain	TRICKLE@EB0UB011
Sweden	TRICKLE@SEARN
Turkey	TRICKLE@TREARN

- Elsewhere–

Columbia	TRICKLE@UNALCOL ()
Israel	TRICKLE@TAUNIVM ()

- FIDO:
  Boards belonging to the PDN (Programmer's Distribution Network) system
- CompuServe:
  In the IBM Programming Forum (GO IBMPRO), Library 6, as `INTrrA.*`, `INTrrB.*`, `INTrrC.*`, and `INTrrD.*`.

The list is also posted to USEnet in `comp.binaries.ibm.pc` about twice a year, concurrently with a new release of the list. Since `comp.binaries.ibm.pc` is archived, you should be able to find a fairly recent release in the various UUCP archives.

- ***System BIOS for IBM PC's, Compatibles, and EISA Computers, Second Edition***
  Phoenix Technologies LTD.
  Addison–Wesley Publishing Company
  ISBN: 0–201–57760–7
- ***Windows Internals***
  Matt Pietrek
  Addison–Wesley Publishing Company
  ISBN: 0–201–62217–3
- ***Windows SDK (Microsoft, Borland)***
  Visual Basic Reference
- ***Writing TCOM Modules in PL/N***
  Intermec Technologies Corporation
  P/N: 541–002–523

DOS 5.0 API

- ***PC Interrupts***
  Ralph Brown and Jim Ryle
  Addison-Wesley Publishing
  ISBN: 0–201–57797–6

Hardware Interface

- ***Intel Peripheral Components 1991***
  Intel Literature Sales
  P.0. Box 7641
  Mt. Prospect, IL 60056–7641
  ISBN: 1–55512–127–6
- ***Motorola Microprocessor, Microcontroller, and Peripheral Data Volume II DL139***
  Motorola Literature Distribution
  P.O. Box 20912
  Phoenix, AZ 85036
- ***PCMCIA Controller Data Book***
  CL–PD6710/PD6720 Advanced Data Book
  Cirrus Logic, Inc.
  3100 West Warren Ave.
  Fremont, CA 94538
  ISBN: 510–623–8300

Appendix A

Sample Configuration Files

Introduction

This section contains sample configurations for the PEN*KEY® 6100 Hand-Held Computer, as summarized below.

Topic Summary

Sample Boot Configurations Files

AUTOEXEC.BAT (Default)

▶ **NOTE:** *The AUTOEXEC.BAT and CONFIG.SYS files shown in this section are configured to go together. These files found in softcopy, in the Tool Kit, are almost exactly like these. If one of these examples is copied, remember these two files need to be configured to work together.*

```
@echo off
Rem Remap keys from BIOS default for new overlay
d:\keymap.exe d:\keys.ini > nul

path=d:\;d:\cs;d:\acn
if “%1” == “” autoexec.bat E: F:

:checkcard
shift
if “%0” == “” goto loadtsrs
Rem DIRCNT.EXE returns 0 if bad drive, or file count + 1.2 will be returned
```

```
Rem AUTOEXEC.BAT is found on a drive.
dircnt.exe %0\autoexec.bat >nul:
if not errorlevel 2 goto checkcard
%0
autoexec.bat %0

:loadtsrs
Rem 4000api is needed by psrom0c to run acnnpcp
LH d:\4000api.exe /PC /10 /16 /C3
LH d:\mininet.exe –s1152 –t0 –c1
LH d:\vrotate.exe 0 0 240 320
LH d:\share.exe /f:3072 /l:128
LH d:\fontsel.exe 1

:romutils
d:\psrom0c.exe –i
if errorlevel 1 goto exit
Rem Errorlevel 0 indicates that psrom0c.exe exited to drive e: to allow
REM executing autoexec.bat
if exist autoexec.bat autoexec.bat
goto romutils

:exit
@echo on
```


EXAMPLE: Sample AUTOEXEC.BAT for Windows
This is a DOS code example that could set up Windows via AUTOEXEC.BAT.

```
@ECHO OFF
REM Disable VROTATE if existed
d:\vrotate.exe –D

REM Disable Video Accesses as Activity
d:\elancfg /V0 /H16 /L1 /D4 /T2 /C0
rem /V0 – Disable Video Accesses as Activity
rem /H16 – High Speed to Low Speed set to 16 seconds
rem /L1 – Low Speed to DOZE set to 16 seconds
rem /D4 – Doze to Suspend set to 4
rem /T0 – Extended timer ticks in DOZE
path=\;d:\;\windows
cd \
911
```

CONFIG.SYS (Default)

```
Rem Display the Flash version message for 15 seconds
device=d:\delay.exe “” “PEN*KEY 6100 FLASH 61FL1000 V1.16” /1000
break=on
buffers=30
files=128
lastdrive=z
stacks=9,256

device=d:\himem.xyx /machine:2
Rem Do not load dos=high, because it causes DOS to use High Memory Area,
Rem which prevents creating a maximum RamDrive.
Rem dos=high
device=d:\elanump.sys /X=C000
dos=umb

devicehigh=d:\elanapm.exe
devicehigh=d:\nordospm.exe
devicehigh=d:\clock.exe

Rem Norand Card and socket services drivers:
devicehigh=d:\norcs\elancsss.exe
Rem /r means auto-retry after suspend
devicehigh=d:\norcs\atabios.sys /r
devicehigh=d:\norcs\norata.sys
```

```
Rem /c4 means install as COM4
Rem /s means save and restore UART values for COM4 on SUSPEND/RESUME
Rem /d m n t means delay t ticks where m=man code and n=man info
Rem 16b 21 refers to the Erickson PIA radio
devicehigh=d:\norcs\normod.sys /c4 .s /d 16b 21 20

Rem Delay for two seconds to ensure ELANCSSS has recognized the cards
device=d:\delay.exe /100 “
devicehigh=e:\stacker.drv /p=1 e: f:
devicehigh\f:\stacker.drv /p=1 f:

shell=d:\command.com d:\ /e:512 /p
```

EXAMPLE: Sample CONFIG.SYS
At a minimum, the following statements should be included:

```
shell=d:\command.com d:\ /p
device=d:\elanapm.exe
device=d:\nordospm.exe
```

EXAMPLE: Another Sample CONFIG.SYS
The following is another sample of a CONFIG.SYS file:

```
shell=d:\command.com d:\ /e:512 /p
device=d:\himem.sys
devicehigh=d:\elanump.sys /x=C000,D400,D800,DC00,E000
device=a:\emm386.exe i=D400–EFFF FRAME=D400
devicehigh=d:\elanapm.exe
devicehigh=d:\nordospm.exe
dos=high,umb
break=on
buffers=30
files=50
```

Other Sample Configuration Files

PENWIN.INI

The following is an example for use with Windows 3.1 on a 6100 Computer:

```
[Current]
User=User 1
InkWidth=1
InkColor=0
SelectTimeout=500

[*User 1]
TryDictonary=100
ErryrLevel=25
EndRecognition=8000
TimeOut=500
WriteDirection=103
MenuDropAlignment=0
Preferences=0
IntlPreferences=0
Recognizer=mars.dll

[Dictionary List]
MAINDICT.DLL=

[Recognizer List]
MARS.DLL=

[MsSpell]
MSSPELL.DLL=

[MsMainDict]
enuMain=
```

```
[User List]
User 1=

[sysges]
C!=xx,0,{Ctrl}{Ins}
P!=xx,0,{Shift}{Ins}
X!=xx,0,{Shift}{Del}
U!=xx,0,{Alt}{Bs}

[Pen Palette]
SKBPos=15 92
```

SYSTEM.INI

The following is an example of a SYSTEM.INI file, for use with Windows 3.1:

```
;-----------------------------------------------------
; Norand 6100 Windows 3.1 SYSTEM.INI
; Based on 6300 Toolkit v1.16
;-----------------------------------------------------

[boot]
shell=winfile.exe
display.drv=6100disp.drv
system.drv=system.drv
keyboard.drv=keyboard.drv
mouse.drv=yesmouse.drv
comm.drv=comm.drv
sound.drv=sound.drv
network.drv=

fonts.fon=egasys.fon
fixedfon.fon=egafix.fon
oemfonts.fon=egaoem.fon

; Grabbers are required for Non-Windows App support.  Optional.
;386grabber=vga.3gr
;286grabber=vgacolor.2gr

language.dll=

;- Installable driver tokens.  See [Drivers] section below for actual driver
;  file names.  Optional drivers can be left out or can be commented out in
;  the [Drivers] section.
drivers=power irda pen penwindows scanner npcp

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
SCRNSAVE.EXE=(None)

[drivers]
;- This is the Required APM driver.  See [Power Driver] section below.
;power=power.crv
power=norwinpm.drv

;- This is the Required pen driver.  See [Pen Driver] section below.
pen=uclkpen.drv

;- This is the Optional scanner driver.  See [Scanner Driver] section below.
;  It is disabled by default.
;scanner=61scan.drv

;- This is one of the settings for enabling Pen Windows, if you use it.  It
;  is disabled by default.
;penwindows=penwin.dll

;- This driver is needed to provide NPCP printing support.  It is disabled
;  by default.
;npcp=nornpcp.drv

;- This driver is needed to provide IrDA printing support.  It is disabled
;  by default.
;irda=norirda.drv

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
[Power Driver]

;– Indicates how often to repeat warnings like “Battery Low”.  Default is 9.
MsgRepeatMinutes=1

;– Sets level of power messages. Default is 1. Here are the currently
;  defined levels. Note that levels above 1 are for debug purposes ONLY!
;    0: No warning or debug messages.  Errors are still displayed.
;    1: Warnings
;    2: Infrequent debug information
;    8: Frequent, but non-repeating debug information
;   10: Frequent, repeating debug information
;   15: All messages plus beep when system activity is detected.
ApmEventDialogs=1

;– Sets the number of seconds without user activity before backlight is
;  turned off.  Default is 30.  Set to 0 to disable Backlight timeouts.
BacklightSeconds=30

;– Sets the number of seconds without system or user activity before system
;  is suspended.  Default is 30.  Set to 0 to disable Suspend timeouts.
SuspendSeconds=60

;– APM Debugging switches ––––––––––––––––––––––––––––––––––––

;– Turns on debug output.  Default is 0.  All messages are written to debug
;  output.
;    0: No debug output.
;    1: Debug output is sent to Windows 3.1 debug monitor.
;       (Example: DBWIN.EXE)
;    2: Debug output is broadcast to DOS and top-level windows.  See Toolkit
;       documents.
;    3: Both methods 1 and 2 above are enabled.
;
;  Run the “DBWIN.EXE” utility that comes with the Windows 3.1 SDK to view
;  debug output.  Clear one output error as Windows starts since NORAPM.DLL
;  tries to output debug data before DBWIN can load.

ApmDebugOutput=0

; Enables Real Time Clock (RTC) diagnostics. Default is 0.  Enables the RTC
; diagnostics built into NORAPM.DLL.  This entry is a DECIMAL representation
; of several bit-mapped options. A diagram of the currently defined fields
; (separated into nibbles for readability) is laid out below. A description
; of each defined field follows:
;
;  0000 | 000M | 0000 | LLLE
;
;   E) Bit 0: Set to 1 to enable RTC checking.  Currently, clock is checked
;          every 10 seconds for regression and writes by other applications.
;          Add 1 to the entry to enable this setting.  If this entry is not
;          set, no other options have any effect.
;
;   L) Bits 1–3: Sets the dialog level for clock regression messages.  See
;          the ApmEventDialogs setting above.  Set to 0 to enable regression
;          warnings at all times.  Set to 1–7 to limit clock warnings.  Add
;          (level * 2) to entry to set a level greater than 0.
;
;   M) Bit 8:  Display a system-model dialog and lock up the system whenever
;          clock regression occurs.  This setting could result in data
;          corruption or loss, since it forces a reboot if the clock goes
;          backwards.  Typical causes of clock regression are setting the
;          date and time via the Control Panel, or communications software
;          that attempts to synchronize the clock on the HHC with a remote
;          system’s clock.  Use of this setting is recommended only for
;          debugging.  Add 256 to the entry to enable this option.
;
;   Bits 4–7 and 9–15 are currently undefined and should be set to 0.
;
; Example settings:
;    0: Default, no clock checking.
```

```
;    1: Enable clock checking, always display clock regression messages.
;   15: Enable clock checking, only display clock regression messages if
;       ApmEventDialogs setting is >= 7.
;  257: Enable clock checking, display message and lock up system upon clock
;       regression.
ApmCheckRTC=0

; Disables the broadcast of Norand-defined power management events to DOS
; drivers and TSR's.  Windows drivers and applications are not affected.
; This sidesteps broken behavior in some DOS TSR's and drivers that monitor
; power management event broadcasts.  Specifically the current SystemSoft
; PCMCIA card management software.  The problem is that DOS drivers only
; examine bottom 8 bits of the event ID.  OEM-defined events are 16 bits,
; thus causing the broken DOS drivers to misidentify events and act
; strangely.  This entry defaults to 1, but should be set to 0, if you are
; using SystemSoft PCMCIA card drivers.  Also, if "ApmFixSystemSoft" entry
; (below) is 1, this entry is forced to 0.
ApmDosOemEvents=0

; This entry enables workarounds required by current SystemSoft PCMCIA
; software (see ApmDosOemEvents above).  This switch currently makes these
; changes to normal APM 1.1 behavior:
;   1. SUSPEND event notifications to DOS are converted to USER_SUSPEND
;      events.  Windows drivers and applications are not affected.  This
;      entry defaults to 0 if not present but should be set to 1 if you are
;      using the SystemSoft PCMCIA card drivers, until further notice.
ApmFixSystemSoft=1

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

[Pen Driver]
; Pen*Key increases wOffsetX to move cursor down relative to pen.
wOffsetX=693
; Increase wOffsetY to move cursor to left relative to pen.
wOffsetY=318

; Increase wDistinctWidth to make cursor move slower relative to pen in the
; up/down direction on screen.
wDistinctWidth=3512

; Increase wDistinctHeight to make cursor move slower relative to pen in the
; left\right direction on screen.
wDistinctHeight=3043

;- 1..200. Set to 120-150 if you install hardware recognition.
PointsPerSecond=50

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

 [Scanner Driver]
; ScannerHardwareType controls the type of hardware the scanner is using.
; This value MUST be set or the driver does not load.  There is no default
; value.  The valid values are:  PEN*KEY/33, PEN*KEY, and TETHERED.
ScannerHardwareType=PEN*KEY

; MessageBeepScanVerification controls the type of beep generated when a
; good scan is obtained.  Valid values are:  OFF, INTERNAL, EXTERNAL, and
; ALL (Does both internal and external).
MessageBeepScanVerification=INTERNAL

; MessageBeepStatusNotification controls the type of beep generated when a
; status change happens.  Valid values are:  OFF INTERNAL, EXTERNAL, and ALL
; (Does both internal and external).
MessageBeepStatusNotification=INTERNAL

; MessageBoxStatusNotification controls whether or not a message box is
; generated when a status change happens.  Valid values are: TRUE and FALSE.
MessageBoxStatusNotification=TRUE

; EnableScannerWhenDriverLoads controls when the scanner is enabled.  If
; TRUE, scanner is enabled when it is loaded by Windows and is always active
; until Windows shuts down.  This option (if TRUE) does not require the
; OpenDriver and CloseDriver calls issued by the application to use the
```

```
; scanner.  If TRUE it does NOT allow multiplexing of the scanner and
; external comm 1 connections.  This option uses more power.  The valid
; values are TRUE and FALSE.
EnableScannerWhenDriverLoads=FALSE

; DisplayScanningDataDialog controls whether or not the Scanning Data.
; Dialog is displayed when the trigger is pulled.  Valid values are:  TRUE
; and FALSE.
DisplayScanningDataDialog=TRUE

; ShowWindowOnLoad controls whether or not the scanner window icon is
; displayed on the screen.  If this is FALSE there is no way to get to the
; scanner window.  The valid values are:  TRUE and FALSE.
ShowWindowOnLoad=TRUE

; EnableScanCodes controls whether or not the scanner includes Scan Codes in
; the key messages that it generates.  Valid values are: TRUE and FALSE
EnableScanCodes=TRUE

; ExternalFlashOnScan controls whether the Good Scan light are flashed
; manually by the scanner driver when the data is received.  The valid
; values are: TRUE and FALSE.
ExternalFlashOnScan=FALSE

; AimingBeamDuration controls the length of time in milliseconds the long
; range scanner using a dedicated UART emits an aiming beam.  All other
; scanners should have this option set to 0 (the default value).
AimingBeamDuration=0

; Use this option to set the Base Address on which the driver looks for the
; Scanner UART.  The default setting is:  488 (1e8).  Valid settings are:
; 1016 (3f8) for COM1, 760 (2f8) for COM2.
DedicatedUARTAddress=488

; Use this option to set the IRQ Line on which the Scanner UART interrupts.
; Default setting is 5.  Other valid settings are:  4 for COM1, 3 for COM2.
DedicatedUARTIRQ=5

; This option is applicable to Hand held Scanners only. It sets the value of
; the Terminating character of the packet sent by the Scanner.  The default
; setting is 03 (ETX).  Other valid settings are: 10 (LF), 13 (CR).
Postamble=03
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

[NPCP Driver]
; Name of device Windows outputs to and is directed out the IR Port.  This
; option must match the settings in WIN.INI
DeviceName=LPT1

; Comm Port Address and IRQ
CommAddress=0x03F8
CommVector=0x0C

; Determines FIFO usage on UART.  A value of 0 disables.
FIFODepth=16

; Determines if power on the port turns on and off.  May not work with all
; non-Norand APM services.
PowerManage=FALSE

; This option disables/enables the QueryAbort processing during printing.
; The default setting is FALSE.  This disables the ability to cancel the
; Print job once it is started.  Valid settings are:  TRUE and FALSE.
ProcessQueryAbort=FALSE

; This option is valid only if the FIFODepth setting is > 0.  Sets the
; Receive interrupt trigger level.  Default value is 0.  Valid settings are
; 0-3.  The Receive interrupt trigger is Chip dependent.  On 550 UART
; setting, a value of 0 causes Chip to interrupt on every character.
; However, on an ST650 UART this implies interrupt on every 8 characters (or
; after 4 char times if there is at least one character in FIFO).  For
; further details, refer to the UART documentation on setting the top 2 bits
; in the FIFO Control Register.
FIFOTriggerLevel=0
```

```
; Use this option to Install/Deinstall the ISR for every Print Job.  This is
; useful if the IRQ is multiplexed with some other device such as the
; Scanner.  The default setting is 0 (ie. no multiplexing).  The valid
; settings are 0 and 1.
IRQMultiplex=0

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

[IrDA Driver]
; DeviceName specifies the name of the device that Windows is sending its
; output to.  The default value is LPT1.DOS which Causes all print output,
; that Windows attempts to send to the DOS device LPT1, is sent to the IrDA
; port.
DeviceName=LPT2

; Use this option to set the Base Address on which the Driver looks for the
; IR UART.  Default setting is 760 (2f8).  Valid settings are 1016 (3f8) for
; COM1, 760 (2f8) for COM2, 1000 (3e8) for COM3.
UARTAddress=1000

; Use this to set the IRQ Line on which the UART interrupts.  Default
; setting is 3 (COM2).  Other valid setting is 4 (COM1). 6100 setting is 14.
UARTIRQ=3

; This sets the type of the Hand Held computer.  Default setting is PENKEY.
; Valid settings are PENKEY, 6100, 6600, JETEYE, and OMNIBOOK.
Technology=PENKEY

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

[Norand 6805 Printer]
; This forces the drivers to use Graphics commands for entire Document.
; Valid settings are TRUE/FALSE. Default value is: FALSE.
DoGraphicsOnly=FALSE

; This sets the idle time in seconds after which the Printer is Awakened
; while printing multiple pages.  Default setting is 10.  Valid settings are
; 10-50.
Timeout=10

; This controls the # of NULL characters sent to the printer, after the
; Initialize command, for it to Wakeup properly.  Default setting is 200.
; Valid settings are 200-500.  Setting a Lower value may disrupt Graphics
; printing.
WakeupChars=200

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

[speaker.drv]

;- This line should be 40-45 on a Pen*Key 25
CPU Speed=40

Volume=900
Version=774
Max seconds=3

Enhanced=0

;- Leave at 0 for best sound.  Set to 1 if having interrupts disabled during
; sound playback causes problems with your configuration.  Disabling
; interrupts during sound playback also causes the DOS clock to lose time
; when sounds are played.  NORAPM.DLL fixes this by correcting the DOS clock
; every 10 seconds or so.  Leave interrupts enabled=0

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 [keyboard]
subtype=
type=4
keyboard.dll=
oemansi.bin=
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;-----------------------------------------------------------------------------
; Some Norand drivers and utilities write a version string to the
; "boot.description" section below to aid in driver identification.
```

```
;-----------------------------------------------------------------------------
[boot.description]
keyboard.typ=
mouse.drv=
network.drv=No Network Installed
language.dll=English (American)
system.drv=MS-DOS System
codepage=437
woafont.fon=English (437)
aspect=100,96,96
display.drv=NORAND Rotate driver for VGA

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

[display driver]
DisplayOrientation=1

; Overrides the Screen Column size used to draw the Alignment screen.  The
; default values are obtained at run time, depending on the Windows mode.
; Valid values are based on the handheld computer being used.
;DisplayColumns=

; Overrides the Screen Row size used to draw the alignment screen.  The
; default values are obtained at run time depending on the Windows mode.
; Valid values are based on the handheld computer being used.
;DisplayRows=

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

[NORAND]
Model=0

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

[standard]

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

;-----------------------------------
; This section for Enhanced mode
;-----------------------------------
[386Enh]
display=*vddvga
keyboard=*vkd
mouse=*vmd
network=*vnetbios, *dosnet

device=vportd.386
device=vpwrd.386
device=vtdapi.386
device=*vpicd
device=*vtd
device=*reboot
device=*vdmad
device=*vsd
device=*v86mmgr
device=*pageswap
device=*dosmgr
device=*vmpoll
device=*wshell
device=*BLOCKDEV
device=*PAGEFILE
device=*vfd
device=*parity
device=*biosxlat
device=*vcd
device=*vmcpd
device=*combuff
device=*cdpscsi

local=CON
FileSysChange=off
```

```
;– COM3 is where the PCMCIA modem installs, if using one.
COM3Irq=10
COM3Base=03E8

;– The following fonts are required for Non-Windows application support.
;  Optional.
;woafont=dosapp.fon
;EGA80WOA.FON=EGA80WOA.FON
;EGA40WOA.FON=EGA40WOA.FON
;CGA80WOA.FON=CGA80WOA.FON
;CGA40WOA.FON=CGA40WOA.FON

;– The following is required for Non-Windows application support
;[NonWindowsApp]
;CommandEnvSize=128
```

EXAMPLE: Example WIN.INI File

The following is an example of a WIN.INI file, for use with Windows 3.1:

```
;-----------------------------------------------------------------
; Norand 6300 Windows 3.1 WIN.INI
;-----------------------------------------------------------------

[windows]
NorShellRun=winfile.exe
spooler=
load=
run=
Beep=yes
NullPort=None
BorderWidth=5
CursorBlinkRate=530
DoubleClickSpeed=595
Programs=com exe bat pif
Documents=
DeviceNotSelectedTimeout=15
TransmissionRetryTimeout=45
KeyboardDelay=2
KeyboardSpeed=31
ScreenSaveActive=0
ScreenSaveTimeOut=120
DoubleClickWidth=96
DoubleClickHeight=96
CoolSwitch=1

DosPrint=no

; Uncomment this line to make the 48xx NPCP printer be the default.
;device=NORAND 4800,NOR4800,LPT1.DOS

; Uncomment this line to make the 6805 IrDA printer be the default.
device=NORAND 6805,NOR6805,LPT2.DOS

[Desktop]
Wallpaper=WINTITLE.RLE
TileWallPaper=0
Pattern=(None)
GridGranularity=0
IconSpacing=50

[Sounds]

; These sounds are played when Windows starts and quits.
SystemStart=tada.wav, Windows Start
SystemExit=chimes.wav, Windows Exit

; These sounds are played when A/C is connected and disconnected.
ApmAcOn=tada.wav, A/C Online
ApmAcOff=chimes.wav, A/C Offline

; To reduce system overhead, do not enable any of these sounds.  Leaving a
; sound entry blank causes a simple beep to be played in its place.
```

```
;SystemDefault=chimes.wav, Default Beep
;SystemExclamation=chimes.wav, Exclamation
;SystemHand=chimes.wav, Critical Stop
;SystemQuestion=chimes.wav, Question
;SystemAsterisk=chimes.wav, Asterisk
 [Extensions]
cal=calendar.exe ^.cal
crd=cardfile.exe ^.crd
trm=terminal.exe ^.trm
txt=notepad.exe ^.txt
ini=notepad.exe ^.ini
pcx=pbrush.exe ^.pcx
bmp=pbrush.exe ^.bmp
wri=write.exe ^.wri
rec=recorder.exe ^.rec
hlp=winhelp.exe ^.hlp
doc=winword.exe ^.doc
dot=winword.exe ^.dot
rtf=winword.exe ^.rtf
[intl]
sLanguage=enu
sCountry=United States
iCountry=1
iDate=0
iTime=0
iTLZero=0
iCurrency=0
iCurrDigits=2
iNegCurr=0
iLzero=1
iDigits=2
iMeasure=1
s1159=AM
s2359=PM
sCurrency=$
sThousand=,
sDecimal=.
sDate=/
sTime=:
sList=,
sShortDate=M/d/yy
sLongDate=dddd, MMMM dd, yyyy
[ports]
; A line with [filename].PRN followed by an equal sign causes [filename] to
; appear in the Control Panel's Printer Configuration dialog box.  A printer
; connected to [filename] directs its output into this file.
LPT1:=
LPT2:=
LPT3:=
COM1:=9600,n,8,1,x
COM2:=9600,n,8,1,x
COM3:=9600,n,8,1,x
COM4:=9600,n,8,1,x
EPT:=
FILE:=
LPT1.DOS=
LPT2.DOS=
[FontSubstitutes]
Helv=MS Sans Serif
Tms Rmn=MS Serif
Times=Times New Roman
Helvetica=Arial
[TrueType]
[Sounds]
SystemDefault=ding.wav, Default Beep
```

```
SystemExclamation=chord.wav, Exclamation
SystemStart=tada.wav, Windows Start
SystemExit=chimes.wav, Windows Exit
SystemHand=chord.wav, Critical Stop
SystemQuestion=chord.wav, Question
SystemAsterisk=chord.wav, Asterisk

[mci extensions]
wav=waveaudio
mid=sequencer
rmi=sequencer

[Compatibility]

[fonts]
Arial Narrow Bold (TrueType)=ARIALNB.FOT
MS Sans Serif 8,10,12,14,18,24 (VGA res)=SSERIFE.FON
Courier 10,12,15 (VGA res)=COURE.FON
MS Serif 8,10,12,14,18,24 (VGA res)=SERIFE.FON
Symbol 8,10,12,14,18,24 (VGA res)=SYMBOLE.FON
Small Fonts (VGA res)=SMALLE.FON

[embedding]
SoundRec=Sound,Sound,SoundRec.exe,picture
Package=Package,Package,packager.exe,picture
PBrush=Paintbrush Picture,Paintbrush Picture,pbrush.exe,picture
;Note-it=MS Note-It,MS Note-It,note-it.exe,picture

[colors]
Background=0 0 0
AppWorkspace=64 0 0
Window=0 0 0
WindowText=255 255 255
Menu=0 0 0
MenuText=255 255 255
ActiveTitle=255 255 232
InactiveTitle=0 0 128
TitleText=0 0 0
ActiveBorder=255 255 232
InactiveBorder=0 64 0
WindowFrame=255 255 255
Scrollbar=128 0 0
ButtonFace=128 0 0
ButtonShadow=0 255 0
ButtonText=255 255 255
GrayText=192 192 192
Hilight=255 255 255
HilightText=0 0 0
InactiveTitleText=128 128 128
ButtonHilight=0 0 0

[NORAND 6805,LPT2.DOS]
Paper Size=256
Paper Length=1450
Paper Width=480
Size Unit=1

[PrinterPorts]
NORAND 4800=NOR4800,LPT1.DOS,15,45
NORAND 6805=NOR6805,LPT2.DOS,15,45

[devices]
NORAND 4800=NOR4800,LPT1.DOS
NORAND 6805=NOR6805,LPT2.DOS
```

SanDisk Card with Stacker

The following example demonstrates a configuration that uses a SanDisk card with Stacker and a custom CONFIG.SYS file. Place these files on the card:

- BOOTDRV.COM
- MMBFLAG.COM
- PC4800.SYS
- RESET.EXE
- STACKER.COM
- XCOPY.EXE
- MYAPP/MYAPP.EXE

EXAMPLE: Sample CONFIG.SYS

```
BUFFERS=20
FILES=128

REM CardSoft drivers (IF USED)
device=d:\cs\sselan.exe
device=d:\cs\cs.exe
device=d:\cs\csalloc.exe
device=d:\cs\atadrv.exe /s:2
device=d:\cs\mtsram.exe
device=d:\cs\mtddrv.exe
device=d:\cs\cardid.exe
device=d:\cs\norcsapm.exe

device=d:\elanapm.exe
device=d:\nordospm.exe

Rem NORAND Card and Socket services drivers:
devicehigh=d:\norcs\elancsss.exe
devicehigh=d:\norcs\stabios.sys /r
devicehigh=d:\norcs\nordata.sys

device=pc4800.sys LPT1 1 /I1
device=stacker.com /p=1 E:
install=d:\4000api.exe
install=d:\mininet.exe
```

KEYS.INI (Key Remapping Parameter File)

This listing is a standard key definition file included in the 6100 Tool Kit. It is a sample for you to modify, as needed. The numbers listed below the "Unshifted Plane" and the "Yellow Shifted Plane" represent the key numbers for the keys on the 6100 Computer. Their physical positions shown below correspond to the physical positions of the keys on the 6100 keypad.

NOTE: *The key numbers, scan codes, and key definition columns may conform to your particular key layout. Adapt this to your unit.*

```
;==========================================================================
; 6100 keys are numbered as follows:
; Unshifted Plane
;      3       2       1       0
;      7       6       5       4
;     11      10       9       8
;     15      14      13      12
; Yellow Shifted Plane
;     67      66      65      64
;     71      70      69      68
;     75      74      73      72
;     79      78      77      76
; special scan codes:
;     0x6d    Backlight On
;     0x6e    Contrast Down
```

```
;    0x6f    Contrast Up
;    0x70    Suspend
;  NOTE:   You cannot remap key 15 or 79 (the Yellow key)
;==========================================================================
;Key #     Scan Code    ; Key Definition
   0   =  0x0A         ; 9
   1   =  0x09         ; 8
   2   =  0x08         ; 7
   3   =  0x70         ; On/Off (Suspend/Resume, I/O)
   4   =  0x07         ; 6
   5   =  0x06         ; 5
   6   =  0x05         ; 4
   7   =  0x0F         ; Tab
   8   =  0x04         ; 3
   9   =  0x03         ; 2
  10   =  0x02         ; 1
  11   =  0x0E         ; BackSpace
  12   =  0x1C         ; Enter
  13   =  0x0B         ; 0
  14   =  0x01         ; Escape
  64   =  0x6D         ; Backlight On/Off
  65   =  0x6E         ; Contrast Down
  66   =  0x6F         ; Contrast Up
  67   =  0x38         ; Alt
  68   =  0x51         ; Page Down
  69   =  0x48         ; Up Arrow
  70   =  0x49         ; Page Up
  71   =  0x47         ; Home
  72   =  0x4D         ; Right Arrow
  73   =  0x50         ; Down Arrow
  74   =  0x4B         ; Left Arrow
  75   =  0x34         ; Period "."
  76   =  0x4FC        ; End
  77   =  0x0C         ; Minus "–"
  78   =  0x1D         ; Control
```

Setups for Third Party Applications

This code may be helpful in setting up third-party applications, but keep in mind the third-party companies are the best sources for support for this subject.

Some of the files required are located in the Tool Kit (such as the mouse and calibration programs), some are supplied with PenPal, and some can be separately purchased from Intermec Technologies Corporation.

EXAMPLE: Sample PenPal (DOS) Setup

The following is an example of DOS code that could set up PenPal (DOS), through the use of an AUTOEXEC.BAT file.

```
PATH a:\;d:\;
rem *** Always put ELANAPM.EXE before ELANCFG ***
ELANAPM.EXE

REM Disable Video Memory writes as Activity
D:/ELANCFG –V0 /H8 /L12 /D20 /T2

REM Load the handwriting recognition
call PENDDSEM.BAT

REM Load the PEN PAL modified pen driver
61MOUSE.COM

REM Calibrate, if required.
CALIB.EXE

REM Run the sample PEN PAL program – Other sample programs may be chosen
PPCP <APP.RUN>
```

Required Files

- PPCP.EXE (PenPal control program)
- DPM16BI.OVL (If protected mode PenPal)
- RTM.EXE (also load these files.)
- PPRCP.RSC (Specific to the 6100 Computer)
- PPCP.REG (Reg. file)
- 61MOUSE.COM (Pen driver program)
- CALIB.EXE (Calibration program)
- N6100.BGI (For calibration program)
- EMULIX.EXE (For handwriting recognition)
- USGREC.EXP
- VLOAD.EXE
- EMM386.EXE
- PENDOSEM.BAT

EXAMPLE: Sample PenRight! (DOS) Setup
The following is an example of DOS code that could set up PenRight! (DOS), through the use of an AUTOEXEC.BAT.

```
PATH a:\;d:\;
REM Disable Video Memory writes as Activity
D:/ELANCFG –V0 /H8 /L12 /D20 /T2
REM Load the modified pen driver
SEST APIRSC=61PENRT.RSC
REM Calibrate, if required.
CALIB
REM Run the sample PEN PAL program
PENR!api /e /k /x=320 /y=240
PENR!hwp /e
REM Start application
APP.EXE
```

▶ **NOTE:** *PENR!API.EXE is not the same as one for a desktop computer. You need one that comes with a PenRight! machine.*

Required Files

- 61MOUSE.RSC {Specific to the 6100 Computer)
- 61MOUSE.COM (Modified pen driver program)
- CALIB.EXE (Calibration program)
- PENR!API.EXE (PenWrite application)
- PENR!HWP.EXE (PenWrite Handwriting Recognition)
- N6100.BGI (for CALIB.EXE)

Handwriting Recognition System Setup

EXAMPLE: Sample AUTOEXEC.BAT Code for Handwriting Recognition

```
PATH A:/;D:\;
REM Load the PEN PAL modified pen driver
call PENDOSEM.BAT
61MOUSE.COM
REM Run the sample PEN PAL program
PPCP 6100.RUN
```

EXAMPLE: Sample CONFIG.SYS Code for Handwriting Recognition

```
shell=d:\command.com d:\ /e:512 /p
device=d:\himem.sys /machine:2
devicehigh=d:\elanump.sys /x=C000,D400,D800,DC00,E000
device=a:\emm386.exe i=D400–EFFF FRAME=D400
device=d:\elanapm.exe
devicehigh=A:\nordosppm.exe
dos=high,umb
break=on
buffers=30
files=50
```

EXAMPLE: Sample PENDOSEM.BAT Code for Handwriting Recognition

```
@ECHO off
REM %1 is the prefix for all files here
LH %1VLOAD %1USAREC.EXP
LH $1EMUL1X
SET PENDOS12=D9
```


APM Event Code Broadcast Values

```
//-----------------------------------------------------------------------------
// APM 1.1 BIOS event codes.
// These codes are broadcast to DOS TSR's by the APM OS Driver.
//      System Standby Request Notification
#define APM_STANDBYREQUEST   1
//      System Suspend Request Notification
#define APM_SUSPENDREQUEST   2
//      Normal Resume System Notification
#define APM_SUSPENDRESUME       3
//      Critical Resume System Notification
#define APM_CRITICALRESUME   4
//      Battery Low Notification
#define APM_BATLOW           5
//      User System Standby Request Notification
#define APM_USERSTANDBY      9
//      User System Suspend Request Notification
#define APM_USERSUSPEND      10
//      User System Standby Resume Notification
#define APM_USERSTANDBYRESUME 11
//      Status Request – Ok to suspend?
#define APM_STATUS           0xFF
//-----------------------------------------------------------------------------
// Windows APM 1.1 event codes.  These events are broadcast to the Windows
// drivers and parent-windows by the Windows APM driver.  Note that these codes
// are one less than the corresponding BIOS codes.  Also, only three codes, 1–3,
// are actually defined in the current Windows 3.1 WINDOWS.H header file.
//      System Standby Request Notification
#define PWR_STANDBYREQUEST   0
//      System Suspend Request Notification
#define PWR_SUSPENDREQUEST   1
//      Normal Resume System Notification
#define PWR_SUSPENDRESUME       2
//      Critical Resume System Notification
#define PWR_CRITICALRESUME   3
//      Critical System Suspend Notification
#define PWR_CRITICALSUSPEND     7
//      User System Standby Request Notification
#define PWR_USERSTANDBY      8
//      User System Suspend Request Notification
#define PWR_USERSUSPEND      9
//      User System Standby Resume Notification
#define PWR_USERSTANDBYRESUME 10
```

BGI Support

N6100.BGI is in the 6100 Tool Kit, and it is a custom Borland Graphics Interface. It develops Borland graphics based applications on the 6100 Computer. There are two ways to use this in a DOS application. The first is to keep the application and the BGI driver separate. This requires loading the driver from your application. The calls to do this are documented in Borland C++ DOS Reference. Start with installuserdriver. This is the method used by Calib, described in this section.

For details on how you may include this into your DOS application see the UTILS.TXT file that describes how to use the BGIOBJ converting utility. Also, see the registerbgidriver routine in the DOS Reference to apply this method. You will need the following files:

- N6100.BGI Real mode PENKEY.BGI driver
- N6100.H Interface file for use with the BGI driver

Using the N6100.BGI Driver

The driver "plugs in" to the Borland BGI graphics library via the installuserdriver() function. Mode selection is done via a series of #define symbols in the N6100.H header file – note that the BGI DETECT autodetect logic does not work with this driver. The following modes are available from this driver:

Resolution	Symbol
230x240x4	BGI_LANDSCAPE
240x320x4	BGI_PORTRAIT

EXAMPLE: Init the driver in 320x240x4 mode:

```
#include <graphics.h>
#include "n6100.h"

int driver, mode;
driver = installuserdriver("N6100",NULL);  /* do this only once */
mode = BGI_LANDSCAPE;
initgraph($drvr,$mode,"");
```

EXAMPLE: Init the driver in 240x320x4 mode:

```
#include <graphics.h>
#include "n6100.h"

int driver, mode;
driver = installuserdriver("N6100",NULL);  /* do this only once */
mode = BGI_PORTRAIT;
initgraph($drvr,$mode,"");
```

See the demo program TEXT.C, included in the Tool Kit, for an example of how to initialize and use the N6100.BGI driver.

▶ **NOTE:** *Init and close the driver as many times as desired during a program run, but only call installuserdriver() once and save the value returned for use in subsequent initializations.*

Bitmap Text Output

The standard Borland BGI drivers do not fill the background color when rendering the bitmap font, which means the text must be erased before it can be written over, and erasure requires the use of the bar() function to clear the text area (text cannot be written over with new text or spaces). Most developers find this behavior troublesome. The N6100.BGI driver fills the background of bitmap text, eliminating tedious text erasure gyrations. This new textmode is set using the BGI setwritemode() function and passing the BGI_NIFTYTEXT constant (defined in N6100.H). To go back to normal BGI text rendering, pass BGI_NORMALTEXT to the setwritemode() function (which also seems troublesome).

EXAMPLE: Sample N6100.H File

The following is a sample listing of the N6100.H file written by Ryle Design used as an interface for a BGI driver.

```
/*-----------------------------------------------------------------------------
   Header N6100.H

      Header file containing public declarations for the Ryle Design
      N6100.BGI driver written for the Norand Corporation.

   Copyright (c) 1995 Norand Corporation

   Written by Ryle Design, PO Box 22, Mt. Pleasant MI 48804 USA
   Voice/Fax: 517.773.0587 Email: 73047.1765@compuserve.com

   V1.00 08.95 thl
-----------------------------------------------------------------------------*/

#ifdef __DPMI32__
#error N6100.BGI not supported under BGI32
#endif

/* landscape mode flag */

#define BGI_LANDSCAPE   0x0000

/* rotated “portrait mode” flag */

#define BGI_PORTRAIT       0x0008

/* bitmap text modes */

#define BGI_NORMALTEXT     0x0002
#define BGI_NIFTYTEXT   0x0003

void _Cdec1 n6100_driver(void);

/* end header N6100.H */
```

Appendix B

Common PEN*KEY 6000 Series Information

Introduction

This section contains information that is common to most of the PEN*KEY® 6000 Series Hand-Held Computers. If you are a new user, you may find this information useful. The following list outlines the major topics in the section.

Topic Summary

Development Support Files

NORAPM.H

```
// NORAPM.H: Header file containing exported structures and functions for
// interfacing with the NORAPM.DLL NORAND Windows APM Driver.
// Created: 01.25.94
// Updated: 07.07.94
// (C) Copyright 1994 by Norand Corporation

//=====================================================================
// Section: NORAPM Exported Structures
//=====================================================================

/* tagPOWER_STATUS structure ****************************************
This structure is used by GetPowerStatus function to relay the APM BIOS system
power status.  Structure fields are direct representations of registers returned
by the APM Get Power Status call (see below):
*********************************************************************/
struct tagPOWER_STATUS {
   BYTE BatteryStatus;  // Battery status bits changed since last call
     // 00H  High
     // 01H  Low
     // 02H  Critical
     // 03H  Charging
     // FFH = Unknown
     // All other values reserved

   BYTE LineStatus;        // AC line status bits
     // 00H  Off-line
     // 01H  On-line
     // 02H  On backup power
     // FFH = Unknown
     // All other values reserved

   BYTE BatteryCharge;     // % of battery left
     // 0–100 = Percentage of full charge
     // FFH = Unknown
     // All other values reserved

   BYTE BatteryFlags;   // Current battery status.
     // bit 0 = 1  High
     // bit 1 = 1  Low
     // bit 2 = 1  Critical
     // bit 3 = 1  Charging
     // bit 7 = 1  No system battery
     // All other bits reserved
     // FFH = Unknown
     // All other values reserved

   WORD BatteryLife;       // Battery time left
     // Bit 15 = 0  Time units are seconds
     //          1  Time units are minutes
     // Bits 14–0 = Number of seconds or minutes
     //                 0–7FFFH = Valid number of seconds
     //                 0–7FFEH = Valid number of minutes
     //                 FFFFH   = Battery life Unknown
};

//=====================================================================
// Section: NORAPM Exported Function Prototypes and Pointer Defines
//=====================================================================

// This section declares a prototype for each function exported by NORAPM.DLL
// as well as a typedef for a pointer to each function to facilitate runtime
// dynamic linking.

//---------------------------------------------------------------------
// extern "C" int FAR PASCAL _export GetPowerStatus(tagPOWER_STATUS far
```

```
// *PwrStat)
//
// Returns current power status from APM.  See NorAPM.H for detailed docs.
// Return value of PWR_OK for success, or PWR_FAIL if error (no APM, etc).
//-----------------------------------------------------------------------
extern "C" int FAR PASCAL _export
          GetPowerStatus(tagPOWER_STATUS far *PwrStat);
typedef int (FAR PASCAL *fpGetPowerStatus)
         (tagPOWER_STATUS far *PwrStat);

//-----------------------------------------------------------------------
// extern "C" int FAR PASCAL _export GetPowerState(int Device)
//
// Returns APM BIOS power state for APM Device.  If device does not exist, OFF
// is returned.
//-----------------------------------------------------------------------
extern "C" int FAR PASCAL _export GetPowerState(int Device);
typedef int (FAR PASCAL *fpGetPowerState)(int Device);

//-----------------------------------------------------------------------
// extern "C" int FAR PASCAL _export
          SetPowerState(unsigned int Device,
// unsigned int State)
//
// Sets APM BIOS device "Device" to power state "State".  Returns PWR_FAIL for
// failure (No APM, unsupported device or power state) or PWR_OK for success.
//-----------------------------------------------------------------------
extern "C" int FAR PASCAL _export SetPowerState(unsigned int Device,
                                   unsigned int PowerState);
typedef int (FAR PASCAL *fpSetPowerState)(int Device, int PowerState);

//-----------------------------------------------------------------------
// extern "C" void FAR PASCAL _export SystemActivity(int EventType)
//
// Call this function to indicate system or user activity and hold off standby
// (backlight) and suspend timeouts.  This function is generally called by a
// device driver that controls background processing or communications
// peripherals.
//
// EventType Parameter:
//   0: Indicates "system" activity.  This resets Suspend timeouts.
//   1: Indicates "user" or input activity.  This resets the Standby
// (backlight), and Suspend timeouts.
//-----------------------------------------------------------------------
extern "C" void FAR PASCAL _export SystemActivity(int EventType);
typedef void (FAR PASCAL *fpSystemActivity)(int EventType);
```

APMCODES.H

```
// APMCODES.H: Definitions of APM 1.1 event codes used by NORAPM.DLL for
// interaction with BIOS, DOS, and Windows.
// (C) Copyright 1994 by Norand Corporation

// Indicate that this include file has been loaded
#define APMCODES 1

//-----------------------------------------------------------------------
// APM 1.1 BIOS event codes. These codes are broadcast to DOS TSRs by the
// APM OS Driver.
#define APM_STANDBYREQUEST       1 // System Standby Request Notification
#define APM_SUSPENDREQUEST       2 // System Suspend Request Notification
#define APM_SUSPENDRESUME        3 // Normal Resume System Notification
#define APM_CRITICALRESUME       4 // Critical Resume System Notification
#define APM_BATLOW               5 // Battery Low Notification
#define APM_PWRCHANGE          6 // Power Status Change Notification
#define APM_TIMEUPDATE           7 // Update Time Notification
#define APM_CRITICALSUSPEND      8 // Critical System Suspend Notification
#define APM_USERSTANDBY          9 // User System Standby Request Notif.
```

```
#define APM_USERSUSPEND          10 // User System Suspend Request Notif.
#define APM_USERSTANDBYRESUME 11 // User System Standby Resume Notif.

//-----------------------------------------------------------------
// NORAND APM 1.1 BIOS OEM event codes that are broadcast to DOS.
#define APM_OEM_XKBD         0x0201// External keyboard detect
#define APM_OEM_CHARGE       0x0202// External charge detect
#define APM_OEM_POD1         0x0203// Pod 1 ring detect
#define APM_OEM_POD2         0x0204// Pod 2 ring detect
#define APM_OEM_EXTRING      0x0205// External ring detect
#define APM_OEM_PCMCIA       0x0206// PCMCIA ring detect
#define APM_OEM_ALARM        0x0207// Real time clock alarm
#define APM_OEM_PENDOWN      0x0208// Pen down detect
#define APM_OEM_SUSPEND      0x0209// Suspend detect
#define APM_OEM_RESUME       0x020a// Resume key detect
#define APM_OEM_BUBLOW       0x020b// Backup battery low
#define APM_OEM_MB0          0x020c// Main battery critical
#define APM_OEM_MB10         0x020d// 10 minutes left on bat
#define APM_OEM_MB20         0x020e// 20 minutes left on bat
#define APM_OEM_MB30         0x020f// 30 minutes left on bat
#define APM_OEM_PDOOR        0x0210// PCMCIA door open  detect
#define APM_OEM_IKBD         0x0211// Internal keyboard detect
#define APM_OEM_PWRFAIL      0x0212// Powerfail detect
#define APM_OEM_PWRTICK      0x0213// Charge tick event
#define APM_OEM_EL           0x0214// Backlight on/off press
#define APM_OEM_BATT_CHANGE  0x0215// Battery change detected
#define APM_OEM_PWRFAILNEW   0x0216// Powerfail detect
#define APM_OEM_PWRMSGTXT    0x0280// Power msg text broadcast.

//-----------------------------------------------------------------
// Windows APM 1.1 event codes.  These events are broadcast to the Windows
// drivers and parent-windows by the Windows APM driver.  Note that these codes
// are one less than the corresponding BIOS codes. Also, only 3 codes 1...3,
// are actually defined in the current Windows 3.1Windows.H header file
#define PWR_STANDBYREQUEST      0 // System Standby Request Notification
#define PWR_SUSPENDREQUEST      1 // System Suspend Request Notification
#define PWR_SUSPENDRESUME       2 // Normal Resume System Notification
#define PWR_CRITICALRESUME      3 // Critical Resume System Notification
#define PWR_BATLOW              4 // Battery Low Notification
#define PWR_PWRCHANGE         5 // Power Status Change Notification
#define PWR_TIMEUPDATE          6 // Update Time Notification
#define PWR_CRITICALSUSPEND     7 // Critical System Suspend Notification
#define PWR_USERSTANDBY         8 // User System Standby Request Notification
#define PWR_USERSUSPEND         9 // User System Suspend Request Notification
#define PWR_USERSTANDBYRESUME 10// User System Standby Resume Notification

//-----------------------------------------------------------------
// NORAND Windows APM 1.1 OEM event codes. Again, these are one less than
// their DOS/BIOS counterparts.
#define PWR_OEM_XKBD         0x0200// External keyboard detected
#define PWR_OEM_CHARGE       0x0201// External charge detected
#define PWR_OEM_POD1         0x0202// Pod 1 ring detected
#define PWR_OEM_POD2         0x0203// Pod 2 ring detected
#define PWR_OEM_EXTRING      0x0204// External ring detected
#define PWR_OEM_PCMCIA       0x0205// PCMCIA ring detected
#define PWR_OEM_ALARM        0x0206// Real time clock alarm
#define PWR_OEM_PENDOWN      0x0207// Pen down detected
#define PWR_OEM_SUSPEND      0x0208// Suspend detected
#define PWR_OEM_RESUME       0x0209// Resume key detected
#define PWR_OEM_BUBLOW       0x020a// Backup battery low
#define PWR_OEM_MB0          0x020b// Main battery critical
#define PWR_OEM_MB10         0x020c// 10 minutes left on battery
#define PWR_OEM_MB20         0x020d// 20 minutes left on battery
#define PWR_OEM_MB30         0x020e// 30 minutes left on battery
#define PWR_OEM_PDOOR        0x020f// PCMCIA door open detected
#define PWR_OEM_IKBD         0x0210// Internal keyboard detected
#define PWR_OEM_PWRFAIL      0x0211// Powerfail detected
#define PWR_OEM_PWRTICK      0x0212// Charge tick event
```

```
#define PWR_OEM_EL            0x0213// Backlight on/off press
#define PWR_OEM_BATT_CHANGE   0x0214// Battery change detected
#define PWR_OEM_PWRFAILNEW    0x0215// Powerfail detected
#define PWR_OEM_PWRMSGTXT     0x027F// Power message text broadcast.
//-----------------------------------------------------------------
// APM BIOS standard device IDs
#define APM_BIOS              0x0000 // APM BIOS itself
#define APM_SYSTEM            0x0001 // All BIOS-managed devices
#define APM_DISPLAY           0x0100 // Display controller (VGA, etc.)
#define APM_PCMCIA            0x0200 // PCMCIA slot controller
#define APM_PARALLEL_PORTS    0x0300 // Printer ports
#define APM_SERIAL_PORTS   0x0400 // UARTs
#define APM_LAN_ADAPTER       0x0500 // Ethernet controller
#define APM_SLOTS             0x0600 // Cards in PCMCIA slots
//-----------------------------------------------------------------
// APM BIOS OEM Norand device IDs
#define APM_OEM_DEVICES       0xE000// Remainder are OEM unique
#define APM_OEM_PODS          0xE000// Two pod devices possible
#define APM_OEM_DIG_PADS   0xE100   // Digitizer interface control
#define APM_OEM_BACKLIGHT     0xE200// Backlight control
#define APM_OEM_HEATER        0xE300// 4600 display heater control
#define APM_OEM_SWV5          0xE400// Wwitch 5 control for 4600
//-----------------------------------------------------------------
// APM power states
#define APM_READY    0  // "ON"
#define APM_STANDBY  1  // Still operational, but in low power state.
#define APM_SUSPEND  2  // Not operational, but saves HW state
#define APM_OFF      3  // Not operational and does not save HW state.
//-----------------------------------------------------------------
// APM Event Return Codes for DOS.
// The Windows versions of these codes (PWR_FAIL, PWR_OK) are in WINDOWS.H.
#define APM_OK                0     // Continue operation
#define APM_FAIL              0x80  // Cancel operation
```

Sample Program Listings

Charge Detection Demo Program: TESTCHRG.CPP

TESTCHRG is a C++ Windows program that demonstrates a method for accessing NORAPM.DLL to get the current charge status (tells you whether you are in the dock or not). The program needs to include NORAPM.H. The sample program TESTCHRG.CPP is presented in the following pages.

```
// TESTCHRG.CPP: NorAPM charge detection test fixture v1.01.
// (C) Copyright 1994 by Norand Corporation
// ************************************************************
/* TESTCHRG.DEF
NAME            TESTCHRG
DESCRIPTION     'Tests NorAPM Charge Detection'
EXETYPE         WINDOWS
STUB            'WINSTUB.EXE'
CODE            PRELOAD MOVEABLE DISCARDABLE
DATA            PRELOAD MOVEABLE MULTIPLE
HEAPSIZE        1024
STACKSIZE       8200
*/
#include <windows.h>
#include "norapm.h"
char szTitle[] = "TESTCHRG.EXE – 94.07.07.01";
HINSTANCE hinstNorAPM;       // For storing NorAPM library instance.
```

B. Common PEN*KEY 6000 Series Info.

```
tagPOWER_STATUS PwrStat;      // Holds current power status.
WORD Error;                   // Holds error codes.

char MsgBuf[100];             // Message buffer

#pragma argsused
int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
          LPSTR lpCmdLine, int nCmdShow) {
  // Turn off the "File not found" error box.
  SetErrorMode(SEM_NOOPENFILEERRORBOX);

  // Get handle to NORAPM.DLL driver

  hinstNorAPM = LoadLibrary("NORAPM.DLL");

  // Did we find the library?
  if (hinstNorAPM > HINSTANCE_ERROR) {
    // Get Entry point for GetPowerStatus()
    fpGetPowerStatus lpfnGetPowerStatus = (fpGetPowerStatus)
      GetProcAddress(hinstNorAPM, "GetPowerStatus");

    if (lpfnGetPowerStatus) {
      // Get and display charge state
          do {
          Error = lpfnGetPowerStatus(&PwrStat);
          if (Error != PWR_OK) {
            wsprintf(MsgBuf,
              "GetPowerStatus() call failed with result of %d", Error);
            Error = MessageBox(NULL, MsgBuf, szTitle,
              MB_RETRYCANCEL + MB_ICONEXCLAMATION);
          } // then
          else {
            // Display battery status
            wsprintf(MsgBuf, "A/C is %s line.\nBattery is %scharging\n",
              (char far *)
              (PwrStat.LineStatus == 1 ? "on" : "off"),
              (char far *)
              (PwrStat.BatteryFlags & 8 ? "" : "NOT "));
            Error = MessageBox(NULL, MsgBuf, szTitle,
             MB_RETRYCANCEL + MB_ICONINFORMATION);
          } // else
        } while (Error == IDRETRY);
    } // then
    else
      MessageBox
        (NULL, "Could not get entry point for GetPowerState()!",
        szTitle, MB_OK | MB_ICONEXCLAMATION);

    // Free NorAPM instance.  It stays around since it was already open.
    FreeLibrary(hinstNorAPM);
  } // then
  else
    MessageBox(NULL, "Could not open NORAPM.DLL!", szTitle,
  MB_OK | MB_ICONEXCLAMATION);

  // Exit application
  return (0);
} // WinMain
```

Critical Error Handler: CRITICAL.C

CRITICAL.C is a C++ program that demonstrates a method for handling critical errors. This sample program was designed for a previous PEN*KEY 6000 Series product and may need altering for the 6100 Computer.

```
#include "stdio.h"
#include "compat.h"
#include "4800.h"
extern  uchar jamarray[4][1024];
```

```
extern  int  jamindex;
extern  FILE *stdprn;
extern  unsigned page_no;
extern  unsigned line_cnt;

/*****************************************************************/
/* Critical error handler routines for the printer follow. ********/
/*****************************************************************/
struct reqpk {
      unsigned char reqlen, unit, cmd;
      unsigned status;
      long rsrv1, rsrv2;
      unsigned char lookahead;
      FP(unsigned char) buff;
      unsigned int bytes;
      };
struct devhdr{
   FP(struct devhdr)    next;
   unsigned             attribute;
   void                 (*strategy)();
   void                 (*interrupt)();
   uchar                name[8];
};
struct  faddr{
   void *offset;
   void *segment;
};
union fncptr {
   FAR (void)   (*fnc)();
   struct faddr    addr;
   FP(struct devhdr) fncx;
};

void set24h(fnc)
FP(void) fnc;
{
#asm
   xor   ax,ax
   mov   es,ax
   mov   bx,24h*4
   mov   ax,[bp+4]
   mov   dx,[bp+6]
   cli
   mov   es:[bx],ax
   mov   es:[bx+2],dx
   sti
#endasm
}
drivercall(fncptr,pak)
FP(void) pak;
FAR (void) (*fncptr)();
{
#asm
   les   bx,8[bp]
   call  dword ptr 4[bp]
#endasm
}
baderr(error,devtype,hdr)
unsigned error,devtype;
FP(struct devhdr) hdr;
{
   union fncptr strat;
   union fncptr inter;
   struct ioargs args;
   struct reqpk devpack;
   uchar buffer[40];
   uchar cmd,exterr;
```

```
uchar ch;
uchar jam;
uchar jam2,jam3,jam4;

cmd = pXERROR;
args.ioctl_cmd = &cmd;
args.ioctl_buf = buffer;

devpack.reqlen = sizeof devpack; /* Length of ioctl packet */
devpack.cmd = 3;                    /* ioctl input read */
devpack.buff = (FP(uchar)) &args;   /* Point buffer to ioargsc */
devpack.bytes = 7;                  /* Read one byte */

inter.fncx = strat.fncx = hdr;
inter.addr.offset = hdr–>interrupt;
strat.addr.offset = hdr–>strategy;
drivercall(strat.fnc,(FP(struct reqpk)) &devpack);
drivercall(inter.fnc,(FP(struct reqpk)) &devpack);

exterr = args.ioctl_buf[0];

if (exterr == HEADJAM)  {
   jam = args.ioctl_buf[3];
   jam2 = args.ioctl_buf[4];
   jam3 = args.ioctl_buf[5];
   jam4 = args.ioctl_buf[6];
}
scr_printf (“\nprinter error\n %d\n”,exterr);

switch (exterr) {
   case PNRDY:
        scr_printf(“Printer not ready. Check connection. Retry.\n”);
        break;
   case RXTMO:
        scr_printf(“Printer not transmitting. Check connection.Retry.\n”);
        break;
   case TXTMO:
        scr_printf(“Printer not receiving. Check connection. Retry. \n”);
        break;
   case BADADR:
        scr_printf(“MAC address error. Abort. \n”);
        break;
   case GAPERR:
        scr_printf(“MAC intercharacter Gap error. Retry.\n”);
        break;
   case LSRPE:
        scr_printf(“MAC length parity error. Retry.\n”);
        break;
   case IFTS:
        scr_printf(“Invalid communication Frame.  Abort.\n”);
        break;
   case NS_NE_VR:
        scr_printf(“NS <> VR.  Abort. \n”);
        break;
   case NR_NE_VS:
        scr_printf(“NR <> VS.  Abort. \n”);
        break;
   case MAC_CRCERR:
        scr_printf(“CRC error. Retry. \n”);
        break;
   case RLENERR:
        scr_printf(“Receive length error. Abort.\n”);
        break;
   case FRMERR:
        scr_printf(“Frame reject. Abort. \n”);
        break;
   case NDMERR:
        scr_printf(“Printer Disconnect. Abort.\n”);
        break;
   case BINDERR:
```

```
            scr_printf("Bind error. Abort.\n");
            break;
         case IPLDUR:
            scr_printf("Invalid PLDU. Abort.\n");
            break;
         case HEADJAM:
            jamarray[0][jamindex] = jam;
            jamarray[1][jamindex] = jam2;
            jamarray[2][jamindex] = jam3;
            jamarray[3][jamindex++] = jam4;
            scr_printf("head jam. Abort.\n");
            scr_printf("page: %u\n",page_no);
            scr_printf("line: %u\n",line_cnt);
            scr_printf( jam & 1 ? "Left ":"Right ");
            scr_printf("\n%s",((jam & 2) ? "Accel": (jam & 4) ? "Decel":
                        "Print"));
            scr_printf("\n%s", ((jam & 16) ? "Cmove": (jam & 32) ? "Nmove":
                   "Pmove"));
            scr_printf("\n%s",((jam &128) ? "highspeed": "lowspeed"));
            scr_printf("\ncnter0 = %d\n",jam2);
            scr_printf("\interrupts = %d\n",jam3);
            tdelay(10000);
            break;
         case PAPEROUT:
            scr_printf("Printer paper out.  Retry.\n");
            break;
         case LOWVOLTS:
            scr_printf("Printer low voltage. Retry.\n");
            break;
         case HIVOLTS:
            scr_printf("Printer over voltage. Retry.\n");
            break;
         case LOWBAT:
            scr_printf("Printer low battery. Retry.\n");
            break;
         case COVEROFF:
            scr_printf("Printer cover off. Retry.\n");
         default:
            break;
      }
      if (exterr != HEADJAM)  ch = (scr_getc());
      while (1) {
         scr_printf("abort, retry, ignore, fail :");
         ch = (exterr != HEADJAM) ? (scr_getc()) : 'r';
         scr_printf("%c \n",ch);
         ch = tolower(ch);
         switch (ch) {
            case '1':
            case 'a': return (2);
            case '2':
            case 'r': return (1);
            case '3':
            case 'i': return (0);
            case '4':
            case 'f': return (3);
            default: break;
         }
      }
   }
   FAR (void) errproc()
   {
   #asm
      sti
      push ds
      push es
```

```
;  pusha
   push  ax
   push  bx
   push  cx
   push  dx
   push  si
   push  di
   mov   dx,dataseg
   mov   ds,dx
   mov   dx,[bp]
   push  dx
   push  si
   push  ax
   push  di
   call  baderr_
   add   sp,8
   mov   -6[bp],al  ;set return code
;  popa
   pop   di
   pop   si
   pop   dx
   pop   cx
   pop   bx
   pop   ax
   pop   es
   pop   ds
   mov   sp,bp
   pop   bp
   iret
#endasm
}
```

IDLE.CPP

```
//
// CPU Idle call sample source code
//
#include <dos.h>     // int86
#include <stdio.h> // printf, etc
//
// CPU Idle calls cause the processor to halt (via HLT instruction) until the
// next hardware interrupt.  This causes good power savings if used during
// input polling loops, or other low activity points in an application.  See
// MS-Intel APM 1.1 specification or Ralf Brown's interrupt list, for info.
//
unsigned char APMCPUIdle(void) {
union REGS regs;
regs.x.ax= 0x5305;
int86(0x15, &regs, &regs); // makes actual function call being tested
if (regs.x.cflag==0) {
   // CPU Idle successful
   return (0);
   }
else {
   // An error occurred, check error code.
   switch (regs.h.ah) {
      case (0x03);
         // Interface not connected
         break;
      case (0x0B);
         // Interface not engaged
      default:
         // Unrecognized Return code
         break;
      }
   return ((unsigned carh)regs.h.ah);      // return the error code
```

```
   }
} // end APMCPUIdle
void main(void) {
printf("Result of CPU Idle call: %2.2X\n", APMCPUIdle());
return;
}
```

Keyboard Remapping, with ANSI.SYS

Another way to remap the keyboard is to use ANSI.SYS, which should work on any computer that supports ANSI.SYS. This is a very portable way to remap keys. ANSI.SYS works the same across all PC-compatible platforms. It provides the capability of reassigning keys, through the use of escape sequences. An escape sequence is a string of text that begins with two preliminary keys: the escape character and the left-bracket character "]". Refer to DOS help or to the Using ANSI.SYS paragraph, below, for additional information on composing the remapping strings for use with ANSI.SYS.

Each key on the keyboard has a specific key code. Refer to any standard keyboard interface definition to determine keycodes for the keys.

Example Key Redefinition

The following example demonstrates one way to remap function keys F1–F10, using ANSI.SYS. This method remaps any key by supplying the appropriate key numbers in place of the "0;xx", where "0;xx" is the key code (such as 0;59).

Add the following line to your CONFIG.SYS file where the /*x* switch selects keyboard redefinition, as specified in the DOS Help for ANSI.SYS:

```
device=c:\dos\ansi.sys /X
```

Add the following assignments (with any modifications) that you prefer to suit your needs) to your AUTOEXEC.BAT file, to set these mappings each time your system is powered up or reset:

```
echo on
prompt $<–[0;59;"INTERSVR C: D:";13p
prompt $<–[0;60;"DIR";32p
prompt $<–[0;61;"BIOS";32p
prompt $<–[0;62;"CD";32p
prompt $<–[0;63;"COPY";32p
prompt $<–[0;64;"DEL";32p
prompt $<–[0;65;"RAMDFMT";32p
prompt $<–[0;66;"C:";13p
prompt $<–[0;67;"EXIT";13p
prompt $<–[0;68;"D:";13p
prompt $p$g
```

Explanation of Example

The listing above is shown is as it will appear, if you create it with a DOS text editor, such as EDIT.COM. The small arrow (<–) after the "prompt $" (representing the escape character) is in EDIT.COM with one character. In most text editors, the escape character is produced by pressing [Ctrl-P], then the [Esc] key. The escape character may be produced and displayed differently, depending on the text editor you are using.

In the first "prompt $" line, the "0;59" indicates that the [F1] key is assigned the quoted text "INTERSVR C: D:". The 13p indicates that the [Enter] key is added after the text. Some of the lines use 32p at the end, which indicates that a space is added after the quoted text, which allows the operator to add other parameters. Then the [Enter] key can be pressed after all parameters are added.

After mapping these keys and executing the code, you can press any of those mapped keys and the quoted text will result, followed by the character at the end of the line (13p, 32p, etc).

Using ANSI.SYS

ANSI.SYS defines functions that change display graphics, control cursor movement, and reassign keys. The ANSI.SYS device driver supports ANSI terminal emulation of escape sequences to control sequence of ASCII characters, the first two of which are the escape character (1Bh) and the left bracket character (5Bh). The character, or characters, that follow these two preliminary characters, specify an alphanumeric code that controls a keyboard or display function.

When you redefine keys, using ANSI.SYS, place the "echo on" before the "prompt" statements, so all escape sequences are echoed to the screen; therefore executing the redefinitions through ASNI.SYS functions. There are other switches allowed with ANSI.SYS. See DOS help on ANSI.SYS for information.

▶ **NOTE:** *ANSI.SYS distinguishes upper- and lowercase letters. For example "P" and "p" are different.*

/X Remaps extended keys independently on PC-compatible keyboards.

Memory Overview (PEN*KEY 6000 Series Computer)

The following paragraphs describe the operation of Windows for the PEN*KEY 6000 Series Hand-Held Computers and examines some noteworthy design aspects, to help you understand the concept. Some of these topics include:

- A little background information about memory.
- A high-level view of memory, as it applies to the PEN*KEY 6000 Series computers compared to a standard desktop PC:
  - The typical memory organization of a standard PC, including a pictorial view of memory, and how each area of memory is used.
  - A summary of the different types of memory.
  - A list of the commands and statements used in the CONFIG.SYS file and the AUTOEXEC.BAT file, and descriptions of their usage.
  - Descriptions of Windows, Storage Devices and Memory, as it relates to the PEN*KEY 6000 Series Hand-Held Computers.
  - How the PEN*KEY 6000 Series system works.
  - Standard mode vs. Enhanced mode.
  - RAM drive integrity-protection.
  - Some non-Windows Systems.

Background

When a typical PC is first turned on, it performs a cold boot. Many pieces of software are loaded into the PC's executable memory, or RAM, during the cold-boot. The first thing that loads is the BIOS. On a notebook or desktop PC, the BIOS is usually stored in a ROM chip on the main CPU board (motherboard). After the BIOS is loaded, DOS is loaded into the random access memory (RAM). In a machine that is set up to run in a Windows environment, Windows is the next item loaded. Finally, an application is selected to run, which is loaded into the RAM.

Standard PC Memory Overview

While installing programs, if you load some of them into the Upper Memory Area (UMA), this helps free up some conventional memory for use by your application. The following overview should provide some insight into how memory is organized and what each part is for.

To better understand the usage of these memory areas, refer to the diagram below, while reading the information that follows.

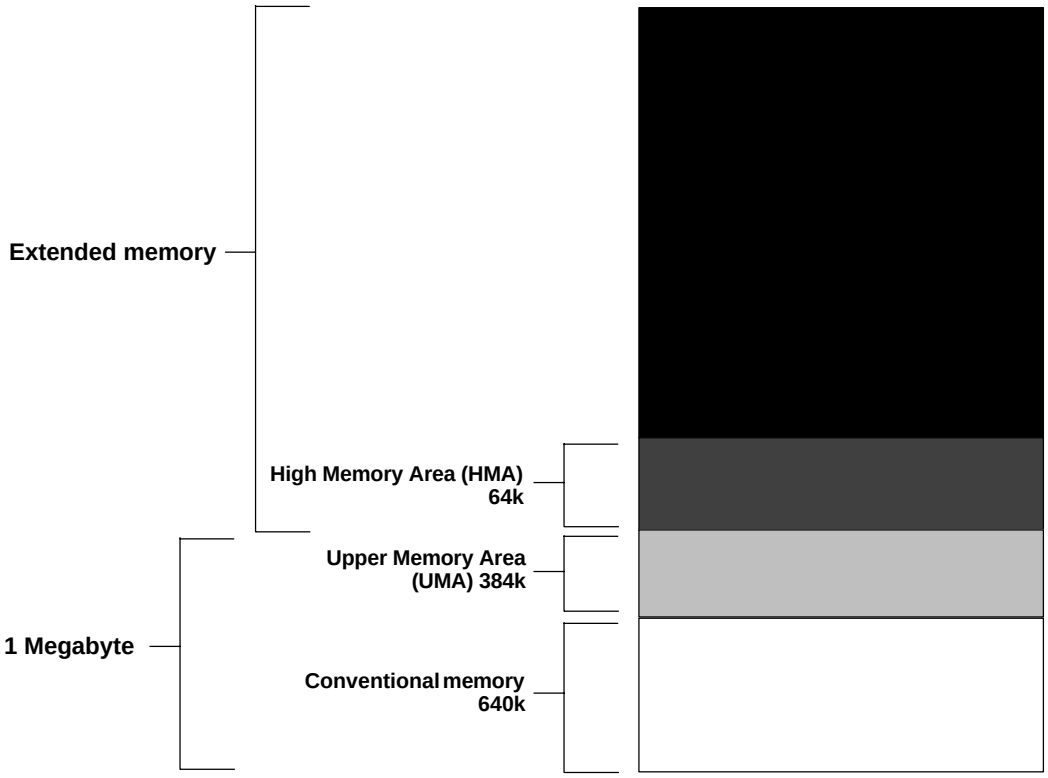


Figure B-1
Typical Memory Organization

Definition of Terms

Conventional Memory – That area of memory in which most DOS applications run. Other programs that could reside there include portions of DOS and any programs called using a “device=” or “install=” line in CONFIG.SYS or any programs that are not executed with a “loadhigh” from AUTOEXEC.BAT. Conventional memory can be up to 640 KB.

UMA – Just above conventional memory. UMA provides memory for system hardware. Any memory not used by the system is usable. The device=elanump.sys. . .” and “DOS=UMB” lines, in CONFIG.SYS, provide access to UMA. After ELANUMP.SYS is loaded and “DOS=UMB” specified, other drivers are loaded into UMA using “devicehigh=” in CONFIG.SYS or “loadhigh” in AUTOEXEC.BAT. This combination of conventional and upper memory consists of 1 MB, meaning there is 384 KB of memory for UMA.

Extended Memory – Above UMA and includes HMA. To access extended memory, add "device=d:\himem.sys" to CONFIG.SYS (where "d:\" is the drive and directory where HIMEM.SYS exists). The most widely known application using extended memory is MS Windows.

HMA = Memory residing above conventional and upper memory. If "DOS=HIGH" is specified in CONFIG.SYS, this loads part of DOS into HMA. This is possible if HIMEM.SYS is loaded before the "DOS=HIGH" statement. This frees up some conventional memory but lowers the memory used for the RAM drive by 64K bytes, because HMA has the lower 64K of extended memory.

Expanded Memory – Type of memory simulated by using a program such as EMM386.EXE. Applications created with the PenRight! development environment are examples of programs that optionally use extended memory. EMM386 can use UMA and extended memory to simulate expanded memory.

Summary of Memory Types

- *Conventional memory* — First 640K of memory where most applications run.
- *Expanded memory* — Type of memory simulated using EMM386.EXE.
- *Extended memory* — All memory above 1 MB.
- *HMA* — Lower 64K of extended memory with DOS loaded.
- *RAM drive* — Section of extended memory, simulates a disk drive.
- *UMA* — 64 KB section located above conventional memory, where drivers are loaded to free space for applications running in conventional memory.

Statements and Programs (CONFIG.SYS, AUTOEXEC.BAT)

Statement	Description
DEVICE=	Loads or installs device drivers into conventional memory from CONFIG.SYS
DEVICEHIGH=	Loads or installs device drivers into UMA from CONFIG.SYS
DOS=HIGH	Loads a portion of DOS into HMA when called from CONFIG.SYS
DOS=UMB	Allows programs to run in UMA when called from CONFIG.SYS
ELANUMP.SYS	Provides access to UMA when called with "device=" in CONFIG.SYS
EMM386.EXE	Simulates expanded memory when called with "device=" in CONFIG.SYS
HIMEM.SYS	Provides access to extended memory when called with "device=" in CONFIG.SYS
INSTALL=	Runs an application in conventional memory from CONFIG.SYS
LOADHI	Runs applications in UMA when called from AUTOEXEC.BAT

Windows, Storage Devices, and Memory

In a PEN*KEY 6000 Series system, DOS, Windows, and the application are loaded into RAM from a disk device. In Figure B-2, the system shown has four megabytes of RAM and a hard disk drive. In this system, all of the software components are stored on the hard drive, along with any application data.

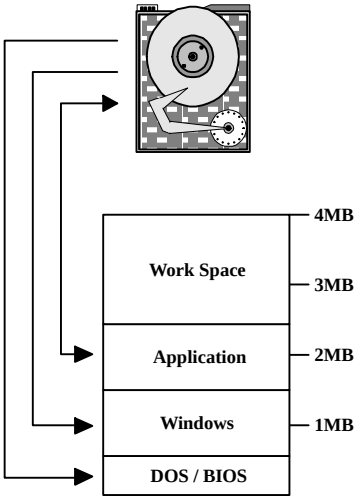


Figure B-2
Desktop/Laptop PC with Hard Disk

The system depicted in Figure B-2 represents the standard desktop or laptop PC. All of the software components (DOS, Windows) and the application are loaded from the hard drive into RAM, as indicated by the arrows in the diagram. The box labeled "Work Space", is part of RAM used by the system for short-term storage of information. The sizes of each area of memory shown, do not necessarily represent the actual implementations.

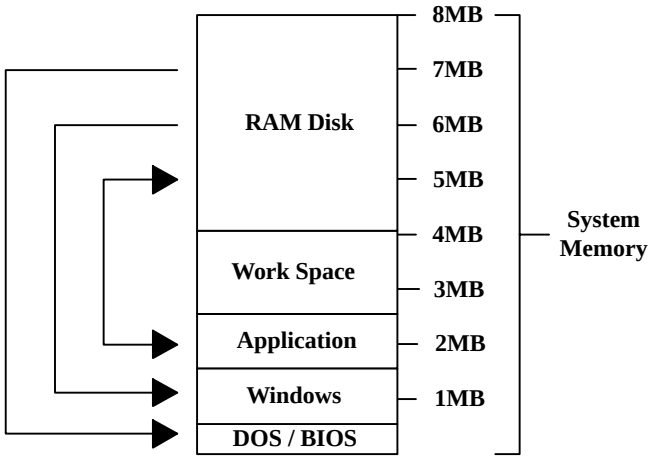


Figure B-3
PC with RAM Disk

The PEN*KEY 6000 Series system, depicted in Figure B-3, is like the standard desktop or laptop PC, except that a physical hard disk drive is not used for storage. Instead, a section of the RAM memory is defined as a *RAM disk*. The only difference between the systems of Figure B-2 and Figure B-3 is the storage location for software components. In a normal PC, this type of system would work only if the power to the RAM was never removed. By its nature, the RAM memory loses (or "forgets") its contents when the power is removed. However, in the PEN*KEY 6000 Series Computer, the RAM is powered continuously. Consequently, the RAM disk retains its contents over long periods of time.

▶ **NOTE:** *In the illustrations, 4- or 8-megabyte memory models are shown. Keep in mind, as the PEN*KEY 6000 Series systems continue to be enhanced, that future releases may allow larger memory models. The double-headed arrow from the "Application" to the "disk" indicates that the application loads the data from the disk, then writes it back to the disk.*

How the 6000 Series PEN*KEY System Works

In the system depicted in Figure B-4, DOS and BIOS are stored in the flash memory. Flash memory is similar to RAM memory, except that flash memory does not "forget" its contents after the power has been removed.

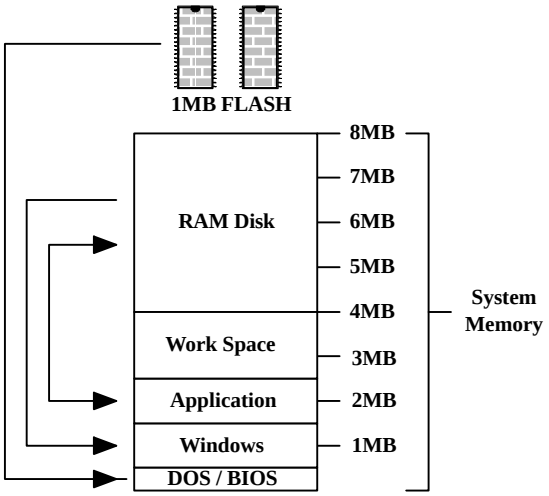


Figure B-4
PEN*KEY with RAM Disk

In the PEN*KEY 6000 Series system, shown in Figure B-4, the RAM disk stores Windows and the application, along with data from the application.

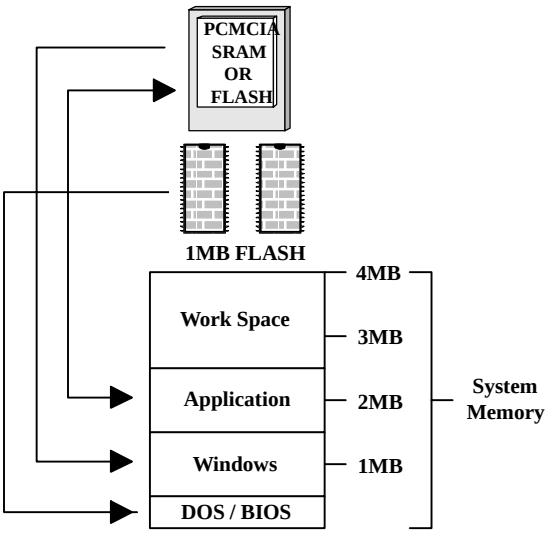


Figure B-5
PEN*KEY with PC Card

In the configuration shown in Figure B-5 above, a PC Card is used instead the RAM disk. The system considers the PC Card as a disk drive. There are several types of RAM and flash PC Cards that can be used in this configuration.

Standard Mode Versus Enhanced Mode

Windows can run in either *standard* mode or *enhanced* mode. The enhanced mode, better known, was created by Microsoft to provide *virtual DOS boxes*, which make it possible for Windows to run DOS programs within the Windows environment. Virtual DOS boxes insulate DOS programs from the hardware, in essence simulating a DOS-managed PC. The main drawbacks to enhanced mode are slower execution speed and larger code size. Despite its name, the enhanced mode is the slower of the two modes; and the size of the Windows code required to support enhanced mode is up to 1 megabyte larger.

The standard mode, which is the native mode for Windows, does not support DOS programs. But the standard mode is significantly faster and smaller than the enhanced mode. Normally, the PEN*KEY 6000 Series Computer runs in *standard* mode only. This means that you cannot utilize both a DOS application and a Windows application in the same environment on the computer. If Windows is running on the PEN*KEY 6000 Series Computer, all applications must be Windows applications. If DOS applications are needed, the PEN*KEY 6000 Series Computer should be operated in the DOS environment. For the application developer, this is critical. Some existing applications may require enhanced mode to run; this is probably because of the following:

1. Applications were originally designed to run under DOS and were later ported to Windows.
2. Applications make use of the Windows 32-bit memory model.

RAM Drive Integrity-Protection

The *integrity-protection* feature is an implementation that protects RAM from instructions that might improperly try to write data to the wrong location. If data were written to the wrong location, a corrupted data file, or even a lockup, could occur. In the PEN*KEY 6000 Series Computer, the RAM drive is protected from that situation. This protection is implemented in the system software (BIOS). Even in the event of a reboot, the RAM drive is protected from loss. This feature enhances the fail-safe operation of the computer.

Non-Windows Systems: PenPal and PenRight!

All implementations of software systems on the PEN*KEY 6000 Series Computers are disk-based. If a specific application is written in PenPal, for example, the same information shown previously in Figure B-2 through Figure B-5 still holds true. The only difference is that, instead of Windows being loaded over DOS, components of the PenPal runtime software are loaded.

Windows Environment

This part of the *Guide* serves two purposes. First, it presents a description of the Microsoft Windows environment to help you to become acquainted with the various Windows components. Second, it describes an approach to determining the memory requirements for a Windows-based PEN*KEY 6000 Series system.

A Brief History of Microsoft Windows

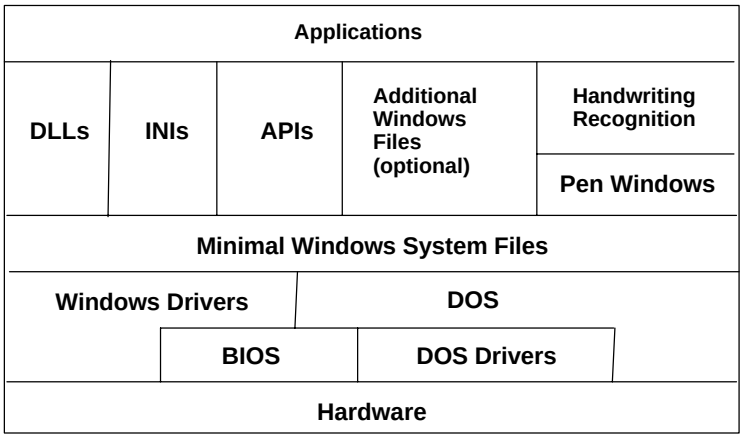
The design for the Windows environment was made for a better user interface. The Macintosh environment had gained significant popularity with many potential customers who felt that the DOS user interface was clumsy and hard to learn. While the Macintosh environment was built without concern for existing applications from previous Apple products, the Windows designers knew that, to succeed within the PC world, the Windows environment must address both the existing DOS applications and the Windows applications being planned. Further, to accumulate the largest audience, the Windows environment must operate on machines that are already installed with the DOS operating system.

While the Windows version 3.0 environment was considered by many to be the first *real* version of Windows (because it worked!), two previous versions had been previously built and released. These versions (versions 1.0 and 2.0) were slow and poorly designed. Windows 3.0 combined an acceptable user interface for the non-technical user who had the ability to use the full capability of the Intel 386 processor. It was in this environment that Windows gained great popularity and success. The main lesson learned by many who adopted Windows early was that Windows required tremendous system resources in terms of processor speed, RAM, disk space, and video capability.

As you become more involved with products like the PEN*KEY 6000 Series Computer, you will find that a working knowledge of the Windows environment is critical to your success in implementing these systems.

Windows Architecture

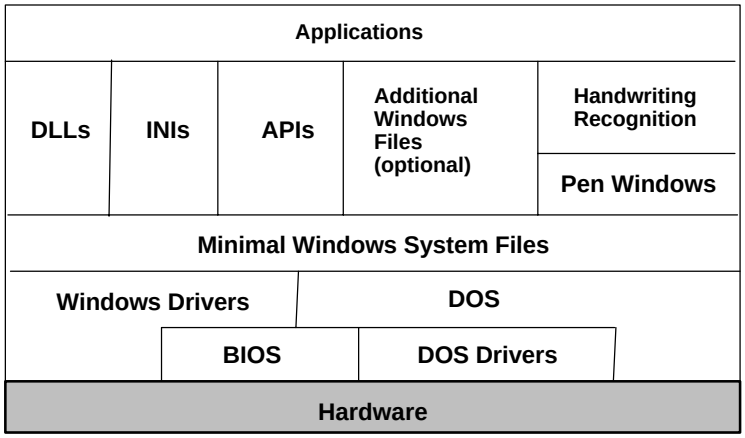
To understand the Windows environment for the PEN*KEY 6000 Series computers, you must look at all of the pieces that make up a PC that has a Windows environment. Below is a block diagram that identifies the significant components of a Windows-based machine and illustrates the way in which they interact. In this diagram, the *Microsoft Windows for Pen Computing* components are explicitly shown, although they are optional.



These paragraphs describe each component shown with each diagram.

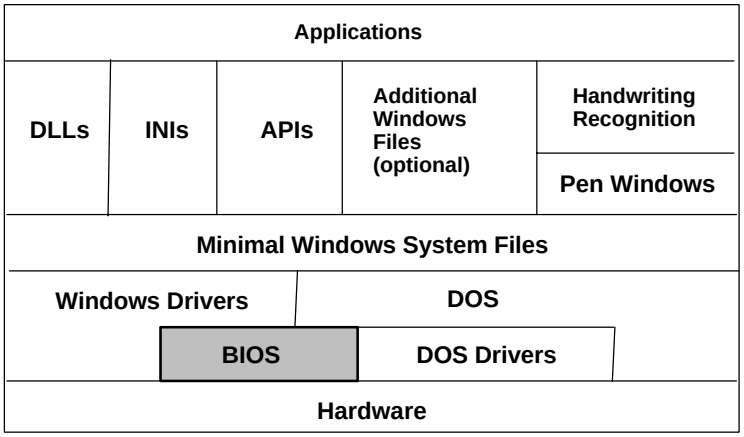
Hardware

Starting at the bottom of the following diagram, the hardware layer represents all of the components that are required to create an operable computer system. These components include, at a minimum, a processor, RAM, disk system, video system, and user input devices such as a keyboard or a pen.



BIOS

In its simplest form, the basic input/output system (BIOS) is composed of software that allows the operating system (in this case, DOS) to interact with the PC hardware. The BIOS may also be known as "firmware" or system software. The BIOS, in most desktop PCs, is typically contained on ROM chips. In the PEN*KEY 6000 Series Computer, the BIOS is programmed into flash memory to allow you to periodically update the system, without physically opening the unit to replace any of the chips.

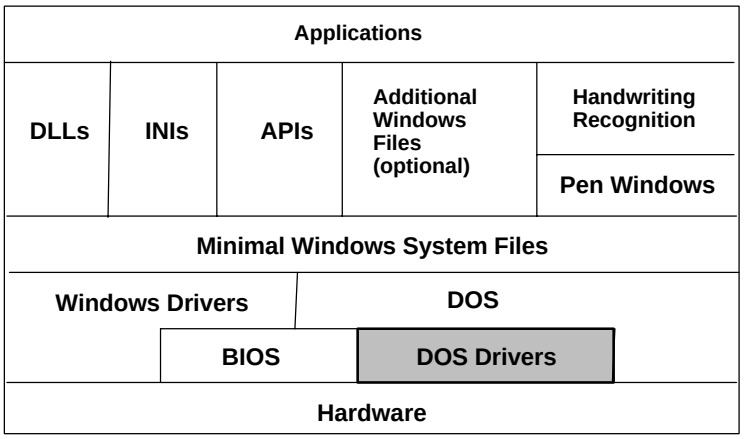


The BIOS manages the system through the use of interrupts, which are basically signals generated by the hardware to indicate that they need the attention of the CPU. For example, when you type characters at the keyboard, your key presses are changed into binary strings and sent to the BIOS. When the BIOS receives the string, it sets an interrupt to alert the CPU that it has data to be processed. All of this happens in just a fraction of a second.

From the software's point of view, BIOS is sometimes referred to as a Hardware Abstraction Layer (HAL). HAL is an important concept, in view of the fact that, an application that is written for a computer with HAL can be easily moved to another computer where the hardware is different. For those peripherals that are supported directly by BIOS, software developers need not be concerned about differences between hardware from various manufacturers. Hard drives are one example, since all these devices look the same *upstream* of BIOS. Between BIOS and the hardware, using the hardware device is often a matter of setting parameters for BIOS. This is commonly the case with hard drives, where the user must first set parameters in BIOS before the hard drive can be "seen" by DOS. Hardware that is not directly supported by BIOS, such as a PC Card drive, must be supported by a BIOS extension (known as a DOS *device driver*).

DOS Device Drivers

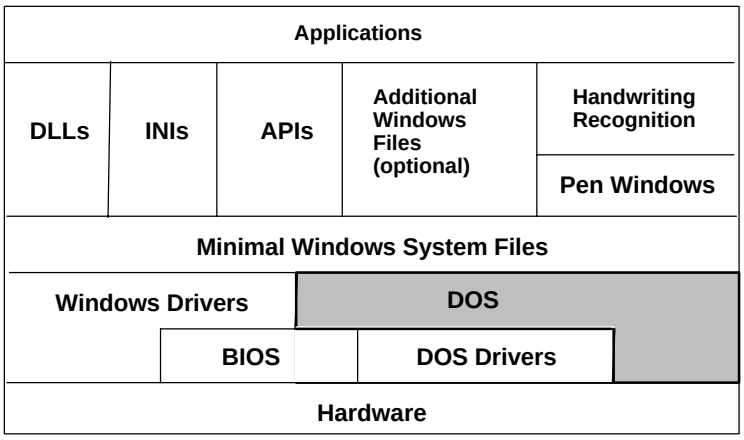
DOS device drivers tell the operating system about specific PC hardware components that are not directly supported by BIOS. Device drivers are loaded into memory at boot time; and they enable peripherals such as CD-ROM drive or memory above 1 megabyte. The PEN*KEY 6000 Series Computer requires at least one device driver, HIMEM.SYS, to control access to the high memory area (HMA); the first 64K RAM above the 1-megabyte boundary. Other DOS device drivers may be loaded, depending on the requirements of attached peripherals. DOS device drivers are always loaded into memory from the CONFIG.SYS file, which is located in the root directory of the default boot drive.



The default configuration in the PEN*KEY 6000 Series Computer not only includes the device driver HIMEM.SYS (in CONFIG.SYS), but also several drivers for PC Card services, allowing for the use of flash memory and modem cards in the PC Card slots. The exact contents of the CONFIG.SYS file depends on the requirements of the individual application and the peripherals selected.

DOS

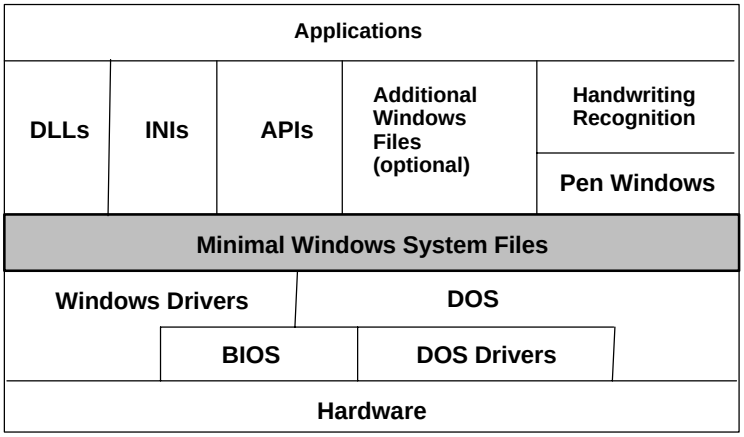
DOS is an acronym for Disk Operating System. It provides file services to the application programs, along with generic support for video and printing. DOS does not provide support for specific displays or printers, nor does it provide services for other peripherals. These services are *not* common to all applications written for the DOS environment. Each application written for a DOS PC must provide its own drivers for most of the peripherals that the application uses.



The actual DOS only consists of a small number of files, including an I/O interface and a command interpreter. The DOS environment that is installed in a typical desktop PC includes a large number of utility programs (e.g., CHKDSK.EXE) which are not required for the actual booting of a DOS PC, but are supporting application files. Note that these additional files are not loaded into the PEN*KEY 6000 Series Computer in the default configuration, but are available in the Tool Kit if needed by an application.

Windows System Files

Windows is an *operating environment*, a special class of DOS application whose primary purpose is to enable other applications to execute.



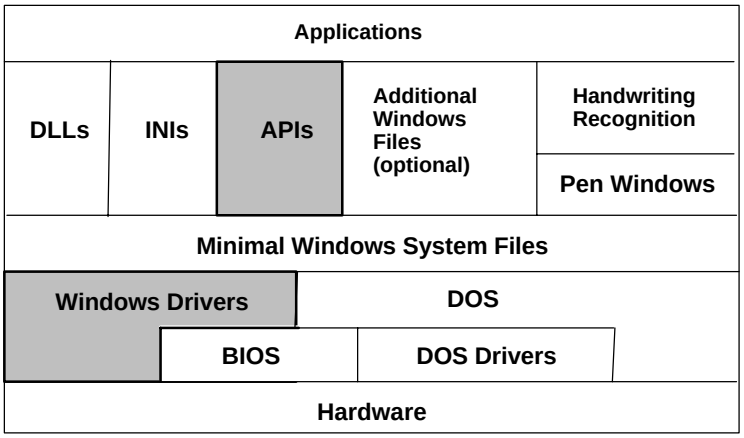
The most visible portion of Windows is the Graphical User Interface (GUI), but the real power of Windows is actually hidden deep inside the computer. Windows also acts as a multitasking system, which means that any number of applications may be loaded into memory and executed all at the same time. However, only a single application would be receiving CPU time at any given moment.

In a typical PEN*KEY 6000 Series Computer configuration, the Windows environment is practically invisible to the end user. So why have it at all? Windows becomes an intermediary between the application and the hardware. Unlike a DOS application that needs to supply its own code to interact with the display and other peripherals, Windows applications need only conform to the general parameters of the device. The intricacies of dealing with the device are handled by a Windows driver, letting the application focus its attention on other tasks.

The retail version of Windows is delivered on several diskettes and requires from 6 to 8 megabytes of hard disk space for installation. There are a vast number of user configurations that can be built with Windows. Many of the Windows components are optional and their uses are dependent on the type of peripherals (sound cards etc.) installed on a given PC. Intermec Technologies Corporation provides a standard minimal configuration for Windows in the PEN*KEY 6000 Series Computer. This configuration provides only those elements that are absolutely necessary in order to start Windows and load the File Manager. Many of the "services" assumed to be present by a given Windows application are not present and must be installed separately. The *Windows Components* paragraph, in the *Supporting Windows Applications* section, provides a list of the files included in the minimal Windows configuration, along with a brief description of each file's purpose. Few, if any, of these files may be removed and still allow Windows to run.

Windows Device Drivers and APIs

The device drivers and the Application Programming Interfaces (APIs) are the real power of Windows as an operating environment. The book, *The Mother of all Windows Books*, by Woody Leonhard and Barry Simon (Addison-Wesley, 1994), does an excellent job of explaining the role of a device driver.

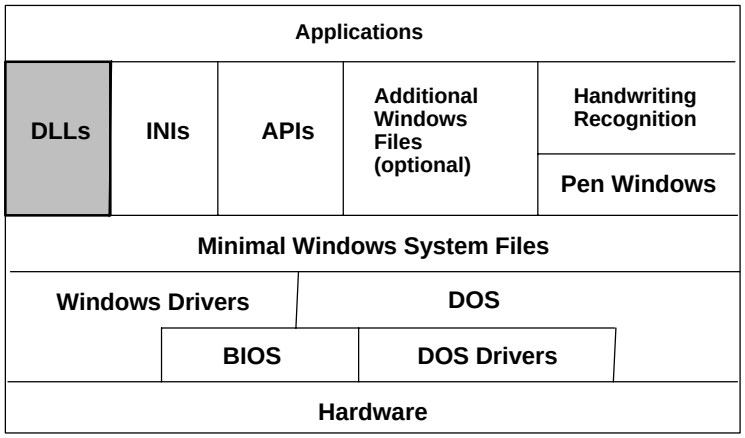


Windows device drivers have a filename extension of DRV. Like the DOS device drivers, the Windows drivers are responsible for controlling access to the peripherals. But unlike DOS, Windows *must* have drivers that become intermediaries for all peripherals. Windows does not allow the application to access the hardware directly; that is the job of the device driver. By following this approach, the application does not need to concern itself with details of the display or other peripheral; as long as it conforms to the API, it works with any device driver. Windows device drivers are typically provided by the hardware manufacturer. However, Windows comes with a small number of generic drivers that are compatible with the basic features of many brands of peripherals.

In the Windows environment for the PEN*KEY 6000 Series Computer, the required device drivers are provided with the minimal configuration. These include the Microsoft communications, the keyboard, the system, the sound drivers; the Pen Windows mouse driver; and the NORAND display driver. To the applications, the display appears as if it was a standard CGA; and the driver handles the actual placement of pixels on the platform specific screen, as well as rotation from landscape to portrait modes.

DLLs

Look in the Windows System directory on your PC, and you will find many files with the extension *DLL*. These are *dynamic link libraries*. These are nothing more than *subroutines*.

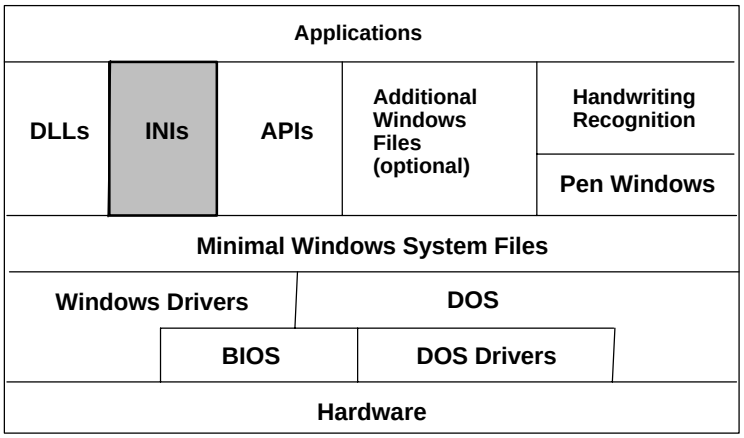


In computer programming, subroutines are subsets of computer code separated from the main program, so that the subset can be reused many times by different parts of the program. The word *dynamic* describes the key difference between the DOS subroutines and the Windows DLLs. The dynamic aspect of a DLL means that the DLL is loaded into execution memory only when it is needed and is removed when its job is done. DOS subroutines are bound to the DOS application program. If two programs on a DOS PC use the same subroutine, each of the DOS programs contains its own copy of the subroutine. The DOS program subroutine is always present in executable memory even if it is not needed at any given time.

Any application that conforms to the API of a specific DLL may use the DLL, and avoid duplicating the logic within its own application. One of the most widely used DLLs is COMMDLG.DLL, which is supplied with Windows. This subroutine library provides many common Windows dialog boxes. Have you ever noticed that the File Open dialog box looks the same across many of your Windows applications? This common look occurs because the dialog is invoked through a call to COMMDLG.DLL. The application itself does not even *have* a File Open dialog! Giving the Windows applications a consistent look and feel, is one of the key benefits of these shared subroutine libraries.

INI Files

Files with the extension of INI are the initialization files for Windows and associated applications. They are the heart and soul of the Windows environment.



A small number of initialization files contain the key information that Windows needs in order to run. Many applications also rely on the INI files to remember user preferences, customization, etc. These INI files are usually plain ASCII text files, containing lines that follow a standard format:

```
[section name]
variable=value
```

The INI file may have one to several hundred section names, and one to hundred variables under each section. Through simple API calls, the programmer can easily reference any INI file and obtain settings for the various variables listed. Editing an INI file manually can be dangerous; you should always make a backup copy before changing *anything* in an INI file.

The main initialization file, WIN.INI, resides in the Windows startup directory, and tells Windows some important information, such as where the fonts are located, how the windows need to look, the speed of the mouse, etc. Many applications create sections of their own within WIN.INI, as well. A thorough treatment of WIN.INI is beyond the scope of this section of the document; information can be obtained from various outside sources, including some entire books devoted to the subject.

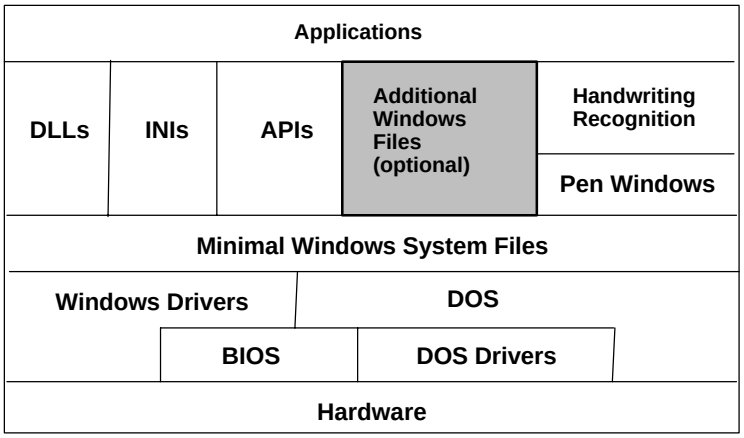
The other key initialization file is the SYSTEM.INI file. This file contains information relating to the hardware configuration, including the names and locations of the device drivers. This file is also where the shell program is identified (see below).

There are often additional INI files present in the system. Pen Windows, for example, has its own INI file that indicates the recognizer in use, and identifies the user. The File Manager's INI file saves the current values of any option settings, such as the format of the display, the font, etc. It is generally considered good programming practice for an application to create its own INI file rather than to cause the WIN.INI file to become too large and complex. The Windows environment can get quite ugly if its WIN.INI gets too large.

And one final point about the INI files: if you damage one, the corresponding application may wander around a bit before it gets its bearings. If you damage the WIN.INI file or the SYSTEM.INI file, the Windows environment may become unstable, or it may not run at all.

Additional Windows Files

While they are not shown explicitly in the block diagram, there are other files that must be present for a Windows application to run. Some applications require only the main executable (or .EXE file), but most require the .EXE, one or more DLLs (either the type supplied with Windows or DLLs specific to the application), as well as data files. If one or more required files are not present, then Windows fails to load the application, and offers the following "helpful" error message: `Cannot find file or one of its components.`



If you see this message when you double-click on an application's filename in the File Manager, it usually means that one or more DLLs are either not present or not in a location on the disk drive that is accessible via the path. Unfortunately, Windows does not provide much assistance in helping you to determine exactly which files are missing (the above message is an example). Determining which files are missing is largely a matter of trial and error, unless you are able to determine exactly which files the application is expecting to have available on the disk. For assistance in determining what these files are — in reference to an application that is documented in this programming guide — see the appropriate paragraph (for that application) in this section of the document.

Shell Applications

The *shell* is the first program that Windows starts, and it is the program that causes Windows to exit when it shuts down. By default, Windows uses the Program Manager as the shell. Ordinarily, almost any application can be designated as the shell, and one of the basic requirements of a shell program is the ability to launch other applications. Rather than require special programming on the part of the customer to implement shell-like capabilities, the PEN*KEY 6000 Series Computer uses an "invisible" shell program that, in turn, invokes the main application program. To see how the File Manager is launched from the shell program, refer to the *NORAND Windows Shell* paragraph, in the *Supporting Windows Applications* section.

Fonts: What They Are and How They Impact

The retail Windows program comes with several fonts. There is also a considerable aftermarket for add-on fonts. Fonts are usually in one of two categories:

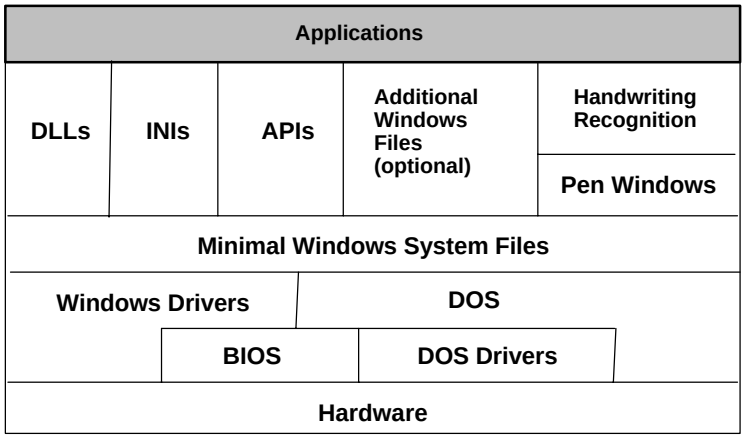
- Bit-mapped fonts have a filename extension of .FON. At least one of these fonts must be present for Windows to use in building dialog boxes and menus. The NORAND minimal Windows configuration includes three bit-mapped fonts: VGAFIX, VGAOEM, and VGASYS.

- TrueType fonts consist of two files each, with extensions of .FOT and .TTF. The .FOT file is actually a Windows DLL that contains, among other things, the path name for the TTF file, which contains the actual font. The retail version of Windows comes with a dozen or so TrueType fonts, including Times New Roman, Arial, and Wingdings. The minimal PEN*KEY 6000 Series Computer configuration contains *no* TrueType fonts.

If a customer application uses TrueType fonts, or bit-mapped fonts, other than those listed above, they must be added to the Windows configuration. If a font file, needed by an application, is missing, then an error is generated and Windows substitutes the "closest" font it can find. This can lead to some interesting results when the variable-pitch system font is substituted for a fixed-pitch font (like Courier New).

Applications

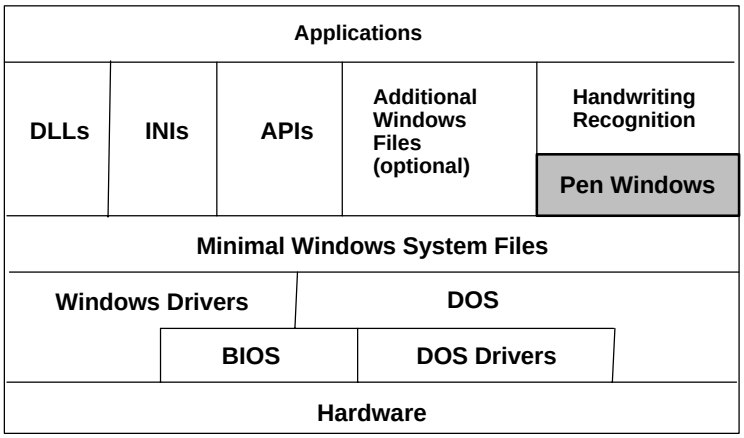
Windows applications include the main executable program (with the .EXE extension), the device driver files (with the .DLL extension), and other files that are required for the main program to execute.



Often, you will find that an application's executable file is much smaller than the functionality of the program would indicate. For example, the Microsoft PowerPoint V4.0 main program is only 97K. Much of the functionality is implemented in DLLs (PowerPoint includes one that is over *3 megabytes*). Therefore, it is important that you get the full picture when sizing an application for the PEN*KEY 6000 Series Computer, by including allowances for any DLLs and other files needed by the main program. It is also significant to note that certain development environments require a huge library of routines to create an application that does *anything*. Power Builder, a prototyping tool, is one such program. The overhead for creating a simple screen is around 3 megabytes.

Pen Windows Files

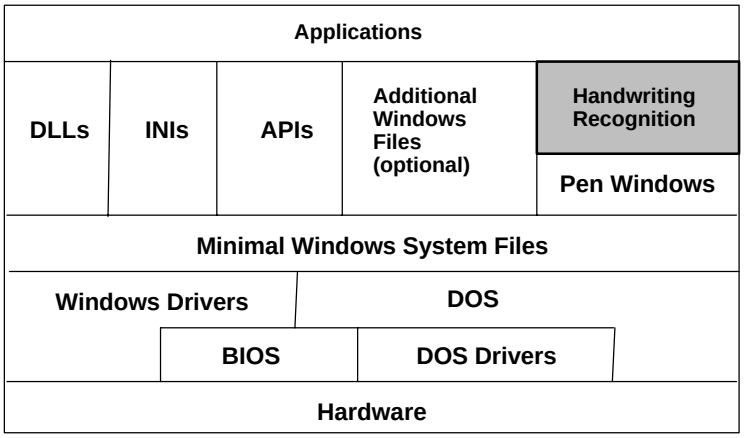
Pen Windows, properly known as, *Microsoft Windows for Pen Computing*, is an extensions of the base Windows environment. It provides support for the pen as an input device. Generally speaking, Pen Windows is not a prerequisite for using a stylus as a navigation tool or for "ink" capture. The pen extensions are actually needed only when the ink must be translated into machine-readable text characters.



Pen Windows adds only a few files to the standard Windows configuration. Basic added functionality includes enabling pen input for non-pen-aware applications, and handwriting recognition (see below). The manufacturer of pen-based hardware devices may supply Pen Windows to users, although Pen Windows is not available to end users by itself from Microsoft or OEMs. The assumption is, "if you need pen support, you also need the hardware". While true for end users, this policy can be a frustrating Catch-22 for programmers who may need to start development of pen-based applications, but do not have a pen-enabled development system. There are several inexpensive peripheral devices available from third parties that add pen input to desktop systems and are delivered with Pen Windows. One of these devices is CIC's *Handwriter for Windows*.

Handwriting Recognition

Handwriting recognition (HWX) refers to the process of converting handwritten electronic ink into machine-readable character strings. Once the driving force in pen computing, HWX has taken a secondary role to the "pencentric" application; that is, an application optimized for use with a pen. With Pen Windows, Microsoft offers a handwriting recognition system known as *MARS*. It gets the job done, though with far less speed and accuracy than most users find acceptable. OEMs, such as Intermec Technologies Corporation, have the option of licensing Pen Windows without the MARS recognizer.



Customers purchasing the NORAND Pen Windows license may also license the handwriting recognition systems (Communications Intelligence Corp. (CIC)) from Intermec Technologies Corporation, at additional cost.

Norand Value Adds

In addition to Windows-specific value added software provided by Intermec Technologies Corporation, there are several BIOS- and DOS-level programs licensed with the PEN*KEY 6000 Series Computer. The following is a list of software included and a brief description of the software's value.

BIOS (Basic Input Output System)

This is the low-level software that allows operating systems, such as DOS and Windows, to work on the PEN*KEY 6000 Series Computer. This BIOS is unique because of the additions that were made to support power management. Standard BIOS on PCs do not have the functionality available to implement the level of power management that is present in the PEN*KEY 6000 Series Computer.

SystemSoft Card and Socket Services

This software, licensed from SystemSoft, implements the software portion of the PC Card specification that allows use of numerous forms of PC Cards in the PEN*KEY 6000 Series Computer. It is key to allowing a customer to change cards over time, as technology moves forward. A computer with a PC Card slot that does not have this software requires custom software drivers for each card the user requires.

NORAND Card and Socket Services

This software, licensed from Intermec Technologies Corporation, supports AMD ELAN PC Card controller. Refer to *Section 1, Getting Started* for details.

Power Management

This software is one of the key value adds for the PEN*KEY 6000 Series Computer. It implements power saving features not available in the industry-standard APM specification. The technology enabling this level of battery life is protected under patents and filings registered to Intermec Technologies Corporation.

NORAND Utilities

This application, unique in the PC world, boots and configures the PEN*KEY 6000 Series Computer with a variety of communications options. The NORAND Utilities are loaded into the flash memory of the PEN*KEY 6000 Series Computer in the factory. The communications capabilities of ACN are the pipeline for the remote diagnostics available in the PEN*KEY 6000 Series Computer. NPCP, IrDA, and TTY communications drivers are included within this application.

Pen Drivers

Pen Drivers for Windows, PenRight, or Pen Pal are included. These drivers are required to implement character recognition, along with signature capture on the PEN*KEY 6000 Series Computer. The BIOS does NOT include the actual character recognition software, which is a separate item, licensed if needed.

Scanner Drivers

Laser Scanner Drivers for Windows are included. These drivers are required to implement scanning of bar codes on the PEN*KEY 6000 Series Computer.

NPCP Printer Drivers

NPCP Printer Drivers for Windows are included. These drivers are required to implement printing using NPCP on the PEN*KEY 6000 Series Computer.

IrDA Printer Drivers

IrDA Printer Drivers for Windows are included. These drivers are required to implement printing using IrDA protocol on the PEN*KEY 6000 Series Computer.

PEN*KEY 6000 Series Memory-Sizing Guidelines

After reading the last few pages, you can probably surmise that selecting the appropriate memory size, for the PEN*KEY 6000 Series Computer, is not always an easy task. Although you must consider many variables, the following few considerations may help your selection process.

All Systems

- You must allocate space for both execution and storage. The RAM Drive is an area of memory that is treated like a disk drive (storage), and cannot be used for execution RAM.
- Windows requires a minimum of 2 MB of execution RAM, but 4 MB is a more practical minimum. 8 MB of execution RAM should cover almost all application requirements.
- The minimal Windows configuration, without pen extensions, requires 1 MB of storage space; with pen extensions, this grows to about 2 MB.
- The 1-MB flash memory area in the PEN*KEY 6000 Series Computer is reserved for system and embedded software; there is no available space for program storage in flash memory.
- Err in the conservative direction. When estimating sizes, round up rather than down. This leaves room for enhancements.

Systems with RAM Drive Storage

- If using a RAM drive, you will trade off execution and "disk" storage space. First determine the amount of execution space needed and then use the balance for the RAM drive. Of course, the RAM drive must be large enough to contain DOS, Windows, the application, the data files, and the files created during execution. If the numbers do not add up, you need to consider an increase in RAM size.
- The RAM drive can only be formatted in 1 megabyte increments; but you should consider the minimum RAM size (for the PEN*KEY 6000 Series Computer) in order to run Windows applications without PC Card storage, is 8 MB; of which at least 4 MB should be devoted to execution space, leaving 4 MB for the RAM drive.
- You can have one RAM Drive, with a maximum size (as specified in the *RAM Drive* paragraph, in the *Getting Started* section of this publication).

Systems with External Storage

- The PEN*KEY 6000 Series Computer offers SanDisk and flash memory cards that can dramatically increase the space available for storing application files. A PC Card solution also eliminates the requirement for a RAM drive, freeing on-board memory for use as execution space.
- Even if you plan to have all files on a PC Card drive, you must be certain that you have enough RAM for execution. In these configurations, it is not possible to increase the size of executable RAM without hardware modification or replacement. You can easily add external storage, but you cannot easily add more RAM.
- Storing data on a PC Card is usually safer, and less prone to data loss.

General Index

NOTE:

This index covers all topics. Those in italics are figures, those in bold are tables.

NUMBERS

A

B

C

D

E

F

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Files Index

NOTE:

This index section is provided to assist you in locating descriptions for device drivers, applications, utilities, batch files, or other files within this publication.

SYMBOLS

NUMBERS

A

B

C

D

E

F

G

H

I

K

L

M

Interrupt Index

NOTE:

This index contains a complete list of interrupts defined in this publication, organized alphabetically by topic. You can find an interrupt with almost any noun in the title of the interrupt name, as well as some verbs. However, interrupts that use verbs such as "read," "write," "set," and "get" are listed only by noun.

The following parameters listed under "Number" are listed by their respective interrupt number in their name.

Page numbers in italics are figures, those in bold are table.

NUMBERS

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z