Intermec

Technologies Corporation

PEN*KEY^R *6400 Computer*

PROGRAMMER'S REFERENCE GUIDE

Acknowledgments

AIM is a registered trademark of Automatic Identification Manufacturers.

Borland and *dBASE* are registered trademarks of Borland International, Inc.

Intel is a registered trademark of Intel Corporation.

Microsoft, MS, and *MS-DOS,* and *Windows,* are registered trademarks of Microsoft Corporation.

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

Antenna Requirements

" NOTICE   FCC rules section 15.203 and Canada's RSS-210 require that this device be operated using an antenna furnished by Norand Corporation. The antenna coupling on this product has been designed to accept only antennas manufactured by Norand. Use of an antenna other than that furnished with the equipment is prohibited by FCC and Industry Canada rules.

Canadian Computer Compliance

This Class B digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Cet appareil numérique de la classe B respecte toutes les exigences du Reglèment sur le material boilleur du Canada.

Canadian Spread Spectrum Radio Certification

" NOTICE   This device complies with RSS-210 of Industry Canada. Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Canadian 2.4 GHz Radio License

" NOTICE   This device requires a radio license, unless it is installed totally inside a building. (Users must obtain this license.)

Une licence radio est requise pour ces dispositifs, sauf pour ceux installés tout à fait à l'intérieur d'un bâtiment. (Il faut que l'utilisateur obtienne cette licence.)

European Notice

The 902–928 MHz SST (Spread Spectrum Transmission) radio referred to in this manual is not available for sale or use in Europe (including, but not limited to, Great Britain, Italy, Germany, France, Spain, Norway, Denmark, Sweden, Finland, Portugal, and the Benelux countries). Any references in this manual to 902–928 MHz SST, or modules containing 902–928 MHz SST radios, should be disregarded by users of this product in Europe.

Compliance to EN50081-1 and CISPR22 Class B for emissions and EN50081-1 for immunity.

Telephone Installation Warning Notices

The following notices apply to equipment that may be connected to telephone lines or systems. For your personal safety, and to protect this equipment from potential electrical or physical damage, do NOT connect equipment to telephone lines or data communication equipment unless the following warnings have been read, understood, and complied with.

- " Never install telephone wiring during a lightning storm.
- " Never install telephone jacks in wet locations unless the jack is specifically designed for wet locations.
- " Never touch uninsulated telephone wires or terminals unless the telephone line has been disconnected at the network interface.
- " Use caution when installing or modifying telephone lines.
- " Avoid using a telephone (other than cordless type) during an electrical storm. There may be a remote risk of electric shock from lightning.
- " Do not use the telephone to report a gas leak in the vicinity of the leak.

Installation du téléphone : avertissements

Les avertissements qui suivent s'appliquent à tout équipement qui peut être branché aux lignes ou systèmes téléphoniques. Pour votre sécurité personnelle et pour protéger l'équipement de tout dommage électrique ou physique potentiel, NE PAS brancher un ordinateur tablette électronique ou ses périphériques aux lignes téléphoniques ou équipements avant que les avertissements suivants aient été lus, compris et observés :

"   Ne jamais installer de câblage téléphonique pendant un orage électrique.

"   Ne jamais installer de prise téléphonique dans un endroit humide à moins que la prise ait été spécifiquement conçue pour être utilisée dans les endroits humides.

"   Ne jamais toucher les fils de téléphone ou de l'équipement terminal non isolés à moins que la ligne téléphonique n'ait été débranchée de l'interface réseau.

"   User de prudence lors de l'installation ou de la modification de lignes téléphoniques.

"   Éviter d'utiliser un téléphone (autre qu'un appareil téléphonique sans fil) pendant un orage électrique. Il pourrait y avoir un faible risque d'électrocution par la foudre.

"   Ne pas utiliser le téléphone afin de signaler une fuite de gaz à proximité de la fuite.

B CAUTION: **Intermec Technologies Corporation suggests you buy cables from us to connect with other devices. Our cables are safe, meet FCC rules, and suit our products. Other cables may not be tested. They may cause problems from electrostatic discharge or induced energy. Our warranties do not cover loss, injury, or damage from other cables.**

B CAUTION: **Intermec Technologies Corporation recommends that you only purchase Norand Mobile Systems Division certified modems. Intermec does not certify all modems available in the marketplace. Intermec does not warrant noncertified modems; furthermore, these modems may cause problems from electrostatic discharge and may not conform to FCC regulations. For a list of Norand Mobile Systems Division certified modems call Customer Support at 1-800-221-9236 in United States or (country code) 800-633-6149 in Canada.**

WARNING: ***The lithium ion battery may explode if replaced incorrectly. Replace only with the same or equivalent type.***

ADVERTISSEMENT: ***La batterie au lithium peut exploser si elle est replacée de manière incorrecte. Elle ne doit être remplacée que par une batterie identique ou similaire.***

WARNING: ***Lithium ion batteries may explode or catch fire if overcharged due to improper dock installation.***

ADVERTISSEMENT: ***Les batteries au lithium peuvent exploser ou prendre feu si elles sont trop changées à cause d'une mauvaise installation de la station d'accueil.***

CONTENTS

SECTION 2

Supporting DOS Applications

SECTION 3

Power Management

SECTION 4

Communications and Device Support

SECTION 6

Reference, System Information

SECTION 7

Reference, Open Systems Publications

APPENDIX A

Sample Configuration Files

INDEX

General Index

Files Index

Interrupt Index

PREFACE

This publication has been written to provide assistance in developing, or porting, end-user applications for use on your PEN*KEY® 6400 Hand-Held Computer.

If you are unfamiliar with the structure of the 6000 Series Programmer's Reference Guides, be sure to read *Locating Information*, to simplify the task of locating topics of interest.

Topic Summary

Scope

The majority of the material in this publication pertains specifically to the 6400 Computer.

This information was developed, with experienced application programmers in mind. However, writing applications for this computer is similar to writing applications for most standard PCs, For any differences, refer to the *Introduction to the 6400 Computer* paragraph, in the *Getting Started* section.

For the New User

If you find that you need further assistance in developing applications, consider the available resources, such as those listed in the *Reference, Open Systems Publications* section.

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

Related Publications

The following publications may be of assistance in using or writing applications for your PEN*KEY 6400 Hand-Held Computer:

PEN*KEY® 6400 Hand-Held Computer User's Guide (961-047-093)

> How to setup, operate, and maintain the 6400 Computer, including terminal emulation menu screens for setting parameters.

SE 1223 Integrated VLD Scanner (961-032-042)

> Includes information about the scanner, and definitions of bar codes.

6710 Access Point User's Guide (961-047-081)

> How the 6710 Access Point operates on the open wireless LAN, and how to install and configure the access point.

PEN*KEY 6400 TCP/IP Client User's Guide (961-054-012)

> Utilizes the TCP/IP network protocol for your application's communication.

The following publications relate to Network Management with your PEN*KEY 6400 Hand-Held Computer:

Norand Management Information Bases Reference Manual (977-051-002)

> Description of the private Management Information Base (MIB) for open wireless LAN devices.

Open Wireless LAN with HP OpenView for Windows User's Guide
(961-051-009)

> How to install and use the OpenView application by Hewlett-Packard.

OWLView for HP OpenView for Windows User's Guide (961-051-010)

> Using OWLView to view open wireless LAN network topology and device status.

Structure of the Book

The following overview shows the structure of this publication.

Section 1, Getting Started

> Contains information to help you become familiar with your 6400 Computer. Topics include: an open system environment, differences between the 6400 Computer and a standard PC, the Tool Kit, some helpful tips, a typical development configuration, reprogramming flash memory, some development environments, set up for development from a PC, and a few development resources.

Section 2, Supporting DOS Applications

> Describes several DOS applications supported for the 6400 Computer, as well as printing and power management for DOS.

Section 3, Power Management

> Describes the Advanced Power Management (APM) supported on the 6400 Computer.

Section 4, Communications and Device Support

> Describes communications, upper-memory management, keyboard definition, and other descriptions of devices (including battery, beeper, display, and docks).

Section 5, BIOS Interrupt Support

> Lists the interrupts supported for 6400 applications.

Section 6, Reference, System Information

> Includes the DOS operating system, boot process, messages, audible alerts, IRQ maps, I/O maps, and system variables.

Section 7, Reference, Open Systems Publications

> Publications referenced from other places within this book, or some that are useful for developing applications for the 6400 Computer.

Appendix A, Sample Configuration Files

> Provides examples of configuration files for your 6400 Computer.

Index

> This section is partitioned into the following subsections:

> ***General Index***
> Consists of a variety of useful topics included in this publication.

> ***Files Index***
> A fairly comprehensive list of the file names, applications, and device driver that are described (or mentioned) in this publication.

> ***Interrupt Index***
> An alphabetically organized index to all of the interrupts defined in this publication.

Locating Information

This paragraph describes several documentation tools included in this publication, to assist you in navigating through the book.

Contents

The *Contents* section is a good place to begin searching for information in this publication, since it is an outline of the book.

Topic Summaries

There is a *Topic Summary* on the beginning of each section in this publication. It provides a (miniature) table of contents for the section. For example, the *Topic Summary* in the *Supporting DOS Applications* section provides an index to each of the DOS applications, including a topic title, the name of each application, and the page number where that topic is described.

Tabs

Since various paragraphs throughout the book refer to information in other sections, the "bleeding tabs" (located along the right margin) are provided to assist you in locating sections quickly.

Index

The *Index* section, at the end of this publication, also provides a quick method of locating information. It is divided into three subsections, as described above.

Locating Files and Applications

There are several ways to locate the information for a particular application, device driver, or other files:

- The **Files Index** (at the end of this publication) is probably the most useful source, because it is a fairly comprehensive list of file names that appear in the book.
- The **Contents** is another source for locating applications. If the topic describes an application, the name of that application is included in the paragraph title.
- The **Topic Summary**, (at the beginning of each section) is a mini-table of contents for the topics included in that section. If the topic describes an application, the name of that application is included in the *Topic Summary*.

There are numerous lists of files, throughout this publication. The following items point to each of those lists.

- **Table 1-2, DOS Device Drivers**, in *Section 1, Getting Started*, on page 1-18, includes the major device drivers supported for your 6400 computer.
- A few **Utility Programs** are briefly described in *Section 1, Getting Started*, beginning on of page 1-19.

Locating Interrupt Definitions

There are several ways to locate information for an interrupt:

The **Contents** section contains entries that can lead you to the general location where each set of interrupts is documented. For example: *INT 10h Display Services*, *INT 13h Disk Services*, *INT 15 System BIOS Services*.

The **Interrupt Cross-Reference** table, in the *BIOS Interrupt Support* section, is organized by interrupt number, and includes page numbers to the locations where these interrupts are defined.

The **Interrupt Index** (at the end of this publication) lists all of the interrupt definitions included in this book, organized alphabetically by topic.

Refer to the **Reference, Open Systems Publications** section, for the names of names of publications that provide additional information about general and special purpose interrupts.

Support Available

As you pursue your PEN*KEY 6000 Series computer development activity, you can obtain information and personal assistance from various resources.

This Programmer's Reference Guide pulls together, under one cover, many diverse and separate topics related to PEN*KEY 6000 Series development activity.

" NOTE: *Other related information is available from third-party suppliers. Many of these sources are identified in the Reference, Open Systems Publications section of this publication.*

Customer Response Center

For additional technical support, contact the Customer Response Center at 1-800-221-9236.

If you have any questions about PEN*KEY 6000 Series Computer development, contact the Customer Response Center for personal consultation and training.

Bulletin Board

The Customer Support Bulletin Board is a source for additional software and documentation:

" Phone number: 1-319-369-3515

" Protocol: Full duplex, ANSI or ANSI-BBS; 300 to 14,400 bps; v.32bis; 8 bits, no parity, 1 stop bit. If you are using a high-speed modem, disable XON/XOFF and enable RTS/CTS.

" If you are signing on for the first time, type "new" and follow the instructions. The BBS takes you through a new-user sign-up procedure.

Getting Started

This section contains information to help you become familiar with your 6400 Computer, so you can begin building applications.

If you are unfamiliar with the structure of the PEN*KEY® 6000 Series Programmer's Reference Guides, be sure to turn back to the introductory section and read *Structure of the Book* and *Locating Information*.

The following lists may be of further assistance in locating topics of interest:

An Open System Environment

The 6400 operating system consists of standard ROM DOS, version 5.0.

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. You have 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 programming experience, you will quickly feel comfortable with the PEN*KEY 6400 platform.

Introduction to the 6400 Computer

From the viewpoint of an application, the 6400 Computer is a PC compatible computer, with some exceptions, as described below.

How the Hardware is Different

The 6400 Computer is a ruggedized, ergonomic, battery powered computer, with input from (keyboard or scanner), backlit display, integrated communications, and various combinations of external and internal peripherals. It is designed for a mobile environment.

The system has no PC Card support, except for the dedicated local area radio usage. It has memory controllers, as well as bus management; and is powered by a smart battery pack.

Processor
AMD ELAN 386 (AM386SC300), 33Mhz

Display
The display is an LCD, and is normally used in landscape mode. The physical display is approximately the size of one eighth VGA (160x128 pixels). However, it uses a CGA controller. The physical display is a window (or viewport) into the full logical display. Refer to the *Display* paragraph, in the *Communications and Device Support* section, for additional details.

User Input
The keyboard consists of a built-in alpha-numeric keypad with control keys. Two keyboard options are available, a 41-key keyboard and a 51-key keyboard.

"   For the 41-key keyboard, function and numeric keys are single keystrokes. Alphabetical characters can be entered by using the green, blue, or yellow shift keys.

"   For the 51-key keyboard, alphabetic and numeric keys are single keystrokes. Special function keys can be entered using the green, blue, or yellow shift keys.

Power Management
Advanced Power Management (APM) is critical for determining useful runtimes. Power is obtained through external sources, the main battery pack and the backup battery. The main battery pack is a smart battery pack. The backup battery maintains data for approximately 100 hours, after the main battery has been removed or has completely run down.

System Memory
System memory is provided through DRAM and Flash memory modules.

"   **Flash** memory provides storage for executable and system files. The upper 256 KB of flash memory is reserved for the BIOS and DOS files. Flash memory is available in 2 or 4 megabytes, currently; but it is expected that on a future release the flash is expandable up to eight megabytes.

"   **DRAM** is available in 2, 4, or 8 megabytes, currently. A RAM disk can be created out of DRAM for fast, temporary storage of data and program files.

After a default software load, the following conventional memory is available (at the DOS prompt):
    upper memory: 75 kilobytes
    lower memory: 574 kilobytes

If your 6400 Computer has additional applications loaded in memory, the amount of available memory will be lower.

After a default software load, the available flash memory is dependent upon the size of your flash card and the amount of space used for files. If your 6400 Computer has additional applications loaded, the amount of available flash will be lower.

Input, Output, and Storage Devices More Varied

Dedicated PC Card
One PC Card type III slot is used for local area radio integration only. Currently, no other types of PC Cards work in this PC Card slot. This slot is also not accessible to the user.

Surface and Pin Connections
These connections support battery charge, printers, scanners, single dock, multidock, and mobile dock, through a communication adapter.

Infrared Printing
IrDA printing is also available.

Serial Data Communications
Communications takes place through standard ports.

Radio Modules
The PC Card slot is dedicated to the local area radio module. There are currently three radio modules available:
    " 2.4 GHz, at 100 mW
    " 902 MHz, at 250 mW

The 2.4 GHz module operates at short distances up to 500 feet (152 meters). The 902 MHz module operates at longer distances up to 1200 feet (366 meters).

Keep the System Environment in Mind

The PEN*KEY 6400 system environment should be kept in mind, while writing applications for this hand-held computer. It has a smaller CGA viewing screen; however, you have the capability of moving the view screen around as if the screen is a full CGA screen.

Hardware Ports

The communication ports and devices supported for those ports are described later, in the *Reference, System Information* section.

Tool Kit for the 6400 Computer

The file complement of the PEN*KEY 6400 Tool Kit differs from one release to the next. For an accurate list of Tool Kit files, consult the RELNOTES.TXT file included in the Tool Kit distribution package.

The Tool Kit is available in a DOS version.

The files and documentation in the Tool Kit are available to purchasers, of a 6400 Computer, who hold the following licenses:

  " NORAND® Operating System (NPN 999–001–014)
  " MS-DOS (NPN 999–001–067)

The DOS Tool Kit requires a DOS license only.

The 6400 Computer, as shipped from the factory, has preloaded flash memory with the files required for booting and telecommunicating.

DOS 5.0 is preloaded on flash drive C. In order to create the environment required by your application software, you select any other system files needed from the Tool Kit and copy them to the 6400 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 the application software is typically loaded onto the RAM drive or the Flash drive.

File Integrity Verification Utility: CRC32.EXE

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

The CRC filename has the pattern 64xx####, where xx would be a two character designation for Tool Kit (TK), Flash (BD), Boot Block Flash (FD), or Utilities (UT); and where #### represents the version number.

Examples:
    for the flash version 1.05, the CRC filename is 64BD0105.CRC
    for the Boot Block Flash, the CRC filename is 64FD0105.CRC
    for the Tool Kit, the CRC filename is 64TK1000.CRC
    for the utilities, the CRC filename is 64UT1000.CRC

Usage is as follows:
    `CRC32 [@][filename | pathname] [/s]`

where:

filename is the name of the file on which the CRC is calculated.

pathname is the location of the filename to be processed.

/s indicates that all subdirectories should be searched for matching filenames.

@ is (optionally) included to indicate that **filename** is an argument file, containing names of files to be checked. A valid argument file for CRC32.EXE has the same format as an argument file for the IPLFMT.EXE program.

" **NOTE:**   *Wildcard processing is not allowed. This means neither the filename, nor any of the pathnames within the argument file, are allowed any wildcards. CRC32.EXE processes directories with any attribute set, such as read-only.*

" **NOTE:**   *CRC32.EXE processes directories with any attribute set, such as read-only, etc.*

One or more files or directories can be processed at one time.

Tips for Getting Started

As you prepare to develop applications for the 6400 Computer (or port existing applications), keep in mind the following basic considerations:

" ROM DOS 5 is the operating system. If you are experienced with a DOS environment, you should easily be able to start developing or porting applications.

" Understanding how the system configuration files are used in the 6400 environment provides easier application development. This includes the DOS configuration files: CONFIG.SYS and AUTOEXEC.BAT.

" Become familiar with the tools and techniques for power management. Monitoring the state of battery power can be an especially critical function.

" Make your applications *drive-independent*. Do not hard-code drive designations. Utilize the available storage solutions, remembering that devices can be moved about as required by different configurations.

" Keep in mind the general system design of the 6400 Computer; with its key oriented input, flash memory file system, built-in CGA screen, etc.

" Finally, for development purposes, you may consider using external devices that are available to be use with the 6400 computer. But keep in mind, however, that these items may not be available (or practical) for use in the field.

System Configuration

The following paragraphs describe a typical minimum configuration for the 6400 Computer; but keep in mind that it can be configured to meet your specific needs. Additional flash RAM can also be obtained, by a service center upgrade, to bring the total flash memory up to maximum capacity.

Minimum Development Configuration

The following lists the items needed for a minimum development setup:

- Desktop or laptop host PC.
- A PEN*KEY 6400 Computer, with the minimum system RAM (as specified in the *Introduction to the 6400 Computer* paragraph) and 2 megabytes of Flash memory.
- Main battery: at least one.
- A NULL modem cable for connecting the PEN*KEY 6400 Dock to a host PC.
- If printing is needed, cable for connecting the PEN*KEY 6400 Dock to a serial printer or to a printer via the communication adapter. No cable is necessary for printers enabled with IrDA.
- External battery charger, single dock or communication adapter.
- The DOS Tool Kit for the 6400 Computer.

The preceding list assumes that you typically download through the RS-232 interface; you can use the device driver INTERLNK.EXE and host application INTERSVR.EXE, which are included with DOS. However, several other transfer techniques are available.

The drive designations for the 6400 Computer, as shipped from the factory, are as follows:

Drive	Type
A:	Local area radio card in the dedicated PC Card slot
C:	Flash drive
D:	RAM drive

The comm port assignments for the 6400 Computer are described in the *System Information* subsection, in the *Reference* section of this publication.

Sample Configuration Files

The configuration files, CONFIG.SYS and AUTOEXEC.BAT, are required for an environment for the 6400 Computer. For examples, refer to *Appendix A, Sample Configuration Files.*

Reprogramming Flash Memory

You can order flash upgrades on diskette, from Intermec, or download them from the Intermec Bulletin Board. Refer to BBS notes in the *Support Available* paragraph, in the introductory section of this publication, for details concerning the Bulletin Board. Contact an Intermec sales representative to obtain the media.

Read all of the instructions below before proceeding.

" NOTE: *Always keep the 6400 Computer on charge while performing any setup, reprogramming, or reflashing.*

INTERLNK Session

Prerequisites for INTERLNK Flash Update Process

The following items are required:

- " If you would like to use the procedure described here, you need a working 6400 Computer (if your flash is corrupted, you must perform a serial master mode boot).
- " A RAM drive (D:) of at least 960 KB is required.
- " A disk file contains the new flash archive, **64BDXXXX. EXE.** The last four numbers indicate the flash version (such as *0129* would indicate flash version 1.29).
- " A standard host PC for connecting to the 6400 Computer. **INTERLNK. EXE** must be loaded by the **CONFIG. SYS** file. For more detailed information on running INTERLNK than is presented here, refer to your DOS manual.
- " A NULL modem cable for connecting the host PC's communications port to the 6400 Computer with a single dock or communication adapter.

INTERLNK Installation

Use the following instructions only if you currently have a version of flash that includes INTERSVR as a possible Comm option on your 6400 Computer. If you do not have INTERSVR, you must update the flash using a serial master mode boot.

The following instructions assume that you have placed all files from the self-extracting archive in a directory (on your host PC) called C:\PENKEY\FLASH. If you choose to place these files in a different location, adjust the instructions accordingly.

" NOTE: *Make sure the self-extracting file is deleted from this directory once the files have been extracted.*

Your host PC must be running INTERLNK, which is part of MS-DOS. Load INTERLNK as a device driver in your CONFIG.SYS file, using the following statement, at the end of the CONFIG.SYS file (after any other statement that creates a drive letter):

    DEVICE=C: \DOS\INTERLNK. EXE /DRIVES: 3

The previous statement assumes that MS-DOS is located in the C:\DOS directory on your host PC. The /DRIVES: 3 parameter allows mapping of three drives from the 6400 Computer.

Reflashing Procedure

Begin with the 6400 Computer at the C:\ prompt. In order to get to the C:\ prompt, you may need to reboot. To do this perform a reset (as described in the paragraph on the *Four-Key Reset* operation, on page 1-12). Quickly after the reset, press the Alt key to bring up the Boot Selection Menu.

1. If you have not already done so, run 6400.BAT to create a RAM drive. If you already have a RAM drive, save the files somewhere and clear the RAM drive of all files. Make sure D:\ is in the DOS path and run INTERSVR with the following statement:

    `INTERSVR A: C: D:`

2. Establish the INTERSVR connection to your host PC, using a PEN*KEY 6400 single dock and cable, or a PEN*KEY 6400 communication adapter with a serial port cable.

3. Copy all of the files from the host PC's C:\PENKEY\FLASH directory to the RAM drive (D:) on the 6400 Computer. You should use the verify option (/v) on the copy to ensure that the files are transferred correctly.

    In order to determine what drive on your host PC corresponds to the 6400 RAM drive, type "INTERLNK" at the DOS prompt (without the quotes). You should see a chart similar to the following:

    ```
    This Computer      Other Computer
      (Client)            (Server)
    ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ      ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
     F:        equals  A:
     G:        equals  C: (1608Kb)
     H:        equals  D:
    ```

    In the example above, you would copy the files to drive H:. Make sure to copy the files to the RAM drive (D:) and NOT to the C: drive; otherwise it writes over the AUTOEXEC.BAT and CONFIG.SYS files.

4. Once the files have been transferred, exit INTERSVR on the 6400 Computer, by pressing Alt + F4.

5. Run XXX.BAT (where XXX is the version number).

    When the unit resets, it begins the flash update process. You are not prompted further unless an error occurs or a file is missing.

6. After reflashing, the 6400 Computer resets itself and boots back to the C drive. To confirm the flash version, observe the screen while booting from drive C:. If necessary, refer to the instructions above for overriding the default boot drive.

Your previous AUTOEXEC.BAT and CONFIG.SYS are renamed to AUTOEXEC.BAK and CONFIG.BAK. COMMAND.COM, RB.BAT, and RESET.EXE should be present on the RAM drive. These files (and the entire RAM drive) may then be removed, using the following command from DOS:

`ramdfmt –sØ`

Then reboot the 6400 Computer to free up that memory used by the RAM drive.

" **NOTE:**   *Refer to the release notes in the Tool Kit for changes from the previous version of flash.*

Development Environments

The 6400 Computer is compatible with the development environments listed in the following paragraphs:

DOS

The 6400 Computer can run any DOS program.

Microsoft Visual C, C++, or Borland C, C++, for DOS

These are tools for experienced C and C++ programmers. The foundation class libraries can be used; but make sure to keep an eye out for space limitations.

Other Environments

There are other usable development environments. Here are some guidelines that can help to determine whether a particular environment is viable for a PEN*KEY 6400 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.

2. The 6400 Computer uses a processor that is common to other PCs. But, be sure to test the performance of your application on a comparable computer. A typical development machine (50- to 100-Mhz 486) could hide some potential speed issues that would be evident on a computer, with a processor comparable to the 6400 Computer.

Setup for PC Development

" **NOTE:**   *Always keep the 6400 Computer on charge while performing any setup.*

The following steps outline a general approach for equipment setup for development of the 6400 Computer with a PC:

1. Connect power to the single dock or communication adapter.
2. Connect a NULL modem cable between the dock (or communication adapter) and a serial port on the desktop or portable PC.
3. Install the main battery.
4. Insert the 6400 Computer into the dock (or communication adapter).
5. As the 6400 Computer starts the boot process, you hear two beeps followed by 30 seconds of quick beeps, if the AC adapter is connected and the PC terminal program is running.
6. After the initial double beep, hold the [ALT] key down until the following ROM DOS 5 "Start from" menu comes up.

```
ROM DOS 5

Start from:

1) Memory card 1 = A:

3) Flash drive   = C:

4) RAMDrive      = D:

5) Boot to flash and bypass startup
```

7. Press the "3" key to choose option 3.
8. Run INTERSVR C: D:
9. Using one of the communication options, described later in this section, transfer the application program files to the 6400 Computer.

Four-Key Reset

When all four keys of the four-key reset (shown in the following figures) are pressed at the same time, for a period of two seconds, the system resets.

Refer to the *Communications and Device Support* section for figures identifying other keys for each of the two keyboards.

Figure 1-1 depicts the 41-key keyboard. The four numbered lines point to the locations of the four keys that reset the 6400 Computer.
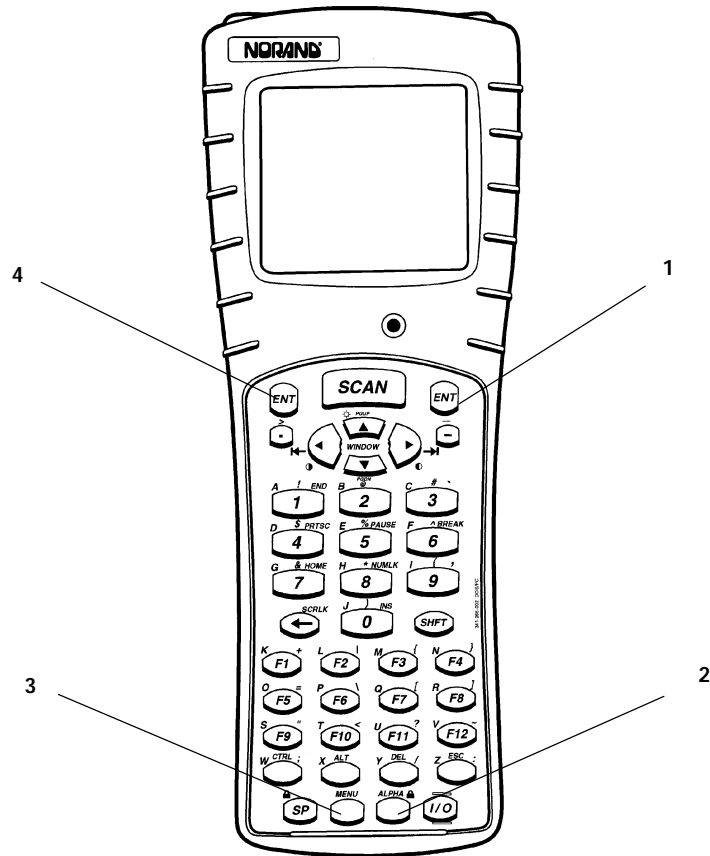
1. **Right enter key**
2. **Yellow (or gold) shift plane key**
3. **Blue shift plane key**
4. **Left enter key**

Figure 1-1
Reset Keys on 41-Key Keyboard

Figure 1-2 depicts the 51-key keyboard. The four numbered lines point to the locations of the four keys that reset the 6400 Computer.
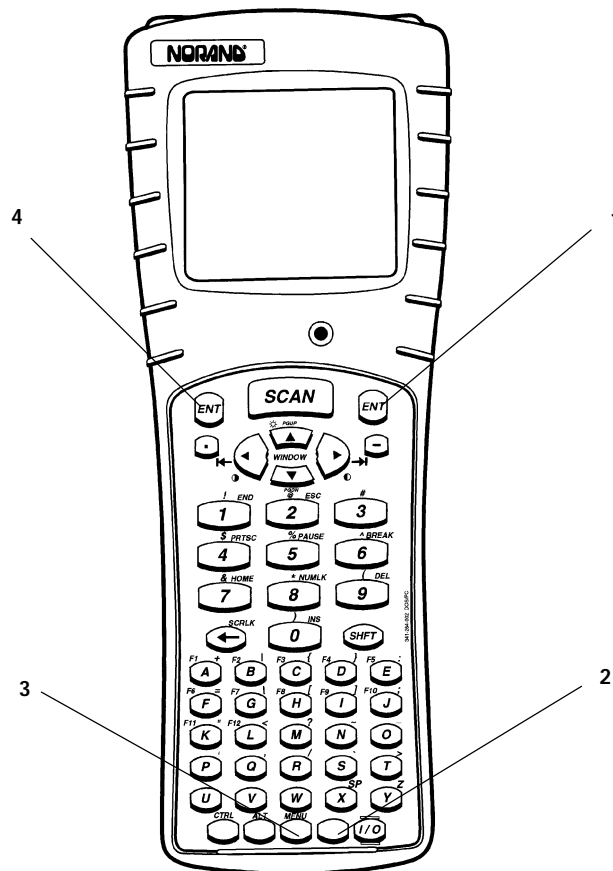
1. **Right enter key**
2. **Yellow (or gold) shift plane key**
3. **Blue shift plane key**
4. **Left enter key**

Figure 1-2
Reset Keys on 51-Key Keyboard

Development Resources

This paragraph covers the following topics:

Topic	Page
Open Systems Resources	
INTERLNK and INTERSVR	1-15
Norand Resources	
RAM Drive	1-15
NORAND Utilities and Communications	1-17
NPCP Network	1-18
TTY Network	
Other NORAND Software	
DOS Device Drivers	1-18
Utility Programs	1-19

6400.BAT	CDSVC1A.EXE	HIMEM.SYS	MMBFLAG.COM
64APMDEM.EXE	CLOCK.EXE	ISRAMDRV.COM	MODE.COM
BIOS.SYS	DELAY.EXE	LSL.COM,	ODIPKT.COM
BOOTDRV.COM	ETHDRV.EXE	MISCTSR.EXE	REBOOT.COM
RL2PCM.COM			

INTERLNK and INTERSVR

INTERLNK is a device driver that interconnects a 6400 Computer and a host PC through 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 PEN*KEY 6400 Tool Kit. A standard null modem cable connects the PC to the 6400 Computer. A TTY TCOM cable also works. A dock is needed for the 6400 Computer or a communication adapter that plugs onto the bottom end of the computer.

INTERLNK causes the 6400 Computer drives to appear as virtual drives on the PC, with drive letters 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 details of INTERLNK and INTERSVR topics, refer to the DOS on-line help text.

INTERLNK is installed on a PC, using the following statement in the CONFIG.SYS file:

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

After installation, you can copy the application files to the 6400 Computer. To terminate INTERSVR, press [ALT] followed by F4.

RAM Drive

The RAM drive on a 6400 Computer is a block of system memory treated as a DOS drive. The data stored on the RAM drive is maintained as long as power is supplied to the system and the drive is not reformatted.

Since the RAM drive consists of memory used by the application programs, this results in less extended memory available for programs. No additional conventional memory is required for the RAM drive. Because the 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-up messages) reserves memory for the RAM drive. Only one RAM drive can be created.

The size selected for a RAM drive represents the total memory it requires, not the free space available on the drive. DOS file system overhead reduces the specified memory by the size of the FAT and root directory created.

If a communication option is selected, the NORAND Utilities program automatically creates a RAM drive (if one does not already exist). The size depends on the amount of total memory in the 6400 Computer, as follows:

Table 1-1
RAM Drive Sizes

Total Memory (megabytes)	Default RAM Drive (megabytes) *
8	4
4	3
2	1

* The default RAM drive size is actually 64 KB less than specified, to allow for the High Memory Area (HMA).

For a system with 8 megabytes of total memory, only 4 megabytes is used as a RAM drive. For lower system RAM capacities, only 1 megabyte 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-megabyte increments. The 6400 Computer has hardware protection for RAM drives only on 1 megabyte boundaries.

When either the NPCP or TTY options are used, another choice exists for creating a RAM drive. If a file named RAMDFMT.CTL is downloaded, the NORAND Utilities executes the program RAMDFMT.EXE to create a RAM drive. RAMDFMT.CTL must be a text file that contains the command-line parameters for RAMDFMT.EXE. For example, if RAMDFMT.CTL contains the following line:

    −s2048

a 2 megabyte RAM drive is created.

You can also create a RAM drive from the DOS prompt, with the following statement:

    RAMDFMT −snnnn

where nnnn represents the desired size of the RAM drive.

Complete list of command line switches (all values in decimal):
    −a = [address of RAM disk in kilobytes]
    −d = [drive number]
    −e = [number of directory entries]
    −s = size of RAM disk in kilobytes]
    −h = [print this screen]

" **NOTE:**   *When formatting the RAM drive, specify either the size or starting address, but not both.*

NORAND Utilities and Communications

Communications Using NORAND Utilities

These utilities support data telecommunications between a PEN*KEY 6000 Series Computer and a host computer. Refer to the *NORAND Utilities* paragraph, in the *Communications and Device Support* section, for a description of the NORAND Utilities.

Creating a Host Download Include File

Most of the communication options for performing an initial program load require a download include file to be used on the host computer. To create a download include file, proceed as follows:

1. Create a download list file.

    A download list file contains the names of the files to be sent to the 6400 Computer. Each line in the file specifies one name. The file names may include path information so that all the files need not be in the same directory on the host computer. However, the file is transmitted to the 6400 Computer without the source path information, and placed in the current directory of the 6400 Computer. To specify a destination path or to rename a file, on the 6400 Computer, use a line with the following format:

    "**newname=oldname**"

    This causes the file "oldname" to be read from the PC, then transmitted to the 6400 Computer with the name "newname". Both "oldname" and "newname" may include path information.

    A sample download list file:

    CONFIG.SYS
    AUTOEXEC.BAT
    6X00DOS\PC4800.SYS
    MYAPP.EXE

2. Create the download include file by executing the following command:

    IPLFMT.EXE <list file> <include file>

    where:
    <list file> is the name of the download list file created in step 1.
    <include file> is 6400IPL.INC. This file is created by IPLFMT.EXE.

" **NOTE:**   *IPLFMT.EXE is included in the Tool Kit.*

Options:

–r Allows creation of subdirectories when needed.
    An output directory can be specified.

@   is (optionally) included to indicate that filename is an argument file, containing names of files to be checked. A valid argument file for CRC32.EXE has the same format as an argument file for the IPLFMT.EXE program.

A valid argument file for IPLFMT.EXE has the same format as an argument file for the CRC32.EXE program.

NPCP Network

NPCP is the NORAND Portable Communications Protocol. This protocol is supported by the NORAND 492x and 498x communication controllers.

1. Create a download include file, as described in the preceding paragraphs.

2. Create a 4920 boot disk, if necessary. 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.

" NOTE: *4000 Series applications are copied to a BOOT subdirectory. PEN*KEY 6000 Series applications are copied to the DOWNLOAD subdirectory.*

3. Create a 4980 Boot Disk, if necessary. Place the 4980 System Files on the 4980 Boot Disk in the same manner as the 4000 Series Applications (using the 4980 Boot Tool Kit, for example). Execute the following commands to place the download include file on the disk:

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

" NOTE: *NCDIR.EXE is included in the 4980 Boot Tool Kit.*

TTY

TTY is an industry standard protocol with a NORAND implementation, supported by the 4920 Communication Controller.

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

DOS Device Drivers

The following are DOS drivers, for power management, scanning, etc.

Table 1-2
DOS Device Drivers

Drivers	Description
64APMOEM.EXE	DOS Norand OEM APM extensions driver
641223.EXE	DOS Scanner driver
BIOS.EXE	Version of BIOS
CLOCK.EXE	DOS Clock bug patch
DOSGAS.EXE	Battery indicator
ELANAPM.EXE	APM BIOS installation for DOS
ELANCFG.EXE	ELAN Configuration Utility
ELANUMP.SYS	
IRDAPDRV.EXE	DOS IrDA Printer Driver
MISCTSR.EXE	
NORDOSPM.EXE	DOS APM driver
PC4800.SYS	Standard DOS 4800 Series Printer driver
PRDRV.SYS	Standard DOS Printer driver
RAMDFMT.EXE	To create RAM drive

Utility Programs

6400.BAT

This utility is used to prepare for a software update. It creates a RAM drive (960 Kb in size), sets the path=C:\, runs intersvr A: C: D:, then switches to drive D:.

64APMOEM.EXE

Provides APM interrupt extensions to the APM 1.2 specification, that are unique to Intermec.

These extensions include:
- " Accessing the non-volatile parameter storage area (e.g., location of unit serial number and other stored data).
- " Getting or setting the keyboard mode, type, and version.

BIOS.EXE

Displays the version level of the BIOS currently in the 6400 computer.

BOOTDRV.COM

This application is specifically designed for use in batch files that are required to know which drive is 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 3 SET BootDrive=C:
IF ERRORLEVEL 4 SET BootDrive=D:
```

CDSVC1A.EXE

This utility provides the PC card and socket services for Proxim's 2.4 GHz radio driver: RL2PCM.COM. It is intended for use only with RL2PCM.COM and NOT a general purpose PCMCIA card and socket driver.

Command line switches:
   None.

CLOCK.EXE

This patch driver corrects the DOS clock after a suspend/resume sequence, where the actual time may be out of sync with the DOS time.

Command line switches:
   None.

DELAY.EXE

Usage:

DELAY /n   where n is the number of seconds to delay.

This file serves four functions:

  " Displays a message to the screen (like the Echo command).

  " Waits for some amount of time to expire before continuing (like a Sleep command or DOS 6 Choice command with a timeout specified).

  " Pauses execution of the system until a key is pressed (like the Pause command).

  " Returns an error level based on which key, if any, was pressed (like the DOS 6 Choice command).

The first three of these 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. To perform each of these functions:

  " 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 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 timeout or a key press, specify a delay timeout of zero (/0).

  " 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 representing the hundredths of seconds that must elapse before DELAY automatically exits. The maximum delay is about 640 seconds (or about 10 minutes).

  " 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.

DELAY.EXE Error Levels

When DELAY is executed from a command shell (as opposed to being loaded as a device driver) the error level that is set, upon return to the calling function, is based on the key (if any) that was pressed prior to exiting. The return value is specifically geared to make it easy to determine which numeric key was pressed.

Table 1-3
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 254	The values returned for these keys consist of the key value minus 30h.
255	No key was pressed, the time expired first

Simple Examples:

```
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 . . . "
```

ETHDRV.EXE

Command line switches:
   None

HIMEM.SYS

Add /machine:2 for A20 gating to be performed by port 92h.
Refer to a DOS manual, or DOS help, for additional information.

ISRAMDRV.COM

This simple program returns an error level that answers the question, "Do I have a RAM drive?" If the error level is set to zero, then a RAM drive exists. This command is specifically designed for use in batch files that have a requirement to know whether a RAM drive has already been formatted.

Batch file example:

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

LSL.COM

Usage:
```
LSL [/?] [/U] [/C=[path\]filename]
```

Command line switches:

/?   Displays a help screen
/U   Removes the resident program from memory
/C=[path\]filename Specifies a configuration file to use
(default is NET.CFG)

MISCTSR.EXE

This utility provides the following features, for use on your 6400 computer.

Command line switches:

```
–A   Enables the reset annunciator line at program termination
–Fn  Select annunciator character set
        n=0 default annunciator character set
        n=1 alternate annunciator character set
–K   Enables get character CPU idle
–L   Enables legacy INT 15h function AX=42FFh
–Pn  Enable/Disable annunciator panning
        n=0 disables annunciator panning
        n=1 enables annunciator panning
–Sn  Enable/Disable scan handle trigger
        n=0 disables scan handle trigger
        n=1 enables scan handle trigger
–T   Enables periodic DOS timer reset RTC
–Wn  Select window key set
        n=0 default window key set
        n=1 alternate window key set
```

For additional information relating to annunciator character set selection, refer to the *Fonts Rows and Columns* paragraph, in the *Communications and Device Support* section of this publication.

" **NOTE:**   *Certain programs (such as FTP's TN.EXE) use the keyboard ports 60h and 64h directly. The W option uses a different set of scan codes for the windowing arrow keys, so this windowing function works properly.*

" **NOTE:**   *The annunciators consume various characters in the font table. By using the F option, a different set of characters from the font table are consumed. If a character you need is consumed by the default set, try the alternate set.*

MMBFLAG.COM

The MMBFLAG.COM program satisfies a couple of batch file needs:

" You can request the boot flag value, by running MMBFLAG.COM, without a parameter. The returned value indicates the current boot drive. This flag keeps track of the default boot drive and serves as an indicator of Master Mode Booting.
The following example shows how you could request the boot flag value:

   `MMBFLAG.COM`

This command merely places the value into the DOS Errorlevel flag.

To retrieve the boot flag value, examine the DOS Errorlevel flag.

Unfortunately, the Errorlevel value contains both the reserved value and the value of the boot flag. To determine the default drive, subtract 128 from the Errorlevel flag value and compare it with the Boot Flag values in the table below.

" You can also change the default boot drive by supplying a simple command parameter, as follows:

   `MMBFLAG.COM n`

where n is the desired boot flag value (range = 0–7).

Use the following table to determine what value of n to supply.

Table 1-4
MMBFLAG.COM Return Values

Boot Flag Value	Meaning
128	Reserved
7	Drive H is the default boot drive
6	Drive G is the default boot drive
5	Drive F is the default boot drive
4	Drive E is the default boot drive
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

MODE.COM

Refer to a DOS manual, or DOS help, for details.

ODIPKT.COM

This is a shim that translates packet driver type calls from a protocol stack to ODI type calls to LSL and an ODI driver. In other words, it allows you to use a packet driver type protocol stack on top of an ODI driver. There are many 'odipkt' type shims available from many different vendors.

"NOTE: *ODIPKT.COM is FTP software PC/TCP packet driver for ODI v2.3, CopyrightE 1986—1993 by FTP Software, Inc. All rights reserved.*

Usage:
```
ODIPKT [?] [unl] [SINT=xx] [MLID=name] [FRAME=frame-string]
```

Command line switches:

?	Displays a help screen
unl	Causes ODIPKT to be unloaded
xx	The software interrupt number (range: 60–80 hex, default: 60)
name	The MLID type: (*values are case sensitive*)
	ETHERNET_II or **ETHERNET_SNAP** or **TOKEN-RING_SNAP**
	Default: **ETHERNET_II** or **TOKEN–RING_SNAP** (first found)
frame-string	The media type: (*values are case sensitive)*
	(**ETHERNET_II** or **ETHERNET_SNAP** or **TOKEN-RING_SNAP**)

REBOOT.COM, Warm Booting Utility

This utility is included in the Tool Kit, and performs a warm reset when run.

RL2PCM.COM

Executing without parameters, installs the program.

Usage:
```
RL2PCM [?] [U]
```

Command line switches:

?	Displays a help screen, showing these command line switches
U	Removes the resident program from memory

Section 2

Supporting DOS Applications

.. ..

Introduction

This section contains information about DOS Applications that are supported for the PEN*KEY® 6400 Computer.

Topic Summary

The following list of tables may assist you in locating topics of interest:

List of Tables

DOS Power Management Driver: NORDOSPM.EXE

NORDOSPM.EXE, which is part of the Advanced Power Management (APM) system when running under DOS, is the DOS NORDOSPM 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 the information 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 NORDOSPM management settings at installation time.

Installation

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

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

The driver, **ELANAPM.EXE,** needs to be loaded first.

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 /IOxn /Kn /Ln /Mn /Rn /Tn /Vn /Wx

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 is 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 Timeout 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 count down 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. Since the Low Speed PLL mode count down timer starts when the processor enters Low speed PLL mode, the processor will never enter 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.

Note: see the *Anomalies Relating to Timeouts* paragraph below for a description of interactions between timeout, bootup, and suspend/resume.

/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 Timeout 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.

Note: see the *Anomalies Relating to Timeouts* paragraph below for a description of interactions between timeout, bootup, and suspend/resume.

/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.

Anomalies Related to Timeouts

When you set timeouts using /Hn and /Ln, you need to be aware of the battery status that may be returned as a result of that setting, as described below.

When FFh is returned from ELANAPM.EXE, this indicates "unknown battery status". This may be a result of long delays for battery status packets being returned. ELANAPM.EXE returns FFh upon bootup or after a resume until it gets the battery packets it needs to calculate the true battery status.

When running in High Speed mode or Low Speed mode this will take 10-90 seconds for the battery packets to be received. However, when in Doze mode the 6400 Computer is unable to receive any battery packets. Thus, it will be a very long time (if ever) for ELANAPM.EXE to give back a non-FFh value.

Some possible ways to work around the problem:

" Set the total time in High & Low speed modes to at least 11 seconds. This assures that at least one valid battery packet will be received upon boot, resume, or after any activity (e.g., a scan, key press, etc.).

" After boot or resume setup the unit to be active for at least 11 seconds, then you can have 1 second in High Speed & 1 second in Low Speed mode. The application can easily determine whether it just booted up or resumed. Then it could cause some activity for the 11 seconds and stop the activity, allowing the ELANCFG setting to take over and maximize the time spent in DOZE mode. This allows at least one valid battery packet to be received.

" Give up on on the APM call while in DOZE mode and merely allow the unit to autosuspend.

DOS Scanning

64SCN7B.EXE / 641223.EXE

" **NOTE:**   *The 64SCN7B.EXE scanner driver is being replaced by the 641223.EXE scanner driver. The information below is valid for both drivers, except as noted.*

This is a DOS TSR driver for the internal laser engine and external serial scanners. The scanner driver must be loaded for any scanning to occur.

Make sure ELANAPM.EXE and NORDOSPM.EXE are installed in the CONFIG.SYS file, because the scanner driver uses features of the APM system. The APM installation is described in the *DOS Power Management Driver* paragraph, on page 2-2.

" **NOTE:**   *Always load the scanner driver before any radio drivers.*

Command Line Options

The scanner driver is generally installed in the AUTOEXEC.BAT file, according to the following format:

        `641223 [–option[value]] ...`

It uses command line switches to enable the desired functions.

For example:

        `641223 –E –A2 –W`

The descriptions of the command line switches, for the scanner driver, are as follows:

Table 2-1
641223 Options

Switches	Description
–? or –H	Both options display help screen, briefly describing various options.
–A[n]	Enable the engine aiming beam in "n" half second intervals for long-range scan engines with aiming beams. The "n" is optional, and specifies the number of intervals desired. The default is n=0=OFF.

–B[n]	Beep on good scan. Range of valid values:
	        0 = no beep
	        1 = beep (default)

–C	Append CR/LF at the end of each barcode.
–E	Bypass Application Program Interface (API) enablement. This switch turns on the scanner immediately and permanently. Without this switch, the scanner must be turned on using the API scanning method. Use this switch when operating in wedge mode.

–I	Internal scan enable. Range of valid values:
	        0 = disable (do not load internal scanner)
	        1 = enable (default)

–P	Selects the hardware keyboard wedge method. Scanner data is sent to I/O port 60h, making the scanner data appear as hardware key presses. Some applications, such as FTP Telnet, bypass the PC BIOS and go to the hardware to get key presses. If the PC BIOS wedge method does not work, try this switch.
–R	Translate tethered scanner barcode type.

–T	Tethered scan enable. Range of valid values:
	        0 = disable (do not load tethered scanner)
	        1 = enable (default)

Table 2-1 (continued)
641223 Options

Switches	Description
–U	Unload TSR, if safe to do so.
–W	Enable the wedge mode of operation. Scanner data is sent to the keyboard buffer, using the method selected by the –P switch.
	The absence of –W requires an application to request scanner data through the API scanning method.

Integrated Long Range Laser Engines

For integrated long range laser engines, the aiming beam needs to be setup. The switch to set the long range aiming beam for the scanner is:

641223 –An

where "n" is the number of 1/2 second intervals.

6400 Laser Engine Decode Capability

There are some differences in the capabilities of the internal laser engine used in the 6400 and the non-decoded laser technology used in the RF terminals. The 1122 and the 1223 scanners are supported as described in the following table.

Table 2-2
6400 Laser Engine Decode Capability

Symbology	Supported for 1122	Supported for 1223
ABC Codabar	No	No
Code 93	No	Yes
Code 11	No	No
Computer Identics 2 of 5	No	No
Straight 2 of 5	No	Yes
UPC–E	Number system 1 shelf labels are not supported	Yes
MSI Plessey	Yes, except alpha characters are not supported	Yes
Interleaved 2 of 5	Decodes 1 of 2 fixed lengths, no minimum or maximum range.	Yes

Scanning Methods

The 6400 Computer supports scanning, with an external tethered serial laser scanner connected to the serial communication port or the optional internal laser engine. Scanner data can be sent to an application via the keyboard (Wedge Mode) or through an API, using INT 7A. Refer to the *BIOS Interrupt Support* section for usage information.

The following paragraphs describe the scanning operation for:
 " Tethered Scanning, using an external serial scanner.
 " Internal Scanning, using the internal laser engine.
 " Wedge Scanning Method, where data is presented to a scanner-unaware application as key strokes.
 " API Scanning Method, where a scanner-aware application enables and disables scanning, symbologies, and other bar code parameters.

Tethered Scanning

Tethered scanning is supported for any external device that sends its serial data at 9600 baud, 8 data bits, no parity, 1 stop bit, with an ASCII STX (hex 02) before the data and an ASCII ETX (hex 03) after the data. No protocols are supported such as ACK/NAK, RTS/CTS or XON/XOFF. All 8-bit characters between the STX and ETX are stuffed into the PC BIOS key buffer. The scanner driver limit for the number of characters is in excess of 100 characters.

When using tethered scanning without an internal laser engine, add the "–i0" switch to the scanner driver command line in order to disable the engine. If the engine is not disabled, then initialization of the scanner driver engine fails and causes the message "`DASI Scanner HW initialization failed`". This does not cause any problems.

Tethered laser power is always on, allowing a tethered laser to scan, decode, and send the data as soon as it is plugged in; even when there is no driver and application to handle the data. Therefore, proper operation of the tethered laser is no indication that you have the correct version of the scanner driver, that it is correctly configured, nor that the application program is designed correctly.

There is no supported method to send configuration parameters to a tethered serial device. Tethered laser devices can only be configured using setup bar codes that are listed in the respective user manual.

Internal Scanning

Internal scanning uses the internal laser engine that connects to the top end of the 6400 Computer. The scanner driver controls the power to the engine, which means the scanner driver must be loaded before the 6400 SCAN key can operate.

When using the internal laser engine without a tethered laser, add the "–t0" switch to the scanner driver command line in order to disable handling of the tethered laser serial data. This is especially necessary if the serial port is used for another device, such as a printer.

You can change the default engine configuration by scanning setup bar codes. Refer to the *SE 1223 Integrated VLD Scanner User's Guide*, for the setup bar codes for the scanning engine.

The engine default values may be restored by scanning the RESET bar code. These values are as follows:

Table 2-3
Engine Default Values

Value	Description
1	No preamble characters
2	0x0d,0x01 (<cr–lf>) postamble. May need to be disabled, depending on the application program.
3	UPC/EAN, Code 39, I 2 of 5, Code 128, Codabar, MSI Plessey are *all enabled.* Disable those not needed. Only enable the needed symbology.
4	Addon-2 and addon-5 supplementals for UPC and EAN are *disabled.*
5	No symbology type prepended to the bar code data.
6	MSI Plessey application check digit is disabled.
7	Code 128 function characters are removed from the bar code data.
8	I 2 of 5 fixed length 1 is 14 characters. Set this to the actual length of the bar code used by the application. A length of zero disables that length option.
9	I 2 of 5 fixed length 2 is zero which disables it. Leave the second length disabled unless the application uses two different length bar codes.

The internal scanner engine may be configured one of two ways:

1. Scan setup bar codes in the *SE 1223 Integrated VLD Scanner User's Guide*, when using the Wedge Scanning Method described below.
2. The application software can send commands to the scanner API when using the API Scanning Method described below.

Some configuration options require scanning a single bar code, after which the computer beeps to indicate that the bar code parameter has been changed. If the parameter is the same as it was, the computer will not beep. Those configuration options, which require more than one bar code to be scanned, will not beep until the LAST bar code in the sequence is scanned. The engine currently uses a power conservation method that causes it to power down after 1/2 to 1 second. Because of this, configuration bar codes in a multiple bar code sequence must be scanned within 1/2 second of each other.

" NOTE: *Tip: After reading a bar code, aim the engine away from the bar codes and press the SCAN key within half a second, to get the laser beam back on. With the laser beam on, you have five seconds to find and scan the next bar code.*

Wedge Scanning

Wedge scanning is a method, in which scanner data is presented to an application as key presses. Wedge scanning is simple to implement, but allows scanning at any time and does not support data validations beyond those provided by the laser device.

Some applications are sensitive to non-key stroke values in the key buffer such as 0x0a or 0x0d or other non displayable characters. So be aware of what the tethered device is sending. If data does not make it to the key buffer, is wrong, or the application hangs, then verify what data is being received from the scanner device. A DOS terminal emulator, such as ProComm, can be used (without the scanner driver loaded) to verify the data received at the serial com port.

Load the scanner driver TSR using command line switches " –e –w". Then load the application. Do **not** load or use 6400IKPS, 6400BIOS, 6418BIOS, nor any FWP64xHx programs with the the scanner driver TSR, when either the "**–e**" or "**–w**" switches are used. These two switches cause the scanner driver to send scan data from the laser to the PC BIOS console key buffer, where an application can read it as key strokes. 6400IKPS, 6400BIOS, 6418BIOS, and FWP64xHx do not look for their scan data in the keyboard buffer. They get the data through an API INT call to the scanner driver (see API SCANNING METHOD below).

Wedge scanning allows an operator to scan ahead of the application because the application cannot disable scanning. The scanner driver buffers scanner data, so that no scanner data is lost, and passes the data to the key buffer as the application asks for keys. The operator gets a good scan indication even though the application may later reject the data. When the application rejects the scanner data, the operator must figure out which scans were accepted and which were not and re-scan the rejected scans.

" **NOTE:**   *The current version of The scanner driver allows itself to be loaded multiple times. Be careful, do not re-load the scanner driver when a copy currently resides in memory. If it is necessary to re-load the scanner driver with different switches, you must reboot between each load. Future versions of the scanner driver will have the capability to detect whether it is already loaded, unload from memory, and send command line switches to the in-memory copy to change its configuration.*

API Scanning

6400IKPS.EXE is a DOS TSR that implements a scanning API. The API facilitates porting of RF terminal Application Development Kit (ADK) applications to the 6400 Computer and development of DOS client scanning applications for the 6400 Computer, by hiding the complexities of scanning and validation algorithms from the application developer. The API enables an application to:

  " Disable the internal scanner engine when scanning is not desired.

  " Validate bar code data beyond what the laser device provides.

  " Process scanner data differently than keyboard data.

  " Programmatically enable or disable symbologies and symbology options for the internal laser engine and certain symbology options for tethered lasers.

The API is called using a software INT 0x7A instruction with parameters passed in registers. The calling interface is patterned after PC BIOS calls, and is described as follows.

API Scanning Application Architecture

A scanning application that uses the API would first load 641223.EXE without any command-line switches, the scanner API 6400IKPS.EXE next, then itself.

Figure 2-1
API Scanning System Architecture

DOS Scanning Program SCANDOS	64SCNPRM.EXE 64SCNPRM.DAT
64SCN7A (single-threaded) API INT 7Ah	6400IKPS (multi-tasking) API INT 7Ah
641223.EXE	
Tethered Serial Device	Internal Laser Engine

The diagram above represents the architecture, depicting the layering of software and hardware. Beginning with the lowest layer, the hardware is included first; then the software is installed, as shown from the bottom toward the top of the diagram.

INT 7Ah API Functions

The scanning API INT 7Ah functions are used to control and obtain the data from the scanner. 6400IKPS and 64SCN7A provide the following INT 7Ah functions.

Table 2-4
Scanning API Functions

AH=	Function	Description
80h	Scanner On	Turns the scanner on and enables the SCAN_Key
81h	Scanner Off	Turns the scanner off.
82h	Set Scanner Type	Establishes the scanner type.
83h	Set Bar Code Options	Tells the scanning system which symbologies to enable and with which options.
84h	Read Bar Code Data	Gets bar code data, if available.
FEh	Flush Scanner Concatenation Buffer Data	Flushes the concatenation buffer automatically, under conditions specified in the *BIOS Interrupt Support* section.

Refer to the *BIOS Interrupt Support* section, for details of these functions, which are provided by Intermec Technologies Corporation.

64SCN7A

64SCN7A.EXE is a single-threaded scanner API. It should be used by single-threaded DOS applications to control and acquire scanner data. Because it does not use multi-tasking functions, it is much smaller than 6400IKPS.

6400IKPS

6400IKPS.EXE is a multi-tasking scanner API. It provides the scanning API and ADK INT 15 multi-tasking functions. 6400IKPS must be used with multi-tasking ADK applications and 64SCNPRM (which uses multi-tasking).

API Calling Sequence

The 6400IKPS API functions do not work if these rules and sequence are not followed:

1. The ***Set Bar Code Options*** function can be called anytime to set the bar code options.
2. The ***Set Scanner Type*** function should be called only while scanner is off.
3. Call the ***Scanner On*** function.
4. Do idle loop
    allow a ***Scanner Task Switch***   ; (see INT 15h, AX=010Ch)
    if a key is pressed   ; (see INT 16h, AH=01h)
      read the key   ; (see INT 16h, AH=00h)
      if the key is the *SCAN_Key*   ; (*SCAN_Key* is setup in the
                                          ;   INT 7Ah, AH=80h, function)
        call ***Read Bar Code Data*** function ; (see INT 7Ah, AH=84h)
        process the bar code data
      else
        process the key
      endif
    endif
    if there is something else to do
      do something else
    endif
  as long as needed
5. Call the ***Scanner Off*** function.
6. Terminate the program.

The application must call the ***Scanner Task Switch*** function in its idle loop. This is when 6400IKPS gets control to do scanning in the background. If the application does not call ***Scanner Task Switch***, then the internal scanner laser beam does not come on when the SCAN key is pressed, and the data from a tethered scanner is ignored. Any delay in the application calling ***Scanner Task Switch*** is reflected in scanner responsiveness.

Internal Laser Configuration Tool: *64SCNPRM.EXE*

64SCNPRM.EXE is provided in the 6400 DOS Tool Kit to facilitate setting the internal laser engine configuration parameters. 64SCNPRM uses multi-tasking functions and must be loaded over 6400IKPS.

The following information applies to the 1122 scanner. The 1223 scanner has additional options (refer to the table on page 2-7).

64SCNPRM uses a hierarchy of menus as the user interface. From the first menu displayed after loading, you can select:

"1" to set symbology Scan options
"3" to scan bar codes to test your setup
"6" to exit to DOS

Loading 64SCNPRM initializes the engine symbology options to those values found in a file named, 64SCNPRM.DAT. If the file is not found, it is created with default settings that turn everything off. These options are then sent to the laser engine. During the time it saves these parameters, a message to that effect is displayed. If the file cannot be created or written to, an error message is displayed, "**Press any key to proceed**". It is possible for the engine parameters to be different from the 64SCNPRM default, or saved values, if the engine scans "setup bar code labels" in any of the Symbol smart laser user guides.

64SCNPRM changes the following engine options in addition to its menu options:

Send No-decode Characters	disabled
Bi-directional Redundancy	disabled
Decode Attempt Time	7
Preamble	disabled
Postamble 1	disabled
Postamble 2	disabled
Send Code ID	disabled
UPC/EAN Security Level	none
UPC/EAN Security Zone	4
UPC/EAN Supplement Auto-D	enabled if addon-2 or 5 enabled in menus
Decode Addon-2, 5 required	disabled
UPC-E Check Digit	disabled
UPC-A Check Digit	disabled
UPC Prefix	system character only
Send Code 128 Function Characters	disabled
Send Scan Direction	disabled

SCANDOS

SCANDOS is a sample single-threaded DOS scanning application, which runs over 64SCN7A. The C source code is provided with comments explaining how to build and use SCANDOS.

Scan Options Menu

The Scan Options menu allows enabling and disabling symbologies, by pressing the number of their menu item. Enabled symbologies and options are indicated by displaying the menu item number in reverse video. Pressing the symbology number again disables it. If the symbology has options, such as fixed lengths 1 and 2 for interleaved 2 of 5, or addons for UPC and EAN, then an option menu is displayed. Pressing the ENTER key returns you back to the previous menu. Returning to the main menu saves the setup to the engine and 64SCNPRM.DAT.

The Scan Options menu lists only those symbology options of the internal laser engine that are in the bar code setup menus of the NORAND RF terminal emulation software. In a future release of 64SCNPRM, all of the user settable engine parameters, specified in the *SE 1223 Integrated VLD Scanner User's*

Guide, will be supported. The software symbology validations are not currently supported.

The "**1)   Laser**" menu item cannot be changed; Laser is always enabled, as there is no other scanner device type supported by the 6400 Computer.

Scan Barcodes Menu

The *Scan Barcodes* menu enables the scanner so you can test your setup. The symbology type is displayed to the right of the "Type>" prompt, in hexadecimal; and if it is a displayable character, it is displayed in ASCII. If the scan was a concatenation, then a "C" is displayed to the right of the type character.. The bar code data is displayed on the next line (or lines), followed by the length of the data to the right of the "Length>" prompt.

Internal Laser Engine Symbology Validations

Some of the symbology validations are handled by the internal laser engine. If those validations fail the engine keeps trying to decode until the trigger is released or the scan timeout is reached.

Those validations, not supported by the internal laser engine, are executed in software after the engine has decoded the bar code. When these validations fail, they cause a no-decode and the operator must release and press the SCAN key on the keyboard to try again.

Software Symbology Validations:
- All symbologies in the *Decoded_Symbology_Types* list:
     Length validation
     Drop leading, trailing characters
- Code 39:

     Full ASCII
     Mod check
     Concatenation
- Interleave 2 of 5:
     Mod check
- Codabar:
     None
- UPC and EAN:
     UPC-A, UPC-E, EAN-8, and EAN-13 type validation
     Addon-2 validation
     Addon-5 validation
     Expand UPC-E to UPC-A
     Expand EAN-8 to EAN-13
- Code 128:
     Strips unwanted function characters
     UPC/EAN-128 F1 detection, F1 formatting
     F2 concatenation
- MSI Plessey:
     None

Tethered Laser Symbology Validations

Tethered lasers must be configured to send a symbology type character. The scanner driver assumes the first character of the tethered laser bar code data is the symbology type character and removes that character from the bar code data.

The tethered laser symbology type characters causes their respective Software Symbology Validations to be executed if the symbology is enabled in the *Barcode_Options* array of the **Set Bar Code Options** function and the symbology type is in the *Decoded_Symbology_Type* list of the **Read Bar Code Data** functions. The Software Symbology Validations available for the tethered laser are the same as those for the internal laser engine.

If the symbology type is not in the *Decoded_Symbology_Type* list, then the symbology type character and the bar code data is returned to the application as-is without any validation checks.

DOS NPCP Printing: PC4800.SYS

NPCP Printing Support, under DOS, consists of DOS device driver PC4800.SYS, which allows DOS and PL/N applications written for 6400 Computers to print to NORAND**R** 4810, 4815, and 4820 Printers, using the NPCP protocol.

Installation and Configuration

Required CONFIG.SYS Entry

To install the DOS driver for the 4815 or 4810 Printers, place the following statement in the CONFIG.SYS file and reboot. If PC4800.SYS is not in the root directory, be sure to include the path.

    device=PC4800.SYS LPT1 1 /I1

NOR4800 is the default name for the device driver that is used to open the device. By using PC4800.SYS, a specific device driver is specified.

Add a command line parameter (LPT1), as shown in the preceding statement, to override this. The driver can then be accessed via this name when the device is opened. Give it any name you want (except PRN) (using up to eight characters).

Add a second parameter to indicate the communications port number to use for output: **1 = COM1**,   **2 = COM2**, etc.

To set the communications port number, you must also specify the device name. Parameter order is important. The default port is COM1. By default, PC4800.SYS supplies only the DOS device driver interface to NPCP Printers.

To support PL/N applications, interrupt 17h must also be installed. To do so, add a /I switch to the command line, as shown in the preceding statement. The digit placed after the "I" refers to the LPT port, which gets redirected to the device driver at the interrupt 17h prompt. For example, if a "1" is specified, then all interrupt 17h calls for LPT1 are redirected to the PC4800.SYS Device Driver.

Typical AUTOEXEC.BAT Entries

Refer to *Appendix A, Sample Configuration Files*, for a typical examples of items that should be in your AUTOEXEC.BAT file.

Functionality and Usage

Intermec 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.

The DOS driver for the 4815 and other NPCP Printers must be opened before it can be used. Use the correct open call, for the language being used, that allows writing to the device. An open mode that opens only devices (files) when they currently exist, checks for the existence of the device and returns an error if it is not there. Open modes, that create a file upon opening, even creates a file if the 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 you can use **fprintf()** .

You should install a critical error handler that processes errors from the printer. The printer driver only returns "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.

The **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 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 to which the call is being directed.

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 the NPCP Protocol. 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.

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.

The printer driver intercepts interrupt 17h (Printer BIOS Services) and translates the requested functions to provide PL/N support. The BIOS interface provided for INT 17h is not 100 percent PC compatible: it is meant to be compatible with PL/N. The support that the BIOS interface provides for standard applications is unknown.

DOS IrDA Printing: PRDRV.SYS, IRDAPDRV.EXE

The 6400 Computer supports IrDA printing under DOS. It 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.

Device Driver Entry Points

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 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 was found to be 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_COELESCING_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 has been 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=c:\prdrv.sys
install=c:\irdapdrv.exe –t6400

where *t6400* is the "technology" of 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 MUST be used, and the optional name selected MUST match this name string exactly. The default is IRDALPT.

B **CAUTION:**   **Use only the switches shown below for IRDAPDRV. It is a multi-platform program, providing features for all of the 6000 Series platforms. None of the other switches are intended for the 6400 Computer and could cause it to malfunction, if used on this platform.**

IRDAPDRV   [–? –bn –dn –fs –n –rn –t6400 –x]

where:

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

–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 quires.

–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.

–t6400   Identifies the 6400 Computer as the IrDA-equipped system, on which the driver is running.

–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.

DOS IrDA Lite Printing: NORIRDA.SYS

This (enhanced) IrDA Lite software is an alternate printing solution, in providing primary DOS character device driver support for IrDA printers.

Installation and Configuration

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

   `NORIRDA.SYS  –t:systemType [–n:driverName] [–d:n] [–r:n]`

where:

–t:systemType specifies the type of system being used. In this case, the number you select from the list: 6100, 6110, 6210, 6300, 6350, 6400, 6600, 6620, 6622, should be 6400. This parameter is NOT optional.

–n:driverName   specifies the new name to which this character device driver should respond. The default is IRDADRVR.

–d:n   specifies the disconnect time (in seconds) requested during negotiation where n is one of 3, 8, 12, 16, 20, 25, 30, or 40. This switch is accumulative and can be used to add any or all of the above disconnect times to the default of 3. The two systems negotiating will select the largest common value.

–r:n   specifies the number of seconds discovery retries will be done at the first DOS write command by the driver. The default is 5 seconds.

Battery Monitor: DOSGAS.EXE

This program monitors the charging and discharging of the main battery pack.

Usage

DOSGAS has optional parameters.

    DOSGAS.EXE [–Un] [–Pn] [–S] [–N] [–Ln] [–Fn] [x y]

where:

–S   Screen — causes the percentage display mode to be used instead of the annunciator icon mode (annunciator is on bottom line of display).

–Un   Update — time between updates if not in popup mode, or duration information is displayed if in popup mode. (n) is time, in milliseconds.

–Pn   Popup display mode — (n) is value of keystroke that will cause the information to display when that key is pressed.

–N   **No display** of icons on screen — DOSGAS runs in background. The only indication of low battery consists of audible warnings, when low capacity battery levels are crossed. This preserves last line of display.

–Ln   **Length** — (n) sets duration of each of the three beeps for low battery warning. Default is 0.

–Fn   **Frequency** — (n) sets pitch of the 3 beeps for low battery warning. Default is 0.

x y   Coordinates for a percentage display (e.g., 75%) on screen. Note the virtual screen on the 6400 computer starts at line 9 in standard 16 line by 20 column font size.

The –L and –F switches are options so the user can adjust the beep to a discernable sound in warehouse or other environment.

Four stages of visual and audible low battery warnings should occur according to the following intervals (except when –N option is chosen).

" NOTE 1:   *By default, frequency and duration of the beeps are set to zero. Switches –Fn and –Ln need to be set, or no beeps can be heard.*

" NOTE 2:   *For additional information on charging and annunciator display, refer to the* **Communications and Device Support** *section.*

High Level Applications

RF Applications

The Link Driver for 6400 Computer

For the appropriate radio type, use the link driver listed to the right, in the following table. The NET.CFG file should include the information listed to the right of the link driver:

Table 2-5
NET.CFG File Information

Radio	Link Driver	Included in NET.CFG	
Proxim PCMCIA 2.4 GHz	RL2PCM.COM	INT	15
		PORT	270
		DOMAIN	0
		MEM#1	C000
		INITIALIZE_365	N

Batch / Mobile Applications

Information that has been gathered during the use of your 6400 hand-held computer is sent to a centrally located host computer. This could be daily or several times during the day.

Terminal Emulation

This provides a means to emulate another computer. Keyboard overlays identify keys on the 6400 Computer as keys for the emulation. Refer to the 6400 User's Guide for samples of some of these overlays.

IrDA Communications

Downloading parameters, program data, or inventory updating is done through the IrDA port with your hand-held computer connected into a dock, or through the connector on the optional Communication Adapter endcap.

Client / Server Applications

TBD

ADK Application Porting

TBD

Emulations

TBD

Power Management

.. ..

Introduction

This section contains information about Advanced Power Management (APM) on the PEN*KEY® 6400 Computer. This following list can assist you in your search for topics of interest in this section:

The following list can further assist you in locating topics of interest:

Power Management BIOS: ELANAPM.EXE

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, functions specific to Intermec 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. 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) participates in managing power levels with 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 1—2 seconds.

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 6400 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.

Table 3-1
Activity Monitoring

I/O Address (Hex)	Device
000h—00Fh 080h—08Fh 0C0h—0DFh	} DOS\SYSTEM
022h—023h	ELAN
060h, 064h	Keyboard (060h and 64h)
2F8h—2FFh	COM2 (2F8h—2FFh)
3E8h—3EFh	COM3
3F8h—3FFh	COM1 (3F8h—3FFh)

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 is ignored and not cause the activity timers associated with *System Idle* or *System Suspend* to 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 also be entered when 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/O 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 will cause 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, or the
battery door latch is pressed. The *System Suspend* state is exited only by a
wakeup event.

There are many possible sources of wakeup events, and are defined by the 6400
Computer, as:

  " ON/OFF or SCAN button press
  " Application of external power (charge)

Wakeup events can be set or masked with 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 with the **Set Power State** function.

Device drivers or applications that control device power should be aware that,
when the system enters the *System Suspend* state, all peripheral devices
managed by the driver or application loses power during *System Suspend*, but
are 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.

" **NOTE:**   *During System Suspend, all power is removed from the PC Card slot. Therefore, if a radio
was in the slot, it would be powered off when the system is Suspended.*

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

The following table contains general definitions for each of the power states.

Table 3-2
Power States (General Definitions)

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

The power states for each device in the 6400 Computer are defined in the
following tables.

Table 3-3
Power States (Display)

State	Description
Ready	Enables LCD and places CGA controller into full-power state.
Standby	Undefined
Suspend	CGA controller placed into hardware suspend.
Off	CGA controller placed into hardware suspend.

Table 3-4
Power States (Serial Port)

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	Undefined
Suspend	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.
Off	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.

Table 3-5
Power States (System)

State	Description
Ready	No action
Standby	No action
Suspend	The system is placed into "Suspend." All devices are powered OFF when the system is suspended. The previous power state that existed before entering "Suspend" is unknown upon awakening.
Off	The system is placed into "Suspend." All devices are powered off, and are left in the power state they were set to before entering "Suspend."
Processing	No action is performed.
Rejected	No action is performed.

Table 3-6
Power States (Backlight)

State	Description
Ready	Backlight is turned on.
Standby	Not defined.
Suspend	Not defined.
Off	Backlight is turned off.

Power Device IDs

The APM Power Device Class/Subclass IDs are defined as follows:

Table 3-7
Power Device IDs (Register Values)

Value in Register BX	Power Device ID
XXh	Unit number (zero based)
FFh	All units in this class

Table 3-8
Power Device IDs (Descriptions)

ID (Hex)	Description
00XXh	System
00h	APM BIOS
01h	All devices power-managed by the APM BIOS
0100h	Display
0200h	Secondary storage (PC Card slot: dedicated to local area radio in 6400 Computer)
03XXh	Parallel ports
0400h	RS-232 serial port
0410h	RS-485 serial port
0500h	Reserved for Ethernet network adaptors
0700h–DFFFh	(reserved)
E000h–EFFFh	OEM-defined power device IDs
E200h	Backlight
F000h–FFFFh	(reserved)

APM Software Interface

The following paragraphs describe the software interface between the APM BIOS and the APM driver.

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.

The following is a list of the defined power management events that the APM BIOS can report:

Table 3-9
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 with this 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 when 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 before doing the suspend operation.

After a system is suspended and the system is resumed, the APM BIOS posts this event. 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 with the *Enable/Disable Power Management* call.

APM BIOS Functions

This paragraph describes the register interfaces to the APM BIOS functions. Functions are accessed through INT 15h. The carry (CY) 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.

The following is a summary of the interrupt 15h functions that are available for APM on the 6400 Computer:

Table 3-10
APM Function Summary

DOS APM Interrupts (INT 15h)	Function AL =
APM Installation Check	00h
APM Real Mode Interface Connect	01h
APM Interface Disconnect	04h
CPU Idle	05h
CPU Busy	06h
Set Power State	07h
Enable or Disable Power Management	08h
Get Power Status	0Ah
Get PM Event	0Bh
Get Power State	0Ch
Enable or Disable Device Power Management	0Dh

" **NOTE:**   *Function AH = 53h for all DOS APM Interrupts (INT 15h) and no connection is required.*

Refer to the *BIOS Interrupt Support* section of this publication, for descriptions of these interrupts. The *Interrupt Cross-Reference* table, in the *BIOS Interrupt section*, and the *Interrupt Index*, in the *Index* section can assist in locating the desired interrupts.

Radio Power Management

TBD

Firmware Error Codes

Table 3-11
Firmware Error Codes

Error (Hex)	APM Function Error Messages	Applicable APM Calls
01	Power Management functionality disabled.	Enable or 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 or 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	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
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 or Disable Device Power Management Enable or Disable Power Management Engage or 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 or Disable Device Power Management Enable or Disable Power Management Engage or 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,* for further details of these firmware error codes. The *Reference* section, in this publication, contains ordering information for that document.

Communications and Device Support

.. ..

Introduction

This section presents some basic communication methods and device support for the PEN*KEY® 6400 Computer.

The following list may be of further assistance in locating topics of interest.

Communications Using INTERLNK and INTERSVR

INTERLNK interconnects two computers through serial ports. INTERSVR is the INTERLNK server, a communication option in the NORAND Utilities program. For information to install INTERLNK, refer to the *Development Resources* paragraph in the *Getting Started* section. For details of INTERLNK and INTERSVR topics, refer to the DOS on-line help text.

NORAND Utilities: 64UTILS.EXE

The NORAND Utilities program, 64UTILS.EXE, provides the basic functions required to prepare the 6400 Computer for use. Refer to the *6400 User's Guide* 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**R** 6000 Series system.

Required System Files (NPCP)

To use the NPCP protocol, you must load the program **MININET.EXE.** You can install the MININET TSR by inserting the following statement into the **CONFIG.SYS** file:

```
install=mininet.exe  –c1 –t1152 –s1152
```

or by placing the following statement into the **AUTOEXEC.BAT** file:

```
mininet.exe –c1 –t1152 –s1152
```

Telecommunications Session Overview (NPCP and TTY)

Under the NORAND Standard File Transfer Session, the computer 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 on the format that the file uses.

The computer must first send a session control file, which identifies the computer to the host computer. The computer may then send a download request file. This optional file prepares the host computer to download one or more files to the computer. Upload files can then be sent to the host.

Session Control File

To identify itself to the host, the computer sends the session control file at the beginning of a telecommunications session. The host uses the application-dependent computer ID to identify the proper files for downloading to the computer. The format of the session control file is as follows:

`<DSCNTRL00001Xnnn>PPPPPPPPPPPPPPPTTTTTTTTTTTTTTTTTTTTYYMMDDHHMMSS . . .`

`<`	`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`	`HHC-ID (determined by application)`
`YYMMDD`	`date`
`HHMMSS`	`time`
`. . .`	`additional computer information`

The application determines the actual data in the session control file. The data fields shown in the preceding list are simply a convention in NORAND applications. However, the session control header will always consist of 18 bytes. Further, NORAND Host Communication Packages usually expect to find the HHC-ID starting in byte 33 of the data stream; this location is configurable, however.

Download Request File

A computer may directly request specific files from the host. To do this, it sends a download request file to the host just after sending the session control file. The files are not downloaded at that moment, but rather after the line has been 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 computer.

" **NOTE:**   *Only files requested in the download request file are downloaded.*

The following is the format of the download request file:

`<DDWNLRQnnnnnX016>[--filename1---][--filename2---] . . .`

`<`	`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 file name 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.

Keep in mind that while the download request file is sent to the host right after the session control file, files are not downloaded until after the line has been turned around.

Upload and Download Files

While the NORAND file transfer protocol supports a number of data types and file types, many of these are specific to the NORAND PL/N Language. If more information is needed on PL/N file formats, refer to *Writing TCOM Modules in PL/N for the Computer*, or your host telecommunications manual. For the sake of simplicity, DOS programmers can use one basic file header for all upload and download files. The following is the format of a typical file:

```
<DffffffmmmmmXnnn>data . . .
```

`<`	`Beginning of file header`
`D`	`File type`
`ffffff`	`Filename`
`mmmmm`	`Decimal number of records in file,` `Right-justified and zero-padded`
`X`	`Data type (X for character)`
`nnn`	`Record length, right-justified and` `Zero-padded`
`>`	`End of file header`

In NORAND Host Communication Packages, a record length of one indicates that the file contains variable length records, in which the first character of each record identifies the type of record. The formatting utilities of the communications packages use these single-character record types 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 has been defined:

```
<DDOSFIL00001Xmmm>F[--dosfilename--] S[filesize]data . . .
```

`<`	`Beginning of file header`
`D`	`File type (fixed)`
`DOSFIL`	`Filename (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 filename (this parameter 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 the binary file (this` `parameter may be up to 8 digits long)`
`data . . .`	`The binary file starts immediately after the` `filesize parameter.`

For example, to send a file named **DATA****DATAFILE. DAT** with a size of 102,000 bytes, the following header would precede the file:

> <DD0SFIL00001X027>F\DATA\DATAFILE. DAT S102000

It is the responsibility of the 6400 Application Program to ensure that the filename specified is unique on the host. This could be accomplished by incorporating the computer ID as part of the filename or path name:

> <DD0SFIL00001X027>F\DATA\DATA0001. DAT S102000
> <DD0SFIL00001X031>F\HH000001\DATAFILE. DAT S102000

Usage

To use **64UTILS. EXE,** use the following syntax on the command line:

> 64UTILS. EXE <ctrl-file>

where <**ctrl-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 of **64UTILS. EXE** is a session status code. Values for this code are described later under "Communication Log File".

PROTOCOL=

This parameter indicates the protocol to be used:

> 3 = TTY
> 4 = NPCP (LAN)

Example:

> PROTOCOL=3

COM=

This parameter is valid only if two-way TTY protocol is specified. It indicates the COM port to use for two-way TTY communications.

Valid values are 1–3

> 1 = COM1, 2 = COM2, 3=COM3

Default value is 1

Example:

> COM=3

SPEED=

This parameter indicates the TCOM speed for two-way TTY. This parameter should be omitted if NPCP protocol is specified.

Valid values are: 1200, 2400, 4800, or 9600

Example:

> SPEED=2400

PARITY=

This parameter is valid only if two-way TTY protocol was specified.

Valid values are: 0, 1, 2

> 0 = None (default)
> 1 = Odd
> 2 = Even

Example:

> PARITY=2

DATABITS=

This parameter is valid only if two-way TTY protocol was specified.
Valid values are 7, 8

Example:
DATABITS=7

STOPBITS=

This parameter is valid only if two-way TTY protocol was specified.
Valid values are: 1–3

    1 = One stop bit
    2 = Two stop bits
    3 = 1 1/2 stop bits

Example:
STOPBITS=1

MODEMTYPE=

This parameter is valid only if two-way TTY protocol was specified. It indicates the type of modem, if any, to which the computer is connected.
Valid values are: 0,1

    0 = No modem, such as direct connect *(default)*
    1 = NORAND Modem, or other Hayes-compatible modem

Example:
MODEMTYPE=1

CONFIG=

This is a modem command string to configure a Hayes-compatible modem. It does not need to be specified if you are using a NORAND Modem. If omitted, a standard configuration string is sent to the modem, based on the modem type and telecommunications speed.

Example:
CONFIG=ATE0V0Q0X4&C1&D2&M0&RS0=0

PHONE=

This modem command string dials a Hayes-compatible modem.
Example:
PHONE=ATDT3693361

TRIES=

This parameter specifies the number of times to attempt a successful communication session. If TRIES is exhausted before a session is successful, 64UTILS.EXE returns the error of the last session. A parameter value of zero indicates that the retry is to continue until a session is successful or the user aborts.

Example Control File for TTY:
    PROTOCOL=3
    COM=1
    SPEED=9600
    PARITY=0
    DATABITS=8
    STOPBITS=1

Example Control File for NPCP:
    PROTOCOL=4

Upload Control File

Information about files to be transferred must be contained in a file named NRUPLD.CTL. Valid parameters for this file are described in the following paragraphs.

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 is 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 filename specified in the FILE parameter is inserted into the header that is uploaded to the host.

Example:
```
HEADER=<DSCNTRL00001X042>
HEADER=<DBYPRD  00000X040>
HEADER=<DBYTRXN00000X001>
HEADER=<DDOSFIL00001X027>F\DATA\DATAFILE.DAT   S102000
HEADER=<DDOSFIL00001X024>F\RT000001\BINFILE.DAT   S
HEADER=<DDOSFIL00001X000>
HEADER=
```

RECTYPE=

RECTYPE and FORMAT parameters upload files that contain variable-length records, in which the first character of 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=

This parameter 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:
```
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, just 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 *Writing TCOM Modules in PL/N for the Computer*, or refer to the host telecommunications manual.

Example:
    FORMAT=X020
    FORMAT=N012X016N004

FILE=

This is the name of an upload file. Files are uploaded according to the preceding header or target information. Multiple FILE parameters may follow a single HEADER or parameter, if the header or target 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.

Example:
    HEADER=<DSCNTRL00001X042>
    FILE=SCNTRL. DAT

The file SCNTRL.DAT would contain the session control data of the form

    PPPPPPPPPPPPPPPTTTTTTTTTTTTTTTTTTTYYMMDDHHMMSS

as described earlier under *Session Control File.* This file could have any name. SCNTRL.DAT is used just as an 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

This parameter indicates the start of a telecommunications session.
   nnnn   Session number. Currently, the session number is always one.

Example:
   BEGS=1

UP=c:\pathname\filename,nnn

This parameter indicates that a file upload was attempted.
   nnn   Error code. Zero indicates a successful upload.

Example:
   UP=C:BYTRXN.DAT,0

DOWN=c:\pathname\filename,nnn

This parameter indicates that a file download was attempted.
   nnn   Error code. Zero indicates a successful download.

   Example:
      DOWN=C:\CUST.DAT,23

ENDS=x,m,n

This parameter indicates the end of a telecommunications session.
   x = Session status.
It is a single character code.
This code is also the return value of **64UTI LS. EXE.**

Code	Meaning
"G"	Good session
"T"	Unexpected end of transmission
"H"	Incorrect file header was encountered
"F"	File error was encountered
"L"	Telecommunications was aborted before the first file header was received

   m = Stage of the communications session.

Value	Meaning
5	Sign-on started
4	Data send started
3	Turn-around started
2	Data receive started
1	Sign-off started

   n = Protocol error code.

   This is a number that specifies the protocol error.

Protocol Errors

Table 4-1
TTY Protocol Errors

Error Code	Description
0	No error
6	User pressed **[EXIT]**, aborting communications
11	Invalid parameter specified in control file
23	End of transmission
101	Line lost
102	Parity error
103	Character gap too long
104	Data loss
105	Excessive NAKs (No AcKnowledgements)
106	Block count error
107	Block check error
108	Block framing error
109	Control character error
2xx	Modem error
101	
101	

2xx = one of the standard Hayes response codes, or a code defined by 64UTILS.EXE, where 2xx = one of the following codes:

Error Code	Description
286	Error sending command to modem
287	Expected numeric response not numeric
288	Invalid response format
289	No significant response from modem
298	Unrecognized English response

Table 4-2
NPCP Protocol Errors

Error Code	Description
0	No error
6	User pressed **[EXIT]**, aborting communications
11	Invalid parameter specified in control file
18	Open rejected
23	End of transmission
24	Session ended abnormally

Serial Communications

The optional communication adaptor endcap 9-pin D-Sub connector supports communications with a number of peripherals, using a serial port cable. These peripherals include battery chargers, printers, scanners, and docks.
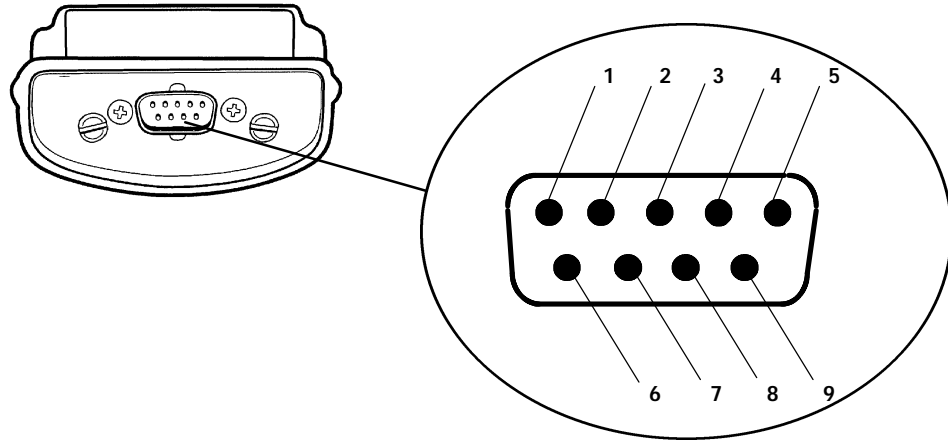

Figure 4-1
9-Pin D-Sub Connector Pinouts

The table below shows pin connections for the 9-pin connector on the communications adapter.

Table 4-3
9-pin D-Sub Connector Pinout

Pin Number	Signal	Function
1	No Connect	Data Carrier Detect
2	RXD	Receive Data
3	TXD	Transmit Data
4	Looped to DSR	Data Transmit Ready
5	GND	Ground
6	Looped to DTR	Data Send Ready
7	RTS	Request To Send
8	CTS	Clear To Send
9	Vcc	5 volts (See Caution, below)

B **CAUTION:** **The 5 volts on pin 9 is used with tethered scanners and printers only; it may damage modems.**

LAN Communications

Overview

MININET.EXE provides communications over Local Area Networks (LANs), using the following switches:

- –s   Speed
- –c   Comport
- –t   Toggle

4000 Series Backwards Compatibility

Using the following switches, the interrupt override from 4000API can be disabled:

     /10 /14 /16 /PC /C3

6400 Keyboard Definition and Redefinition

The following paragraphs present the definitions for the keys on the 6400 Computer and specify the interface used for keyboard redefinition.

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.

Keyboard Definition

Logical Keyboard

The logical keyboard is defined as the view of the keyboard that the programmer uses. That is, in terms of how each physical key is represented to the user. However, the logical keyboard usually defines 64 key numbers instead of only the 41 or 51 keys of the 6400 Keyboard.

Keyboard Overlays

There are overlays for the 41-key keyboard and the 51-key keyboard. Refer to the *6400 Hand-Held Computer User's Guide* (NPN: 961-047-093) for examples of overlays for the 6400 Computer. One example of a keyboard overlay for the 6400 Computer is the "standard DOS/PC" for use with most applications that would not otherwise require a specific overlay.

Physical Keyboard

The physical keyboard is defined to be the mechanical 41-key or 51-key keypad that is built into the 6400 Computer. Figure 4-2 and Figure 4-3 show the 41-key keyboard and the 51-key keyboard for the 6400 Computer, without an overlay.

" The 41 key keyboard function and numeric keys are single keystrokes. Alphabetical characters can be entered by using the green, blue, or yellow shift keys.
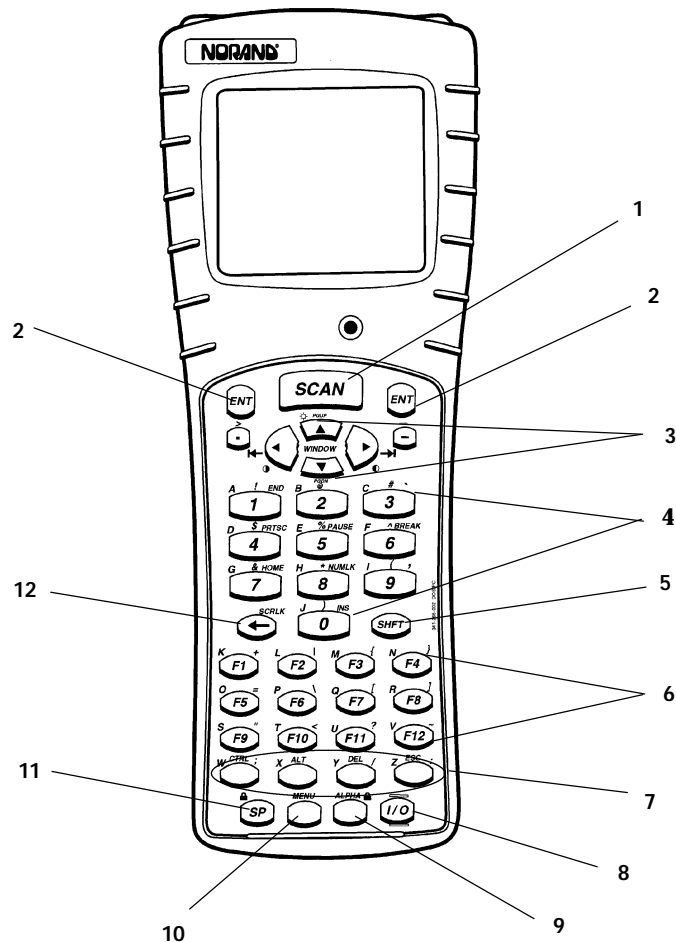
1. **Scan key**
2. **Enter keys**
3. **Windows panning (or scrolling) keys**
4. **Numeric keys (0–9)**
5. **Green shift key**
6. **Function keys (F1–F12)**
7. **Special function keys** *(depending on overlay used)*
8. **I/O (Suspend/Resume) key**
9. **Yellow shift key**
10. **Blue shift key**
11. **Space key**
12. **Backspace key**

Figure 4-2
41-Key Keyboard

" The 51 key keyboard alphabetic and numeric keys are single keystrokes. Special function keys can be entered by using the green, blue, or yellow shift keys.
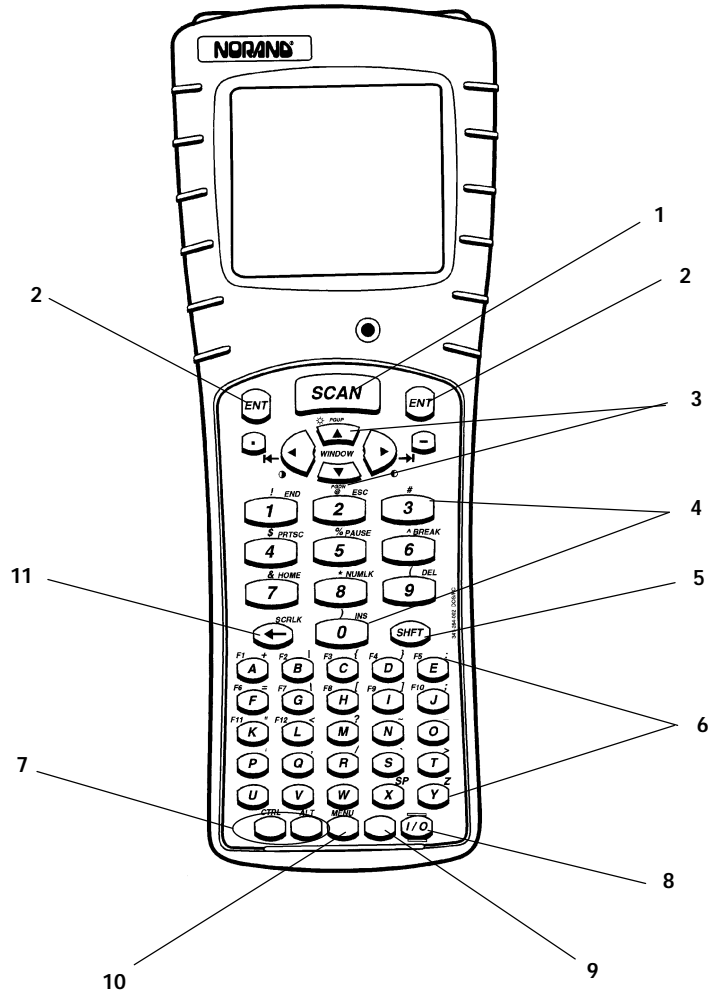
1. **Scan key**
2. **Enter keys**
3. **Windows panning (or scrolling) keys**
4. **Numeric keys (0–9)**
5. **Green shift key**
6. **Alphabetic keys (F1–F12)**
7. **Special function keys** *(depending on overlay used)*
8. **I/O (Suspend/Resume) key**
9. **Yellow shift key**
10. **Blue shift key**
11. **Backspace key**

Figure 4-3
51-Key Keyboard

Keyboard Redefinition

Most of the keys on the keyboard can be redefined, shifted and unshifted. The suspend/resume (I/O) key and the SCAN key cannot be remapped. Each key on the keyboard has a specific key code. Refer to any standard keyboard interface definition to determine keycodes for the keys on the 6400 Keyboard.

ANSI.SYS can be used to reassign keys through the use of escape sequences.

" **NOTE:**   *Keep in mind that ANSI.SYS only works with default fonts. This means if you are using any Intermec fonts, or other acquired fonts, you could encounter difficulties in remapping keys to with these fonts.*

An escape sequence is a string of text that begins with two preliminary keys, which includes the escape character and the left-bracket character "[". This is not the only way to remap keys. However, it is very portable. ANSI.SYS works the same across all PC compatible platforms. Characters, such as Enter, Space, and quoted text strings, can be redefined into an escape sequence. After these escape sequences have been issued to the system, you can press a "redefined" key to produce the remapped text string, key, or set of keys.

Example Key Redefinition, Using a TSR

There are other ways to remap keys, some of which allow mapping to non-standard fonts. A TSR program, such as the sample listed in *Appendix A, Configuration Files*, could be used remap keys to non-standard fonts.

Example Key Redefinition, Using ANSI.SYS

The following example demonstrates a method of remapping function keys F1 through F10, using ANSI.SYS. This method remaps keys, by supplying the appropriate content for each of the parameters that follow the "[". In the first example below, 0;59 is the first parameter. It is the key to be redefined. Everything after the first parameter is mapped to the key specified in the first parameter. Quoted text or key numbers can be used for any of the parameters.

Refer to the DOS help or the *Using ANSI.SYS* paragraph below, for additional information on remapping keys.

Add the following line to your CONFIG.SYS file:

```
device=c:\dos\ansi.sys /x
```

The /x switch selects keyboard redefinition, as specified in the DOS Help for ANSI.SYS.

Add the following assignments to your AUTOEXEC.BAT file, in order 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
```

The listing above, shows exactly how the text appears if you create it using a DOS text editor, such as EDIT.COM. The small arrow (<–) after the "prompt $" represents the escape character (in the escape sequence). This is similar to the way it appears using EDIT.COM. In order to insert the escape character with EDIT.COM, press [Ctrl-P], then the escape key.

For example, using EDIT.COM, the first assignment in the example would be entered as follows:

```
prompt $<Ctrl-P><Esc>[0;59;"INTERSVR C: D: ";13p
```

where <Ctrl-P> and <Esc> mean that you press the [Ctrl] key, press the "P" key, then press the [Escape] key. This inserts the escape character into the text for the escape sequence. The escape character may appear differently, depending on the text editor you are using.

The 0;59 indicates that the text in quotes is assigned to the [F1] key. The 13p (after several of the assignments) indicates that the [Enter] key is added after the text in quotes. The 32p indicates that a space character is added after the text in quotes, which allows for command line switches or other parameters.

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, it is important to note that the "echo on" must be placed before the "prompt" statements, so that all escape sequences are echoed to the screen. ANSI.SYS functions are executed through the display.

"NOTE: *ANSI.SYS distinguishes between uppercase and lowercase letters. For example, "P" and "p" have completely different meanings.*

Syntax

The following syntax is should be included in the CONFIG.SYS file:

```
device=[drive:][path]ANSI.SYS [/X]
```

where [drive:][path] specifies the location of the ANSI.SYS file in your system.

There are other syntax forms allowed for ANSI.SYS. Refer to DOS help on ANSI.SYS for further information.

Switches

```
/X
```

Remaps extended keys independently on PC-compatible keyboards.

There are other switches allowed with ANSI.SYS.

"NOTE: *Refer to DOS help on ANSI.SYS for further information.*

Sticky Keys

The operation of certain key combinations on the 6400 keyboard is somewhat different than the standard PC keyboard. The green SHFT, GOLD, BLUE, ALT, and CTRL keys are referred to as "Sticky" keys. The way a "Sticky Key" works, is as follows:

Press (and release) one of the "Sticky" keys. Then press another key.

Normal Results

Using these "Sticky" keys on the 6400 keyboard in the manner described in the previous paragraph, provides the same results as pressing and holding the shift key on the standard PC keyboard.

Some Unusual Results

If you press (and release) one of the "Sticky" keys, then press (and hold) a second key, then press yet a third key; the results are not as you might expect. The third key press causes the same result as if you were to press (and release) the "Sticky" key, then press the third key.

Some Examples

If you press (and release) the green SHFT key, then press the "e" key, the result is the upper-case "E". Whereas, if you press (and release) the green SHFT key, then press (and hold) the "e" key, then press the "f" key, the result is the upper-case "F".

Shift Keys

The 6400 Computer has three different "shift" function keys. These keys temporarily shift the function of the keyboard. Both the 41-key and the 51-key keyboards have a green SHFT key, a blue key and a yellow key. The green SHFT key is a standard shift key. Both the blue key and the yellow key work just like the shift key, allowing access to different alternate functions. There may be up to three sets of alternate functions on the keyboard. The blue key allows access to functions coded with blue characters on the keyboard. The yellow key allows access to functions coded with yellow, or gold, characters on the keyboard. The green SHFT key shifts between lowercase and uppercase characters, similar to a standard PC keyboard.

For more detailed explanations of keys on the keyboard, and how they work, refer to the 6400 Computer User's Guide.

Display

Video Modes

Refer to the *Controlling the Display with APIs* paragraph, page 4-24, for a list of APIs that can be used for controlling the video display from an application.

Standard Mode 3 vs Graphics Mode

The 6400 Computer is CGA compatible in standard modes. In these modes, you can write directly to the video memory.

Since the BIOS puts the system into graphics mode 43h, whenever the character size is changed from 8x8, the video memory could produce undesired results if your application writes directly to the video memory.

Therefore, if your application requires the system go into graphics mode, make sure the development platform is capable of directing the display output to the BIOS, rather than directly to the video memory.

Mode 43h

This 6400 specific video mode allows for larger and smaller characters than the typical 8x8 offered by the standard CGA modes. Once set in this mode, all video access should go through the BIOS or DOS interface.

" NOTE: *Blinking is not supported in Video Mode 43h.*

Cursor Mode

Within emulations, you can change the style of the cursor, using the "Cursor Mode" option in the "Display Options" menu. The possible choices are as follows:

- **Blinking Cursor** — Within emulations, the blinking cursor (represented by an underline character) only works with standard mode 3. This is the 18 rows by 20 columns mode. When the blinking cursor is enabled for other font sizes, the underline (text and cursor) does not work.
- **Block Cursor** — A "Block" cursor is indicated by a reverse video (highlighted) block (**J**) character.
- **Block Blink Cursor** — The "Block Blink" cursor is represented by blinking a reverse video block at the current position in the line.
- **Underlined Cursor** — An "Underlined" cursor is indicated by a line under the character at the cursor position.
- **Underline Blink Cursor** — The "Underline Blink" cursor is represented by blinking an underline below the character at the current position.

Backlight Control

The backlight time is adjusted by selecting Backlight from the Display Options menu. The value chosen sets the number of seconds the backlight stays on after a key is pressed. You can turn the backlight on or off with the *[green SHFT]* key and the *up arrow* key.

Refer to the *Sticky Keys* paragraph on page 4-17, for an explanation of the use of *[SHIFT]* keys on the 6400 Computer.

Contrast Control

The following procedures can adjust the contrast; but keep in mind, on a Four-Key reset, the system comes up with the default conditions set.

   " **Darker** — Press the *[green SHFT]* key once, then press the *right arrow* key to make the contrast darker. Repeat this key sequence to make it even darker. Holding the *right arrow* key down, after pressing the *[green SHFT]* key, also repeats the operation.

   " **Lighter** — Press the *[green SHFT]* key once, then press the *left arrow* key to make the contrast lighter. Repeat this key sequence to make it even lighter. Holding the *left arrow* key down, after pressing the *[green SHFT]* key, also repeats the operation.

Refer to the *Controlling the Display with APIs* paragraph, page 4-24, for a list of APIs that can be used to change the contrast from an application.

Refer to the *Sticky Keys* paragraph on page 4-17, for an explanation of the use of *[SHIFT]* keys on the 6400 Computer.

Display Annunciators

The annunciators show the current status of the operation in progress. Each annunciator has a designated position on the annunciator status line.

You can select and change the location of the annunciators on your display screen. The choices are any of the four corners of the display. The lower right corner is the default location. You can also select Stealth Mode, which only displays the annunciators when you make a change; then it disappears with the first key pressed.

Refer to the *6400 User's Guide* for additional information on the annunciators.

Refer to the *Controlling the Display with APIs* paragraph on page 4-24, for a list of APIs that can be used to change the annunciators from inside an application.

Refer to the *Battery Status* paragraph on page 4-28, concerning how the status is displayed with the battery annunciator.

" **NOTE:**   *In order to eliminate unnecessary activity, the annunciators are only updated when there is an actual change (i.e., from 3 to 2 icons), or when you exit a DOS program, or when you make an Intermec specific call to refresh the annunciators to current keyboard settings. Thus, if an application uses the annunciator line on the display, they may disappear for awhile, but should return when the status changes.*

Viewport

The 6400 display is only one-eighth the size of a full CGA screen. It is merely a viewport into the logical (or virtual) display. For applications that display information beyond the physical boundaries of the 6400 display, the viewport can be positioned to access all of the information, as described in the following paragraphs.

To control the viewport from an application, refer to the *Controlling the Display with APIs* paragraph, page 4-24, for a list of APIs.

Positioning the Viewport

Using MOVSCRN.EXE

The program, MOVSCRN.EXE positions the viewport to one of two specific locations, using the following commands. It can either be run from the DOS prompt or from the AUTOEXEC.BAT file.

To position the viewport at the bottom left corner of the logical display:

```
MOVSCRN   0
```

To position the viewport at the top left corner of the logical display:

```
MOVSCRN   1
```

Within an Application

To position the viewport, within an application, supply the following register values and call INT 10h.

If the viewport is not "locked", its position can be changed. An application can set the position of the viewport, using a API BIOS call, as described below.

See the *Window Pan Mode and Borders* interrupts, in the *BIOS Interrupt Support* section, for further details.

The position of the viewport is moved to the lower-left corner of the display, by setting the following registers, then call INT 10h:

```
   On entry:
     AX = 1200h
     BH = 00h
     BL = 0F6h
```

The position of the viewport is moved to the upper-left corner of the display, by setting the following registers and call INT 10h:

```
   On entry:
     AX = 1201h
     BH = 00h
     BL = 0F6h
```

Refer to the *Controlling the Display with APIs* paragraph, page 4-24, for other APIs that can be used for controlling the viewport from an application.

Panning the Viewport

The viewport can be panned (or scrolled) to view other portions of the virtual screen. If you hold the Blue Shift key down (item 10 in Figure 4-2 on page 4-13) and tap the Window Panning Keys (items identified as 3 in the same figure), you can pan up, down, right, and left.

Panning can also be programmed into an application, by utilizing the APIs listed in this section and in the *BIOS Interrupt Support* section of this publication.

The "Scroll Window" settings, in the "LCD Parms" menu, determine the increment at which it scrolls.

Setting the Viewport to Follow the Cursor

The specified coordinates merely provide a starting point for the viewport. When the cursor position exceeds the boundary defined by the parameters, the viewport follows it as text is entered.

Using MOVSCRN.EXE

Position the viewport at at a specified location on the logical display, and allow it to follow the cursor, with the command:

```
MOVSCRN  2 wwxx yyzz
```

where: ww = row for the top left corner;
        xx = the column for the top right left;
        yy = row for the bottom right corner;
        zz = column for the bottom right corner.

MOVSCRN.EXE requires that row/column for both the **wwxx** and the **yyzz** parameters be in the specific order shown above.

Within an Application

To set the viewport to follow the cursor, within an application, supply the following register values and call INT 10h.

An application can set the viewport to follow the cursor, using an API BIOS call. Set the following registers and call INT 10h to set the viewport size:

On entry:
```
AX = 1202h
BH = 00h
BL = 0F6h
CH = top row
CL = left col
DH = bottom row
DL = right col
```

The CX and DX registers define the window in which the cursor can move without the viewport moving. Once the cursor moves outside, the viewport follows.

When the cursor reaches the boundary defined by registers CH, CL, DH, and DL, the viewport will follow it as text is entered.

Locking the Viewport

After the previous code has been executed, you should lock it in place. Set the following registers and call INT 10h to accomplishes this:

On entry:
```
AX = 1206h
BH = 00h
BL = 0F6h
```

Unlocking the Viewport

Once the viewport is locked, you must unlock it before you can use the keys to move the viewport window, or to send commands to move it. Set the following registers and call INT 10h to unlock the viewport:

On entry:

```
AX = 1207h
BH = 00h
BL = 0F6h
```

Fonts, Rows, and Columns

When you change the font size, the number of rows and columns that can appear in the viewport is automatically changed. When the number of rows and columns is changed, the appropriate font size is automatically changed to match the rows and columns.

Changing Font Sizes

Using the *Set Physical Display Size* API, provides a means of changing the font size. This is done by setting the following registers, then calling INT 10h:

On entry:

```
AX = 1200h
BX = 00FAh                  (change screen size function)
CH = number of rows
CL = number of columns
```

Once this is done, the system is in NORAND CGA video mode 43h, in which only TTY read/writes through DOS or BIOS is permitted. Writing directly to video memory does not work correctly.

Standard Display

This display (formerly the Wide Display) is a monochrome LCD with 160 pixels in width and 128 pixels in length. The character size is changeable by parameter menus. The default size for this display is 20 columns by 16 rows. Other available sizes are as follows:

```
Rows:       5,   8,  10,  16
Columns: 10,  13,  16,  20,  26,  32
```

Any of these combinations of rows and columns are available, using the LCD Parms menu.

Display Used on Older Units

" **NOTE:**   *This display (formerly the Standard Display) has been discontinued. The description is included here for the benefit of those who may still be using the display described in this paragraph. Refer to the Standard Display (formerly the Wide Display) for details of the display currently being used.*

This display is a monochrome LCD with 160 pixels in width and 144 pixels in length. The character size is changeable by parameter menus. The standard size is 20 columns and 18 rows. Other available sizes are as follows:

```
Rows:       6,   9,  12,  18
Columns: 10,  13,  16,  20,  26,  32
```

Any combination of rows and columns are available, using the values listed above.

The program, ROWCOL.EXE, can set screen size. Values can be entered at the DOS command line; or entering no parameters, the program prompts you for the values.

The following is an example of using command line parameters to set the screen size for 18 rows and 20 columns.

```
ROWCOL 18 20
```

The following is an example of allowing the program to prompt for the values to set the screen size to 12 rows and 32 columns. You enter the name of the program, without any parameters. The program prompts you for the "Number of rows"; and you respond with the value for the number of rows you want. The program prompts you for the "Number of columns"; and you respond with the value for the number of columns.

```
ROWCOL
Number of rows>12
Number of columns>32
```

Non-PC Fonts

The fonts for user-defined, non-PC, fonts are stored in upper flash. If you have any of these fonts that need to be used, they must be downloaded each time the screen size API is called; and must be done after the screen size is set. There is an INT 10h API that can be used to replace any font. Set the following registers and call INT 10h to accomplish this:

```
On entry:
  AX = 1400h
  BL = 00h
  BH = number of bytes per character
  CX = number of characters to store
  DX = character offset into font of first character to store
  ES:DI = address of start of user font table
```

Unavailable Characters

Several characters, from the upper 128 characters in the 255 character set (0x80 through 0x83), are unavailable for use, as well as several characters from the character set between 00h through 20h. This also means that some of the international characters are missing. These characters are dedicated for use with the annunciators.

MISCTSR.EXE provides the use of different set of characters.

–F1 provides use of an alternate character set, which includes:
00h–06h, 09h, 0Bh, 0Fh, 1Ah, 1Bh, 7Fh, FEh, and FFh.

–F0 option restores the default character set, which consists of the following:
10h–16h, 1Ah, 1Bh, 1Eh, and 80h–84h.

Controlling the Display with APIs

You can control the display from an application, using INT 10h APIs. The categories in the table below have been alphabetized.

Refer to the BIOS Interrupt Support section for detailed interrupt definitions of these and other display functions.

Table 4-4
INT 10h Display Services

INT 10h Display Services	AH=	AL=	BL=
3x4x Specific Modes			
Select Chase Mode	12h	E0h	FCh
Select Fixed Mode	12h	E1h	FCh
Select Setparms Mode	12h	E2h	FCh
Select Graphics Mode	12h	E3h	FCh
Annunciator			
Reset Annunciator Line to Current Keyboard Settings	12h	03h	F6h
Enable/Disable Annunciator Panning	12h	04h	F6h
Write BIOS Annunciator Values *(with attribute)*	12h		F8h
Write BIOS Annunciator Values *(without attribute)*	12h		F9h
Backlight Control			
Turn Backlight On	12h	00h	FFh
Turn Backlight Off	12h	01h	FFh
Toggle Backlight	12h	02h	FFh
Read Backlight Intensity	12h	03h	FFh
Set Backlight Intensity	12h	04h	FFh
Set Backlight Timer	12h	05h	FFh
Refresh Backlight	12h	06h	FFh
Read Backlight Timer	12h	07h	FFh
Character and Attribute			
Read Character and Attribute at Current Cursor Position . .	08h		
Write Character and Attribute at Current Cursor Position .	09h		
Write Character (no attribute) at Current Cursor Position . .	0Ah		
Teletype Character Write	0Eh		
Contrast Control			
Get Contrast (temperature compensated)	12h	00h	FEh
Set Contrast (temperature compensated)	12h	01h	FEh
Get Contrast (absolute, no compensation)	12h	02h	FEh
Set Contrast (absolute, no compensation)	12h	03h	FEh
Compensate Contrast	12h	05h	FEh
Turn Contrast Heater Off	12h	06h	FEh
Turn Contrast Heater On	12h	07h	FEh
Cursor			
Set Cursor Type	01h		
Set Cursor Position	02h		
Read Cursor Position and Type	03h		

INT 10h Display Services (continued)	AH=	AL=	BL=
Display			
Set Display Mode	00h		
Turn Display On	12h	00h	36h
Turn Display Off	12h	01h	36h
Fonts			
Return Pointer to Current Display Parameter Packet and Font Information	12h		FBh
Load User Font	14h	00h	
Graphics			
Write Graphics Dot	0Ch		
Read Graphics Dot	0Dh		
Page			
Set Active Display Page	05h		
Size			
Set Physical Display Size	12h		FAh
Return Physical Display Size	12h		FDh
Scrolling			
Scroll Window Up (active page)	06h		
Scroll Window Down (active page)	07h		
Viewport			
Set Viewport position to Lower-Left Corner	12h	00h	F6h
Set Viewport position to Upper-Left Corner	12h	01h	F6h
Set Viewport to Follow Cursor	12h	02h	F6h
Get Current Viewport Location	12h	05h	F6h
Lock Viewport	12h	06h	F6h
Unlock Viewport	12h	07h	F6h
Miscellaneous			
Read Current Video Mode	0Fh		
Draw Pixel Map	F0h		
Refresh the Screen	FFh		

Memory

The 6400 Computer uses two types of memory: Main memory DRAM and flash memory. The main memory is protected by the backup battery during low or dead main battery conditions.

Creating RAM and Flash Disks

Refer to the *Reprogramming Flash Memory* paragraph, in the *Getting Started* section of this publication for a detailed description.

Flash File System

The flash memory stores files for BIOS and DOS. The software stored in flash is copied into DRAM during execution of that software. Flash memory could also store information during operation. This information stored in flash memory is retained, even if all of the power sources have been depleted.

RAM Disk

Refer to *The RAM Drive* paragraph, under the *Development Resources* paragraph, in the *Getting Started* section of this publication, for a detailed description.

Upper Memory Provider: ELANUMP.SYS

ELANUMP.SYS is an upper memory provider. 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 C000h–EFFFh, on a 6000 Series Computer. You can use it in conjunction 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 high memory area. 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 also much smaller and simpler than EMM386. Since EMM386 is a virtual mode system extension, it also slows down processing somewhat 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 automatically scanned for, and reserved (that is, excluded from being used). 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 C400, C800, CC00, D800, DC00, and E000. Note that the last block is 32KB in size instead of 16 KB. This is a hardware constraint defined by the memory controller in the processor chipset.

The 6400 Computer provides upper memory (C400–CFFF, D800–E3FF) and HMA memory, using both **HIMEM.SYS** and **ELANUMP.SYS**:

   DEVICE=HIMEM.SYS
   DOS=HIGH
   DEVICE=ELANUMP.SYS /X=C000, D000, D400, E400, E800, EC00
   DOS=UMB
   DEVICEHIGH=...

Where:

   " C000 is for the Proxim 2.4GHz radio card.

   " D000, D400 are for the Flash File System.

   " E400, E800, and EC00 are for DOS.

Battery

The 6400 Computer contains a rechargeable lithium ion battery pack for the main power source and a nickel-metal hydride battery for the backup power source.

" **NOTE:**   *For detailed information on DOSGAS.EXE, charging, and annunciator display, refer to the Supported DOS Applications section.*

Battery Management

Charge Rate

Before using a new battery pack, it needs to be charged for twelve hours in a dock, or an AC power cube and communications adapter.

The charge rate should average about 1 Ampere.

The normal charge time, from empty to full, for the main battery pack is 1.5 hours for a standard 2 cell pack, and 3 hours for a 4 cell pack.

The battery packs are rated for 500 full charge and discharge cycles, after which you can expect only 50 percent of their original capacity. The main and backup batteries do not charge below 0 °C (32 °F) and above 40 °C (104 °F). The life of each cell in the battery is reduced when running in cold temperatures.

Discharge Rate

The maximum discharge rate is 2 Amperes.

A fully charged battery pack, left outside of a charger or your 6400 Computer, self-discharges in about 20 days.

Capacity

You can check to see if the batteries need to be replaced, by touching the top two contacts on the battery with your fingers. The capacity, or charge level, remaining in the battery is indicated by the number of LEDs that are lighted, as follows:

   " One LED = 21 to 25 percent charge level
   " Two LEDs = 26 to 50 percent charge level
   " Three LEDs = 51 to 75 percent charge level
   " Four LEDs = 76 to 100 percent charge level

The 960 mAH lithium ion (LiION) battery pack has been replaced by the 1350 mAH lithium ion battery pack, PN 318-001-001.

Battery Status

The LiION battery pack sends status packets through I/O addresses 278–27Fh, which are captured by ELANAPM.EXE and DOSGAS.EXE. DOSGAS then displays the capacity on the screen. ELANAPM forwards the battery capacity and number of charge cycles to the APM 1.1 functions that can be called by your application and by NORDOSPM.EXE.

Annunciators

The battery annunciators indicate the level of the main battery.

Table 4-5
Low Battery Indications

Low Battery Level	% Capacity	Icons	Beeps
0	76—100%	BBBB	
	51—75%	BBB	
	26—50%	BB	
	21—25%	B	
1	16—20%	LLLL	
2	11—15%	LLL	
3	6—10%	LL	Three beeps on 10%
			Three beeps on 7%
4	0—5%	L	One 5 second beep on 5%

" **NOTE:**   *The Bs in the preceding table are actually displayed on the 6400 Computer as small battery icons.*

When the capacity reaches 20%, the single annunciator icon changes to four Ls, as described in the preceding chart.

When unit is charging, the annunciator icons change to Cs, as shown below.

Table 4-6
Charging Indications

% Capacity	Icons
25%—0%	C
50%—26%	CC
75%—51%	CCC
100%—76%	CCCC

" **NOTE:**   *There may be a delay of one minute for the icons to update when the battery pack is replaced.*

Terminal Emulations

If there is no main battery pack, or the charge level in the pack cannot be determined, only a single 'C' is displayed.

Also note that when the unit first boots up, or after a Suspend/Resume, it displays a single 'C', until the unit obtains valid information from the battery pack. After 10–50 seconds, if valid information is received the number of Cs may increase which reflects the charge level of the battery pack.

External Power

Any time external power is supplied to the 6400 Computer, the icons randomly switch between Cs and battery icons.

Internal Power

The standard annunciators are displayed whenever the unit is not powered by an external power source.

Charging Your Battery Pack

The battery pack can be charged in your 6400 computer, or in a charger. When you remove fully charged batteries from the charger, it is best to place it directly into a 6400 computer.

The following devices can be used to charge battery packs:

- " Single pack charger
- " Multipack charger
- " Single dock
- " Multidock
- " Vehicle dock
- " Communication adapter with AC power cube

Troubleshooting

Battery Charger LED and Error Codes

When your battery pack is placed in either a multipack or a single pack charger, the LEDs on the charger indicate the battery pack charging status. This indicates the charge capacity remaining in the battery.

The battery pack charger cups and docs have a tri-color LED. A constant red LED indicates the battery is charging. A constant green LED indicates the battery pack is fully charged.

If one of the following conditions exist, take the action listed:

Indication	Meaning	Action
No LEDs lit	No power or no connection	Reinsert into charger, Turn charger power on, Check for bad charger
A combination of blinking LEDs	A bad connection:	Remove and reinsert battery pack into charger
	A faulty battery:	Replace it
	A more serious charging or product error:	See chart below

The following chart shows a list of serious error codes and possible solutions. If the battery cannot be successfully charged, one or more LEDs flash. By observing the sequence of colors, use the table below (and the *Possible Solutions* on page 4-31) to determine the error code:

Table 4-7
Battery Charger LED Error Codes

Color Sequence				Description
Orange	Orange	Orange	Off	Battery pack in "fault" state (see 1 and 2 below)
Red	Orange	Red	Off	No packets in charger, after 90 seconds (see 1 below)
Orange	Red	Red	Off	Checksum error (see 1 below)
Red	Red	Red	Off	Upper cell voltage fault (see 1 and 3 below)
Red	Red	Orange	Off	Lower cell voltage fault (see 1 and 3 below)
Red	Orange	Orange	Off	Received five sequential bad packets (see 1 below)
Orange	Orange	Red	Off	Wrong type of battery (see 4 below)
Orange	Red	Orange	Off	Wrong type of battery (see 1 below)

Charger Problems and Solutions

Perform the following check, if one of the states above referenced note 1:

1. To isolate the issue to the 'faulting' battery pack, place it into a 6400 computer and operate the computer for a few minutes. Then try the battery pack in different charger cups to see if the LED indicators still create this fault. If this fault continues to show up, send the battery pack to your service center.

Perform the following check, if one of the states above referenced note 2:

2. This issue could be caused by a bad charger cup or a bad 6400 computer. Insert different batteries into this same charger cup. If the cup produces this same fault state (in the other batteries), then this cup on the charger is bad. To get the battery back into a good state, put it back into a 6400 computer, and operate it for a few minutes. Then place the battery pack on a good charger cup.

Perform the following check, if one of the states above referenced note 3:

3. Leave the battery pack in the charger for i minimum of 16 hours. If the error condition still exists, send the battery pack to your service center.

Perform the following check, if one of the states above referenced note 4:

4. Remove the battery pack and replace with the appropriate battery.

Common Problems and Solutions

Symptom	Possible Solution
A fully charged battery pack is inserted into the 6400 computer. The unit powers down. The battery pack appears dead (no LEDs lit on gas gauge).	Leave the same battery pack in the 6400 computer for at least thirty minutes. Then replace the battery pack with a fully charged battery. Place the original battery pack into an external charger.
The gas gauge on the display does not match the gas gauge on the battery pack.	Wait: It takes up to one minute to update the gas gauge on the display.
When a fully charged battery pack is inserted into the 6400 computer, after a weekend, it does not last the normal full length of time.	You should always keep a battery pack in the 6400 computer (while not in use), especially over a weekend.
The 6400 computer performed a reboot when the battery pack is changed.	♦ Always suspend the 6400 computer (press the I/O button) before you change the battery. ♦ The backup battery may have completely drained. To overcome this, charge the backup battery. ♦ If the problem persists, send the unit into your service center.
Different colors of charge LEDs blink when a battery is inserted into the charger cup.	There is a fault state occurring , either in the battery pack or the charger. Refer to Possible Solutions, above, to isolate the issue.

Beeper

The computer has a beeper that emits a sound under different conditions. *Key Click* is the sound that is made when a key is pressed. *Error Tone* is the sound that is made when an error occurs (for example, pressing an invalid key). When you make a good scan, a sound is also heard.

Adjusting Volume, Length, and Frequency

The setup (or adjustments), for *Key Click* and *Error Tone*, are supported through the Terminal Emulation Menu screens (if your hand-held is emulation equipped). Refer to the 6400 User's Guide for detailed descriptions of the Terminal Emulation Menu Screens.

NORAND Utilities also provides options for adjusting these parameters.

The volume parameter cannot currently be adjusted. To adjust the length and frequency parameters, refer to the following paragraphs.

Using the Terminal Emulation Menu

" **Key Click** — can be turned off or on, as desired. The "Key Click" option, in the "Beeper Setup" menu, adjusts the frequency and duration of a sound.

" **Error Tone** — option, in the "Beeper Setup" menu, adjusts the frequency and duration of a sound.

" **Internal Beeper** — can also be turned off by means of the "Beeper Select option", in the "Beeper Setup" menu.

Using NORAND Utilities

Select the *Advanced Functions* option, and choose the *Set Key Click* suboption to adjust the frequency and duration of the sound.

Within an Application

Use the *Set Key Click Parameters (AX=0402h)* API, the *Set Error Beep Parameters (AX=0404h)* API, and the *Beep the Buzzer, AX=0409h*, API.

" Register BL is the volume adjustment (currently not implemented).
" Register pair CX is the adjustment for length, in milliseconds.
" Register pair DX is the adjustment for frequency, in Hertz.

Use the *Get Key Click Parameters, AX=0403h*, API and the *Get Error Beep Parameters, AX=0405h*, API to read the parameter values.

Refer to the *BIOS Interrupt Support* section for further details.

BIOS Interrupt Support

.. .. " .. ..

Introduction

This section contains detailed descriptions of the BIOS interrupts supported for the PEN*KEY*r* 6400 Computer. Several methods have been provided for locating the interrupts and their definitions, as described below:

- " The **Topic Summary**, below, contains pointers to each of the major topics in this section.
- " The **Interrupt Cross-Reference** table, at the beginning of the topic entitled, *Cross-Reference by Interrupt Number*, has been organized by interrupt number, in ascending order.
- " The **Interrupt Index**, at the end of the *Index* section, has been organized alphabetically, to assist in locating the interrupts by topic.

Topic Summary

Cross-Reference by Interrupt Number

The table below contains a list of interrupts supported for the 6400 Computer. They are organized by interrupt number, in ascending order.

" NOTE: *A checkmark in the column labeled as, **Ext.**, indicates the interrupt is an extensions to the standard BIOS. Whereas, a blank means it is supported by the standard BIOS.*

Table 5-1
Interrupt Cross-Reference

INT 09h Keyboard Data Ready Services	Ext.	AH=	AL=	BH=	BL=	Page
(This is the keyboard hardware interrupt)						5-6

INT 10h Display Services	Ext.	AH=	AL=	BH=	BL=	Page
Set Display Mode		00h				5-7
Set Cursor Type		01h				
Set Cursor Position		02h				5-8
Read Cursor Position and Type		03h				
Set Active Display Page		05h				
Scroll Window Up (active page)		06h				5-9
Scroll Window Down (active page)		07h				
Read Character and Attribute at Current Cursor Position		08h				5-10
Write Character and Attribute at Current Cursor Position		09h				
Write Character (no attribute) at Current Cursor Position		0Ah				5-11
Write Graphics Dot		0Ch				
Read Graphics Dot		0Dh				5-12
Teletype Character Write		0Eh				
Read Current Video Mode		0Fh				
Alternate Select Functions						
Turn Display On		12h	00h		36h	5-13
Turn Display Off		12h	01h		36h	
Set Window Pan Mode and Borders						
Set Viewport position to Lower-Left Corner	p	12h	00h		F6h	
Set Viewport position to Upper-Left Corner	p	12h	01h		F6h	
Set Viewport to Follow Cursor	p	12h	02h		F6h	5-14
Reset Annunciator Line to Current Keyboard Settings	p	12h	03h		F6h	
Enable/Disable Annunciator Panning	p	12h	04h		F6h	
Get Current Viewport Location	p	12h	05h		F6h	5-15
Lock Viewport	p	12h	06h		F6h	
Unlock Viewport	p	12h	07h		F6h	
Write BIOS Annunciator Values *(with attribute)*	p	12h			F8h	5-16
Write BIOS Annunciator Values *(without attribute)*	p	12h			F9h	
Set Physical Display Size	p	12h			FAh	
Return Pointer to Current Display Parameter Packet and Font Information	p	12h			FBh	5-17

INT 10h Display Services (continued)	Ext.	AH=	AL=	BH=	BL=	Page
Select 3x4x Specific Display Modes						
Select Chase Mode	p	12h	E0h		FCh	
Select Fixed Mode	p	12h	E1h		FCh	
Select Setparms Mode	p	12h	E2h		FCh	5-18
Select Graphics Mode	p	12h	E3h		FCh	
Return Physical Display Size	p	12h			FDh	
Contrast Control						
Get Contrast (temperature compensated)	p	12h	00h		FEh	5-19
Set Contrast (temperature compensated)	p	12h	01h		FEh	
Get Contrast (absolute, no compensation)	p	12h	02h		FEh	
Set Contrast (absolute, no compensation)	p	12h	03h		FEh	5-20
Compensate Contrast	p	12h	05h		FEh	
Turn Contrast Heater Off	p	12h	06h		FEh	
Turn Contrast Heater On	p	12h	07h		FEh	5-21
Backlight Control						
Turn Backlight On	p	12h	00h		FFh	
Turn Backlight Off	p	12h	01h		FFh	
Toggle Backlight	p	12h	02h		FFh	5-22
Read Backlight Intensity	p	12h	03h		FFh	
Set Backlight Intensity	p	12h	04h		FFh	
Set Backlight Timer	p	12h	05h		FFh	5-23
Refresh Backlight	p	12h	06h		FFh	
Read Backlight Timer	p	12h	07h		FFh	
Load User Font		14h	00h			5-24
Draw Pixel Map	p	F0h				
Refresh the Screen		FFh				

INT 13 Disk Services	Ext.	AH=	AL=	BH=	BL=	Page
Reset Disk System		00h				5-26
Get Status of Last Operation		01h				
Read Sectors into Memory		02h				5-27
Write Disk Sectors		03h				5-28
Verify Disk Sectors		04h				5-29
Format Track (floppy disk)		05h				5-30
Get Drive Parameters		08h				5-31
Get Disk Type		15h				5-32
Detect Disk Change		16h				
Set Disk Type for Format (floppy disk)		17h				5-33
Set Media Type for Format		18h				

INT 13h Extensions to Std. Disk Services BIOS	Ext.	AH=	AL=	BH=	BL=	Page
Disable Access to RAM Drive Memory	p	DAh				5-35
Enable Access to RAM Drive Memory	p	EAh				
Enable Checksum of RAM Drive Integrity Memory	p	ECh				

INT 14h Communications Services	Ext.	AH=	AL=	BH=	BL=	Page
Initialize Communications Port		00h				5-36
Transmit a Character		01h				5-37
Receive a Character		02h				5-37
Read Communications Port Status		03h				5-37
Extended Initialize		04h				5-38
Read Modem Control Port		05h	00h			5-39
Write Modem Control Port		05h	01h			5-39

INT 15h System BIOS Services	Ext.	AH=	AL=	BH=	BL=	Page
Scanner Task Switch	p	01h	0Ch			5-41
Beep the Buzzer	p	01h	FAh			5-41
Query A20 Gate Support		24h	03h			5-41
Request Low Power Sleep		42h	FFh			5-42
Keyboard Intercept		4Fh				5-42
APM Installation Check		53h	00h			5-43
APM Real Mode Interface Connect		53h	01h			5-44
APM Interface Disconnect		53h	04h			5-44
CPU Idle		53h	05h			5-45
CPU Busy		53h	06h			5-45
Set Power State		53h	07h			5-46
Enable/Disable Power Management		53h	08h			5-47
Get Power Status		53h	0Ah			5-47
Get PM Event		53h	0Bh			5-48
Get Power State		53h	0Ch			5-49
Enable/Disable Device Power Management		53h	0Dh			5-50
OEM Installation Check		53h	80h	7Fh		5-50
Get System Information	p	53h	80h	C3h		5-51
Device Open		80h				5-51
Device Close		81h				5-52
Program Termination		82h				5-52
Set Event Wait Interval		83h	00h			5-53
Cancel Event Wait Interval		83h	01h			5-53
Read Joystick Input Settings		84h				5-53
System Request (SysReq) Key Pressed / Released		85h				5-54
Wait (Elapsed Time)		86h				5-54
Copy Extended Memory Block		87h				5-55
Read Extended Memory Size		88h				5-55
Switch Processor to Protected Mode		89h				5-56
Device Busy Loop		90h				5-57
Complete the Interrupt		91h				5-58
Get System Configuration Parameters		C0h				5-59
Return Extended BIOS Data-Area Segment (EBDA)		C1h				5-60

INT 16h Keyboard BIOS Services	Ext.	AH=	AL=	BH=	BL=	Page
Read Next ASCII Character from Keyboard		00h				5-61
Get Keyboard Status		01h				5-61
Get Shift Status		02h				5-62
Buzzer Control						
Turn Off Key Click		04h	00h			5-62
Turn On Key Click		04h	01h			5-62
Set Key Click Parameters	p	04h	02h			5-63
Get Key Click Parameters	p	04h	03h			5-63
Set Error Beep Parameters	p	04h	04h			5-63
Get Error Beep Parameters	p	04h	05h			5-64
Sound Key Click	p	04h	06h			5-64
Sound Error Beep	p	04h	07h			5-64
Select Buzzer	p	04h	08h			5-64
Beep the Buzzer	p	04h	09h			5-65

INT 1Ah Timer and Real-Time Clock Services	Ext.	AH=	AL=	BH=	BL=	Page
Read System Timer Ticks		00h				5-66
Write System Timer Ticks		01h				5-66
Read the Real-Time Clock Time		02h				5-67
Set the Real-Time Clock Time		03h				5-67
Read the Real-Time Clock Date		04h				5-68
Set the Real-Time Clock Date		05h				5-68
Set the Real-Time Clock Alarm		06h				5-69
Reset the Real-Time Clock Alarm		07h				5-69
Read the Real-Time Clock Alarm		09h				5-70
Load the Real-Time Clock to Internal Clock		FFh				5-70

INT 7Ah Scanner Services — 6400IKPS.EXE	Ext.	AH=	AL=	BH=	BL=	Page
Scanner On	p	80h				5-71
Scanner Off	p	81h				5-71
Set Scanner Type	p	82h				5-71
Set Bar Code Options	p	83h				5-72
Read Bar Code Data	p	84h				5-76
Flush Scanner Concatenation Buffer	p	FEh				5-77

Interrupt Definitions

This paragraph contains a detailed listing of the interrupts supported for 6400 applications.

To **search for an interrupt by interrupt number**:

- refer to *Table 5-1, Interrupt Cross-Reference*, on page 5-2, in this section, (which will direct you to the page where an interrupt is defined)
- thumb through the detailed interrupt listing

To **search for an interrupt alphabetically**:

- use the *Interrupt Index*, at the end of the *Index* section
   (which will direct you to the page where an interrupt is defined)

To determine what applications support each of these interrupts, a reference token has been included at the beginning of each interrupt definition. This token is immediately following the interrupt title, and is a logical name that represents either the BIOS or the application handling the interrupt.

The reference tokens for the BIOS and the applications are as follows:

System BIOS	**BIOS**
Power management BIOS application, . . ELANAPM.EXE . . .	**ELANAPM**
DOS power management driver, ... NORDOSPM.EXE .	**NORDOSPM**
Miscellaneous TSR, ... MISCTSR.EXE	**MISCTSR**
Scanner Services, ... 6400IKPS.EXE	**SCANBIOS**
RF BIOS	**RFBIOS**
NORAND APM OEM Functions, ... 64APMOEM.EXE .	**64APMOEM**

In some cases, there may be more than one application supporting a particular interrupt. In that case, you will find more than one token at the beginning of that interrupt definition. For example, if the tokens for both **ELANAPM** and **RFBIOS** are listed for a particular interrupt, that means the interrupt is supported by both the power management BIOS, ELANAPM.EXE, and the RF BIOS application.

" **NOTE:**   *All functions involving the carry flag, return with the carry flag set if there is an error. Otherwise, the carry flag is cleared.*

INT 09h, Keyboard Data Ready Services

BIOS

INT 09h is generated from IRQ1 when data is received from the keyboard. This is normally a scan code (from either a keypress or a key release); but could also be from an ACK or a NAK of a command. IRQ1 could be masked by setting bit 1 on I/O port 21h.

After reading the scan code from the keyboard (at port 60), INT 15/AH=4Fh is called. This is the keyboard intercept, which converts the keyboard matrix scan codes to PC compatible scan codes. All further processing uses this PC scan code.

Refer to a DOS reference book for details of these scan codes.

See Also: INT 15/AH=4Fh, INT 15/AH=85h, INT 16

INT 10h, Display Services

INT 10h, Function AH = 00h; Set Display Mode

BIOS

The following actions are taken when this function is called:

   " The start address of the REGEN buffer data item is initialized to zero.

   " The crt_mode parameter is set to the selected mode.

   " The crt_cols parameter is set to number of columns for the selected mode.

   " The page_len parameter is set to 1000h for 80x25 modes, and to 800h for 40x25 modes.

   " The page_offset parameter is set to zero for all modes.

   " The cursor_loc parameter is initialized to zero for all modes.

   " The curstype parameter is set to 0x607 for all modes.

   " The curpage parameter is set to 0 for all modes.

   " If bit 7 of register AL is set, the display buffer is not cleared.

At power up, the default mode of 3 (80x25 Color Text) is set by the 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

On Return:
   None

INT 10h, Function AH = 01h; Set Cursor Type

BIOS

Sets the cursor type. The cursor is not displayed in graphics modes and the selected cursor type applies to all video pages. The cursor *stype* parameter is set to the value in register-pair CX. The cursor is displayed on (up to) 8 scan rows, numbered 0 (top) to 7 (bottom). The default cursor type is the same as when settings are CH = 6, CL = 7. When bit 5 of CH is set, the cursor disappears.

On Entry:
   AH = 01h
   CH = Starting scan row in character box
       (range: 0 to 7)
   CL = Ending scan row in character box
       (range: 0 to 7)

On Return:
   None

INT 10h, Function AH = 02h; **Set Cursor Position**

BIOS

Sets the cursor position for the display page represented by register BH. This function applies to all video modes, although the cursor is not visible in graphics modes. A page does not need to be currently active for the function to perform properly. Attempting to move the cursor out of the virtual screen, results in positioning it on the last line and/or column accessible. Position (0, 0) is the upper-left corner of the display.

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 Type**

BIOS

Returns the cursor position for the display page represented by register BH. This function applies to all video modes, although the cursor is not visible in graphics modes. A page does not need to be currently active for the function to perform properly. The cursor can be moved off the screen. Positioning the cursor off the screen causes it to disappear. Position (0, 0) is the upper-left corner of the display. The cursor type is returned in register CX.

On Entry:
    AH = 03h
    BH = Display page number (zero-based)

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

Makes the page represented by register AL the active 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 through 7 may be selected for modes 0 through 7 and 0x0D. Pages 0 through 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.

On Entry:
    AH = 05h
    BH = Display page number (zero-based)

On Return:
    None

INT 10h, Function AH = 06h; **Scroll Window Up (active page)**

BIOS

Scrolls the active page *up*, by the number of lines indicated by register AL. If register AL = 0, the entire window is blanked. Information outside of the scroll window, as indicated by the values in registers CH, CL, DH, and DL, is not altered.

This function 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 graphics modes, the position is based on an 80x25 display.

On Entry:
    AH = 06h
    AL = Number of lines to scroll
    BH = Attribute for blank lines
    CX = Upper-left corner
    CH = Line
    CL = Column
    DX = Lower-right corner
        CH = Line
        CL = Column

On Return:
    None

INT 10h, Function AH = 07h; **Scroll Window Down (active page)**

BIOS

Scrolls the active page *down* by the number of lines indicated by AL. If AL = 0, the entire window is blanked. Information outside of the scroll window, as indicated by the values in registers CH, CL, DH, and DL, is not altered. This function 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 graphics modes, the position is based on an 80x25 display.

On Entry:
    AH = 07h
    AL = Number of lines to scroll
    BH = Attribute for blank lines
    CX = Upper-left corner
        CH = Line
        CL = Column
    DX = Lower-right corner
        CH = Line
        CL = Column

On Return:
    None

INT 10h, Function AH = 08h; **Read Character and Attribute at Current Cursor Position**

BIOS

Reads the character and the attribute at the current cursor position, for the selected page. This function works in all display modes. The page does not need to be active for the function to operate correctly.

On Entry:
    AH = 08h
    BH = Display page number

On Return:
    AH = Attribute of display character
    AL = Value of character at the cursor position

" **NOTE:**   *The above function is only supported for page zero (0).*

INT 10h, Function AH = 09h; **Write Character and Attribute at Current Cursor Position**

BIOS

Writes the character and the attribute at the current cursor position, for the selected page. If the number of characters exceeds the end of a line, they 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 the special characters are displayed. Control characters do not alter the cursor position. The cursor is not advanced. This function works in all display modes. The page does not need to be active for the function to operate correctly.

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 (no attribute) at Current Cursor Position**

BIOS

Writes the character (only) at the current cursor position for the selected page. If the number of characters exceeds the end of a line, they 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 the special characters are displayed. Control characters do not alter the cursor position. The cursor is not advanced. This function works in all display modes. The page does not need to be active for the function to operate correctly.

On Entry:
     AH = 0Ah
     AL = Value of the character
     BH = Display page number
     CX = Count of characters to write

On Return:
     None

INT 10h, **Function AH = 0Ch**; **Write Graphics Dot**

BIOS

Writes a specified pixel to video memory at the location represented by registers CX and DX. The value in register AL is the color value for graphic modes. If bit 7 of register AL is set, the color value is XORed with the current screen location contents. The range of pixel values depends on the current video mode.

On Entry:
     AH = 0Ch
     AL = Dot value
         0 = off
         1 = on
     BH = Page number
     CX = Dot column number
         0–320 = Modes 4–5
         0–640 = Mode 6
     DX = Dot row number

On Return:
     AL = Color value

INT 10h, Function AH = 0Dh; **Read Graphics Dot**

BIOS

Reads the picture element value at the location specified by registers CX and DX. The value returned in register 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

On Return:
> AL = Dot value
>> 0 = off
>> 1 = on

INT 10h, Function AH = 0Eh; **Teletype Character Write**

BIOS

Writes characters to the display, as a serial terminal, in all display modes. Carriage return homes the cursor on the current line. Line feed advances the cursor to the next line. Tab advances the cursor to the next tab stop (every 8 spaces). Backspace backs the cursor, but only to the beginning of the current line. Bell beeps the buzzer. All other characters are displayable characters. The cursor advances after a character is written. If the end of a line is reached, the cursor advances to the next line. If the position of thelast line of the display is written to, the display scrolls up one line.

On Entry:
> AH = 0Eh
> AL = Character to write
> BH = Page number
> BL = Foreground color in graphics mode

On Return:
> None

INT 10h, Function AH = 0Fh; **Read Current Video Mode**

BIOS

Returns information stored in locations maintained by Function AH=00h and Function AH=05h.

On Entry:
> AH = 0Fh

On Return:
> AH = Number of columns on screen
> AL = Current video mode
> BH = Current active display page

INT 10h, Function AX = 1200h; **Turn Display On**
    Subfunction BL = 36h: **Alternate Function**

BIOS

Turns the video display on.

On Entry:
    AH = 12h
    AL = 00h
    BL = 36h

On Return:
    AL = 12h (function supported)

INT 10h, Function AX = 1201h; **Turn Display Off**
    Subfunction BL = 36h: **Alternate Function**

BIOS

Turns the video display off.

On Entry:
    AH = 12h
    AL = 01h
    BL = 36h

On Return:
    AL = 12h (function supported)

INT 10h, Function AX = 1200h; **Set Viewport Position to Lower-Left Corner**
    Subfunction BL = F6h: **Alternate Function**

BIOS

Positions the viewport to the lower-left corner of the full VGA screen.

On Entry:
    AH = 12h
    AL = 00h
    BH = 00h
    BL = 0F6h

On Return:
    None

INT 10h, Function AX = 1201h; **Set Viewport Position to Upper-Left Corner**
    Subfunction BL = F6h: **Alternate Function**

BIOS

Positions the viewport to the upper-left corner of the full VGA screen.

On Entry:
    AH = 12h
    AL = 01h
    BH = 00h
    BL = 0F6h

On Return:
    None

INT 10h, Function AX = 1202h; **Set Viewport to Follow Cursor**
    Subfunction BL = F6h: **Alternate Function**

BIOS

Sets the viewport to follow the cursor.

On Entry:
    AH = 12h
    AL = 02h
    BH = 00h
    BL = 0F6h
    CH = Top line   ; Window in which cursor can move
    CL = Left column   ; without the viewport moving.
    DH = Bottom line   ; Once the cursor moves outside,
    DL = Right column   ; the viewport follows.

On Return:
    None

INT 10h, Function AX = 1203h; **Reset Annunciator Line to Current Keyboard Settings**
    Subfunction BL = F6h: **Alternate Function**

BIOS

Resets the annunciator line to the current keyboard settings.

On Entry:
    AH = 12h
    AL = 03h
    BH = 00h
    BL = 0F6h

On Return:
    None

Note: This was implemented in flash version 1.14, in the event the annunciator display becomes altered.

INT 10h, Function AX = 1204h; **Enable/Disable Annunciator Panning**
    Subfunction BL = F6h;   **Alternate Function**

BIOS

Enables and disables the annunciator panning.

On Entry:
    AH = 12h
    AL = 04h
    BH = 00h
    BH = 0F6h
    CL = Switch value
        0 (disable)
        1 (enable)

On Return:
    None

INT 10h, Function AX = 1205h; **Get Current Viewport Location**
    Subfunction BL = F6h: **Alternate Function**

BIOS

Returns the location of the viewport (the visible window).

On Entry:
    AH = 12h
    AL = 05h
    BH = 00h
    BL = 0F6h

On Return:
    CH = Line
    CL = Column

INT 10h, Function AX = 1206h; **Lock Viewport**
    Subfunction BL = F6h: **Alternate Function**

BIOS

Locks the viewport into the current position and size.

On Entry:
    AH = 12h
    AL = 06h
    BH = 00h
    BL = 0F6h

On Return:
    None

INT 10h, Function AX = 1207h; **Unlock Viewport**
    Subfunction BL = F6h: **Alternate Function**

BIOS

Unlocks the viewport so the position and size can be changed.

On Entry:
    AH = 12h
    AL = 07h
    BH = 00h
    BL = 0F6h

On Return:
    None

INT 10h, Function AH = 12h; **Write BIOS Annunciator Values (with attribute)**
Subfunction BL = F8h: **Alternate Function**

BIOS
RFBIOS
Writes the annunciator number and its bitmap, including the display attribute, to the annunciator.

On Entry:
AH = 12h
BH = 00h
BL = 0F8h
AL = Annunciator number
CXDX = Annunciator bitmap
SI = Attribute
0 = Normal
4 = Blinking

On Return:
None

INT 10h, Function AH = 12h; **Write BIOS Annunciator Values (without attribute)**
Subfunction BL = F9h; **Alternate Function**

BIOS
RFBIOS
Writes the annunciator number and its bitmap to the annunciator, without changing the display attribute.

On Entry:
AH = 12h
BH = 00h
BL = 0F9h
AL = Annunciator number
CXDX = Annunciator bitmap

On Return:
None

INT 10h, Function AH = 12h; **Set Physical Display Size**
Subfunction BL = FAh; **Alternate Function**

BIOS
Changes the number of lines and columns in the viewport and the font size.

On Entry:
AH = 12h
BH = 00h
BL = 0FAh
CH = Number of lines
CL = Number of columns

On Return:
CH = Actual number of lines
CL = Actual number of columns

Note: Refer to *The Standard Display* and *The Wide Display* paragraphs, in the *Communications and Device Support* section of this publication, for details concerning available lines and columns.

INT 10h, Function AH = 12h; **Return Pointer to Current Display Parameter Packet and Font Information**; **Subfunction BL = FBh**; **Alternate Function**

BIOS

Returns a pointer to the current display parameter packet and font information.

On Entry:
    AH = 12h
    BH = 00h
    BL = 0FBh

On Return:
    ES:DI = Address of display parameter packet
    CX = Maximum number of programmable characters
    DS:SI = Address of current font table
    BH = Number of bytes per character in font table

Note: If CX = 0, then DS:SI and BH have no meaning.

INT 10h, Function AX = 12E0h; **Select Chase Mode**
    Subfunction BL = FCh;   **Alternate Function** *(select 3x4x specific display mode)*

BIOS

Selects a 3x4x specific display mode.

On Entry:
    AH = 12h
    AL = 0E0h (display mode = MODE_CHASE)
    BH = 00h
    BL = 0FCh

On Return:
    None

INT 10h, Function AX = 12E1h; **Select Fixed Mode**
    Subfunction BL = FCh;   **Alternate Function** *(select 3x4x specific display mode)*

BIOS

Selects a 3x4x specific display mode.

On Entry:
    AH = 12h
    AL = 0E1h (display mode = MODE_FIXED)
    BH = 00h
    BL = 0FCh
    DH = Window coordinate (Line)
    DL = Window coordinate (Column)

On Return:
    None

INT 10h, Function AX = 12E2h; **Select Setparms Mode**
   Subfunction BL = FCh;   **Alternate Function** *(select 3x4x specific display mode)*

BIOS

Selects a 3x4x specific display mode.

On Entry:
   AH = 12h
   AL = 0E2h (display mode = MODE_SETPARMS)
   BH = 00h
   BL = 0FCh
   ES:DI = Pointer to display parameter packet

On Return:
   None

INT 10h, Function AX = 12E3h; **Select Graphics Mode**
   Subfunction BL = FCh;   **Alternate Function** *(select 3x4x specific display mode)*

BIOS

Selects a 3x4x specific display mode.

On Entry:
   AH = 12h
   AL = 0E3h (display mode = MODE_GRAFIX)
   BH = 00h
   BL = 0FCh

On Return:
   None

INT 10h, Function AH = 12h; **Return Physical Display Size**
   Subfunction BL = FDh;   **Alternate Function**

BIOS
RFBIOS

Returns the physical display size set by the "Set Physical Display Size" function.

On Entry:
   AH = 12h
   BH = 00h
   BL = 0FDh

On Return:
   AH = Number of columns
   AL = Number of lines
   ES:DI = Address of table of the supported sizes

INT 10h, Function AX = 1200h; Get Contrast *(temperature compensated)*
    Subfunction BL = FEh;   **Alternate Function**

BIOS

This is a *Contrast Control* function.

On Entry:
    AH = 12h
        AL = 00h   (Get contrast function)
        BH = 00h
        BL = 0FEh

On Return:
    BH = Contrast value
        (range: 0 = lightest to 255 = darkest)

INT 10h, Function AX = 1201h; Set Contrast *(temperature compensated)*
    Subfunction BL = FEh;   **Alternate Function**

BIOS

This is a *Contrast Control* function.

On Entry:
    AH = 12h
    AL = 01h   (Set contrast function)
    BL = 0FEh
    BH = Contrast value
        (range: 0 = lightest to 255 = darkest)

On Return:
    None

INT 10h, Function AX = 1202h; Get Contrast *(absolute, no compensation)*
    Subfunction BL = FEh;   **Alternate Function**

BIOS

This is a *Contrast Control* function.

On Entry:
    AH = 12h
    AL = 02h   (Get contrast function)
    BH = 00h
    BL = 0FEh

On Return:
    BH = Contrast value
        (range: 0 = lightest to 255 = darkest)

INT 10h, Function AX = 1203h; **Set Contrast** *(absolute, no compensation)*
    Subfunction BL = FEh;   **Alternate Function**

BIOS

This is a *Contrast Control* function.

On Entry:
    AH = 12h
    AL = 03h   (Set Contrast function)
    BL = 0FEh
    BH = Contrast value
       (range: 0 = lightest to 255 = darkest)

On Return:
    None

INT 10h, Function AX = 1205h; **Compensate Contrast**
    Subfunction BL = FEh;   **Alternate Function**

BIOS

This is a *Contrast Control* function.

On Entry:
    AH = 12h
    AL = 05h   (Compensate Contrast function)
    BH = 00h
    BL = 0FEh

On Return:
    None

INT 10h, Function AX = 1203h; **Turn Contrast Heater Off**
    Subfunction BL = FEh;   **Alternate Function**

BIOS

This is a *Contrast Control* function.

On Entry:
    AH = 12h
    AL = 06h   (Turn Contrast Heater Off function)
    BL = 0FEh
    BH = Off time for heater

On Return:
    None

" NOTE: *The function, described above, is NOT currently supported.*

INT 10h, Function AX = 1207h; **Turn Contrast Heater On**
          Subfunction BL = FEh;   **Alternate Function**

BIOS

This is a *Contrast Control* function.

On Entry:
   AH = 12h
   AL = 07h   (Turn Contrast Heater On function)
   BL = 0FEh
   BH = On time for heater

On Return:
   None

" **NOTE:**   *The function, described above, is NOT currently supported.*

INT 10h, Function AX = 1200h; **Turn Backlight On**
     Subfunction BL = FFh;   **Alternate Function**

BIOS
RFBIOS

This is a *Backlight Control* function.

On Entry:
   AH = 12h
   AL = 00h   (Turn Backlight On function)
   BH = 00h
   BL = 0FFh

On Return:
   None

INT 10h, Function AX = 1201h; **Turn Backlight Off**
     Subfunction BL = FFh;   **Alternate Function**

BIOS
RFBIOS

This is a *Backlight Control* function.

On Entry:
   AH = 12h
   AL = 01h   (Turn Backlight Off function)
   BH = 00h
   BL = 0FFh

On Return:
   None

INT 10h, Function AX = 1202h; **Toggle Backlight**
      Subfunction BL = FFh;   **Alternate Function**

BIOS
RFBIOS

This is a *Backlight Control* function.

On Entry:
    AH = 12h
    AL = 02h   (Toggle Backlight function)
    BH = 00h
    BL = 0FFh

On Return:
    None

INT 10h, Function AX = 1203h; **Read Backlight Intensity**
      Subfunction BL = FFh;   **Alternate Function**

BIOS
RFBIOS

This *Backlight Control* function reads the current backlight intensity.

On Entry:
    AH = 12h
    AL = 03h   (Read Backlight Intensity function)
    BH = 00h
    BL = 0FFh

On Return:
    BH = Intensity
       (range: 0 = lightest to 255 = darkest)

INT 10h, Function AX = 1204h; **Set Backlight Intensity**
      Subfunction BL = FFh;   **Alternate Function**

BIOS
RFBIOS

This *Backlight Control* function sets backlight intensity to specified value.

On Entry:
    AH = 12h
    AL = 04h   (Write Backlight Intensity function)
    BL = 0FFh
    BH = Intensity
       (range: 0 = lightest to 255 = darkest)

On Return:
    None

INT 10h, Function AX = 1205h; **Set Backlight Timer**
      Subfunction BL = FFh;   **Alternate Function**

BIOS
RFBIOS

This *Backlight Control* function sets the backlight timer to specified value.

On Entry:
    AH = 12h
    AL = 05h   (Set Backlight Timer function)
    BL = 0FFh
    BH = Timer value in seconds

On Return:
    None

INT 10h, Function AX = 1206h; **Refresh Backlight**
      Subfunction BL = FFh;   **Alternate Function**

BIOS
RFBIOS

This Backlight Control function refreshes the backlight (due to a keypress).

On Entry:
    AH = 12h
    AL = 06h   (Refresh Backlight function)
    BH = 00h
    BL = 0FFh

On Return:
    None

INT 10h, Function AX = 1207h; **Read Backlight Timer**
      Subfunction BL = FFh;   **Alternate Function**

BIOS
RFBIOS

This *Backlight Control* function reads the backlight timer.

On Entry:
    AH = 12h
    AL = 07h   (Read Backlight Timer function)
    BL = 0FFh

On Return:
    BH = Timer value in seconds

INT 10h, Function AX = 1400h; Load User Font

BIOS

Loads and stores user fonts, for use in place of the default fonts.

On Entry:
    AH = 14h
    AL = 00h
    BL = Request type
        00h   Load main font
        0FFh   Load 16-line display font
    BH = Number of bytes per character
    CX = Number of characters to store
    DX = Character offset into font of first character to store
    ES:DI = Address of start of user font table (bit map)

The font data is stored byte-by-byte (as a bit map), starting with the top row in the first byte, ending with the bottom byte in the last byte. For example, if you are replacing a single character, "1", in a particular font:

-   The data for the replacement character needs to be stored as shown in the following example, where ES:DI + 0 (below) represents the address pointed to by ES:DI, and the binary data listed represents the data stored at that location.
-   BH=8 because there are eight bytes to store for this character.
-   CX=1 since you are only storing one character.
-   DX=31h which is the offset to that character in the standard font table.

Location	Binary Data
ES:DI + 0	0001 0000
ES:DI + 1	0001 0000
ES:DI + 2	0001 0000
ES:DI + 3	0001 0000
ES:DI + 4	0001 0000
ES:DI + 5	0001 0000
ES:DI + 6	0001 0000
ES:DI + 7	0000 0000

On Return:
    None

Note: The largest bitmap font that can be supported is 8 kilobytes.

INT 10h, **Function AH = F0h**; **Draw Pixel Map**

BIOS
RFBIOS

This function specifies pixel picture size, position, points to pixel map, and draws the map.

On Entry:
> AH = 0F0h
> BX = Pixel picture height
> CX = Pixel picture width
> DX = Pixel picture top line
> DI = Pixel picture left column
> DS:SI = Points to pixel map

On Return:
> None

INT 10h, **Function AH = FFh**; **Refresh the Screen**

BIOS

Refreshes the display screen.

On Entry:
> AH = 0FFh

On Return:
> None

INT 13h, Disk Services

INT 13h, Function AH = 00h; Reset Disk System

BIOS

Resets the specified disk drive.

On Entry:
AH = 00h
DL = Drive number (if bit 7 is set, both hard disks and floppy disks reset)

On Return:
Carry = Status flag
AH = Status of operation
if successful:
    Carry = 0
    AH =00h
if NOT successful:
    Carry = 1
    AH =error code

Error Codes: Refer to *Table 5-2, Disk Status Error Codes*, page 5-34, for a list of the
possible error codes.

Note: This function forces controller to recalibrate drive heads (seek to track 0)

INT 13h, Function AH = 01h; Get Status of Last Operation

BIOS

Gets the status of the previous operation on the specified disk drive.

On Entry:
AH = 01h
DL = Drive number (bit 7 set for hard disk)

On Return:
Carry = Status flag
AH = Status of operation
if successful:
    Carry = 0
    AH =00h
if NOT successful:
    Carry = 1
    AH =error code

Error Codes: Refer to *Table 5-2, Disk Status Error Codes*, page 5-34, for a list of the
possible error codes.

Note: Some BIOS systems return the status in register AL.

INT 13h, Function AH = 02h; Read Sectors into Memory

BIOS

Reads the specified sectors, from the specified disk drive, into memory.

On Entry:

AH = 02h

AL = Number of sectors to read (Must be nonzero)

CH = Low eight bits of cylinder number

CL = Sector number 1–63 (bits 0–5)

   high two bits of cylinder (bits 6–7, hard disk only)

DH = Head number

DL = Drive number (bit 7 set for hard disk)

On Return:

Carry = Status flag

AH = Status of operation

AL = Number of sectors transferred

   (if register AH returns error code 11h, then register AL = burst length)

if successful:

   Carry = 0

   AH =00h

if NOT successful:

   Carry = 1

   AH =error code

Error Codes: Refer to *Table 5-2, Disk Status Error Codes*, page 5-34, for a list of the possible error codes.

Note 1: Errors on a floppy may be due to the the motor failing to spin up quickly enough. Retry the read (or write) at least three times, resetting the disk (with AH = 00h) between attempts.

Note 2: Many BIOS systems use only the low four bits of DH (head number) since the WD-1003 controller (the standard AT controller and the controller that IDE emulates) only supports 16 heads.

See Also: Functions AH=03h and AH=0Ah.

INT 13h, Function AH = 03h; **Write Disk Sectors**

BIOS

Writes from memory to the specified sectors on the specified disk drive.

On Entry:

AH = 03h

AL = Number of sectors to write (must be nonzero)

CH = Low eight bits of cylinder number

CL = Sector number 1–63 (bits 0–5)

   high two bits of cylinder (bits 6–7, hard disk only)

DH = Head number

DL = Drive number (bit 7 set for hard disk)

On Return:

Carry = Status flag

AH = Status of operation

AL = Number of sectors transferred

   (if register AH returns error code 11h, then register AL = burst length)

if successful:

   Carry = 0

   AH =00h

if NOT successful:

   Carry = 1

   AH =error code

Error Codes: Refer to *Table 5-2, Disk Status Error Codes*, page 5-34, for a list of the possible error codes.

Note 1: Errors on a floppy may be due to the the motor failing to spin up quickly enough. Retry the read (or write) at least three times, resetting the disk (with AH = 00h) between attempts.

Note 2: Many BIOS systems use only the low four bits of DH (head number) since the WD-1003 controller (the standard AT controller and the controller that IDE emulates) only supports 16 heads.

See Also: Functions AH=02h and AH=0Bh.

INT 13h, Function AH = 04h;Verify Disk Sectors

BIOS

Verifies the specified sectors on the specified disk drive.

On Entry:
   AH = 04h
   AL = Number of sectors to verify (must be nonzero)
   CH = Low eight bits of cylinder number
   CL = Sector number 1–63 (bits 0–5)
      high two bits of cylinder (bits 6–7, hard disk only)
   DH = Head number
   DL = Drive number (bit 7 set for hard disk)

On Return:
   Carry = Status flag
   AH = Status of operation
   AL = Number of sectors verified
   if successful:
      Carry = 0
      AH =00h
   if NOT successful:
      Carry = 1
      AH =error code

Error Codes: Refer to *Table 5-2, Disk Status Error Codes*, page 5-34, for a list of the possible error codes.

Note 1: Errors on a floppy may be due to the the motor failing to spin up quickly enough. Retry the read (or write) at least three times, resetting the disk (with AH = 00h) between attempts.

Note 2: Many BIOS systems use only the low four bits of DH (head number) since the WD-1003 controller (the standard AT controller and the controller that IDE emulates) only supports 16 heads.

Note 3: This function does not compare the disk with memory. It checks whether the sector's stored CRC matches the data's CRC.

See Also: Function AH=02h.

INT 13h, Function AH = 05h; **Format Track** *(floppy disk)*

BIOS

Formats the specified track, for the specified sectors on the specified disk drive.

On Entry:

    AH = 05h
    AL = Number of sectors to format (must be nonzero)
    CH = Track number
    DH = Head number
    DL = Drive number (bit 7 set for hard disk)

On Return:

    Carry = Status flag
    AH = Status of operation
    if successful:
       Carry = 0
       AH =00h
    if NOT successful:
       Carry = 1
       AH =error code

Error Codes: Refer to *Table 5-2, Disk Status Error Codes*, page 5-34, for a list of the possible error codes.

Note: On AT or higher, call AH = 17h first. The number of sectors per track is read from the diskette parameter table pointed to by INT 1E.

See Also: Functions AH=05h "FIXED", AH=17h, AH=18h, and INT 1E.

Format of address field buffer entry (one per sector in track):

Offset Size		Description
00h	BYTE	Track number
01h	BYTE	Head number (zero-based)
02h	BYTE	Sector number
03h	BYTE	Sector size:
		00h = 128 bytes
		01h = 256 bytes
		02h = 512 bytes
		03h = 1024 bytes

INT 13h, Function AH = 08h; Get Drive Parameters

BIOS

Returns the parameters for the specified disk drive.

On Entry:
    AH = 08h
    DL = Drive number (bit 7 set for hard disk)

On Return:
    AH = Status of operation
    Carry = Status flag
        if successful:
            AH = 00h
            Carry = 0
            BL = Drive type *(AT/PS2 floppies only)*
                01h = 360K
                02h = 1.2M
                03h = 720K
                04h = 1.44M
                All others = unknown
            CH   = Cylinder number, bits 0–7
            CL bits 6, 7 = Cylinder number, bits 8, 9
               bits 0–5 = Maximum sector number
            DH bits 6, 7 = Cylinder number, bits 10, 11
               bits 0–5 = Maximum head number (range: 0 to 63)
            DL = Number of drives, 1 or 2
        if NOT successful (but at least one drive is active)
            AH = 07h
            carry = 1
            AL = CX = DX = 0
        if NOT successful (no active drives, or drive number out of range)
            AH = 01h
            Carry = 1
            AL = BX = CX = DX = ES = 0

" NOTE: *The maximum cylinder number must be increased by one to get the total number of cylinders, which does not include the diagnostic cylinder.*

" NOTE: *The maximum head number must be increased by one to get the total number of heads on the drive (1 to 64).*

Error Codes: Refer to *Table 5-2, Disk Status Error Codes*, page 5-34, for a list of the possible error codes.

See Also: Functions AH=15h, INT 1Eh, INT 41h.

" NOTE: This function may return successful even if specified drive is greater than the number of attached drives of that type (floppy or hard). Check register DL to ensure validity.

INT 13h, Function AH = 15h; Get Disk Type

BIOS

Gets the disk type for the specified disk drive.

On Entry:
AH = 15h
DL = Drive number (bit 7 set for hard disk)

On Return:
if successful:
Carry = 0
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 512-byte sectors (If AH = 03h)
if NOT successful
Carry = 1
AH = Status of operation

Error Codes: Refer to *Table 5-2, Disk Status Error Codes*, page 5-34, for a list of the possible error codes.

See Also: Functions AH=08h, AH=16h, AH=17h.

INT 13h, Function AH = 16h; Detect Disk Change

BIOS

Detects if a change has occurred with floppy disk in the specified disk drive.

On Entry:
AH = 16h
DL = Drive number (bit 7 set for hard disk)

On Return:
if successful:
Carry = 0 (if change line not active)
AH = 00h (disk not changed)
if NOT successful:
Carry = 1 (if change line is active)
AH = Status
06h = change line active or not supported *(See Note, below)*
80h = Drive not ready or not present

Note: Call AH=15h first, to determine whether the drive supports a change line.

See Also: Functions AH=15h.

INT 13h, Function AH = 17h; **Set Disk Type for Format** *(floppy disk)*

BIOS

Sets the specified format type for the floppy disk in the specified disk drive.

On Entry:
> AH = 17h
> AL = Format type
>> 01h = 320KB / 360KB disk in 360KB drive
>> 02h = 320KB / 360KB disk in 1.2MB drive
>> 03h = 1.2MB in 1.2MB drive
>> 04h = 720KB disk in 720KB drive
>
> DL = Drive number (bit 7 set for hard disk)

On Return:
> if successful:
>> Carry = 0
>> AH = 00h
>
> if NOT successful:
>> Carry = 1
>> AH = Error code

Error Codes: Refer to *Table 5-2, Disk Status Error Codes*, page 5-34, for a list of the possible error codes.

See Also: Functions AH=15h, AH=18h.

" **NOTE:** *This function does not handle 1.44MB drives. Use AH=18h, instead.*

INT 13h, Function AH = 18h; **Set Media Type for Format**

BIOS

Sets the specified media type for the specified disk drive.

On Entry:
> AH = 18h
> DL = Drive number (bit 7 set for hard disk)
> CH = Lower 8 bits of highest cylinder number (one cylinder)
> CL = Sectors per track (bits 0–5),
>> top 2 bits (6,7) of highest cylinder number

On Return:
> AH = Status
>> 00h = Requested combination supported
>> 01h = Function not available
>> 0Ch = Not supported or drive type unknown
>> 80h = There is no disk in the drive

See Also: Functions AH=05h, AH=07h, AH=17h.

Disk Services Return Status

Any of the following error codes could be returned for any of the disk services in this set of services (INT 13):

Table 5-2
Disk Status Error Codes

Status	Description
00h	Successful completion
01h	Invalid function in AH or invalid parameter
02h	Address mark not found
03h	Disk write protected (floppy)
04h	Sector not found
05h	Reset failed (hard disk)
06h	Disk changed (floppy)
07h	Drive parameter activity failed (hard disk)
08h	DMA overrun
09h	Attempted DMA across 64K boundary
0Ah	Bad sector detected (hard disk)
0Bh	Bad track detected (hard disk)
0Ch	Unsupported track or invalid media
0Dh	Invalid number of sectors on format (hard disk)
0Eh	Control data address mark detected (hard disk)
0Fh	DMA arbitration level out of range (hard disk)
10h	Uncorrectable CRC or ECC error on read
11h	Data ECC corrected (hard disk)
20h	Controller failure
40h	Seek failed
80h	Timeout not ready
AAh	Drive not ready (hard disk)
BBh	Undefined error (hard disk)
CCh	Write fault (hard disk)
E0h	Status register error (hard disk)
FFh	Sense operation failed (hard disk)

INT 13h, NORAND Extensions

INT 13h, Function AH = DAh; **Disable Access to RAM Drive Memory**

BIOS

Disables all access to the RAM drive memory, by reprogramming the memory
controller to disable access to the RAM associated with the RAM drive.

On Entry:
AH = 0DAh
DL = 00h

On Return:
AH = 00h

INT 13h, Function AH = EAh; **Enable Access to RAM Drive Memory**

BIOS

Enables all access to the RAM drive memory, by reprogramming the memory
controller to enable access to the RAM associated with the RAM drive.

On Entry:
AH = 0EAh
DL = 00h

On Return:
AH = 00h

INT 13h, Function AH = ECh; **Enable Checksum of RAM Drive Integrity Memory**

BIOS

Enables the checksum of the RAM drive memory.

On Entry:
AH = 0ECh
ES:SI = Pointer to Global Descriptor Table (GDT)
offset 00h = uninitialized, null descriptor
offset 08h = uninitialized, will be made into GDT descriptor
offset 10h = uninitialized, descriptor for source of move
offset 18h = uninitialized, descriptor for destination of move
offset 20h = uninitialized, used by BIOS
offset 28h = uninitialized, made into SS descriptor

On Return:
if successful:
Carry = 0
AH = 00h
if NOT successful:
Carry = 1
AH = Status of operation
01h = Parity error
02h = Interrupt error
03h = Address line 20 gating failed

INT 14h, Communications Services

INT 14h, Function AH = 00h; Initialize Communications Port

BIOS

Programs the Baud Rate Register and Byte Format Register for the selected
port, and returns Line Status Register value and Modem Status Register value.
It enables UART power on the intercept, then passes the call on to the BIOS
interrupt 14h entry point for initialization.

On Entry:
>   AH = 00h
>   DX = Communications port number
>   AL = Initialization parameters
>>   Bits 0–1: Word Length
>>>   10b = 7 bits
>>>   11b = 8 bits
>>   Bit 2: Stop Bits
>>>   00b = One
>>>   01b = Two
>>   Bits 3–4: Parity
>>>   00b = None
>>>   01b = Odd
>>>   10b = None
>>>   11b = Even
>>   Bits 5–7: Baud Rate
>>>   000b = 110
>>>   001b = 150
>>>   010b = 300
>>>   011b = 600
>>>   100b = 1200
>>>   101b = 2400
>>>   110b = 4800
>>>   111b = 9600

On Return:
>   AH = Line status register
>   AL = Modem status register

Error Codes: Refer to *Table 5-3, Line Status Register Values* (page 5-40),
>   and *Table 5-4, Modem Status Register Values* (page 5-40), for a list of the bit values.

INT 14h, **Function AH = 01h**; **Transmit a Character**

BIOS

Transmits the character through the specified communications port, which raises DTR and RTS. DSR and CTS are expected within a timeout period specified by an array at 40:7C. Once DSR and CTS are received, the character to be sent is placed in the transmit buffer register.

On Entry:
    AH = 01h
    DX = Communications port number
    AL = Character to be sent

On Return:
    AH = Line status register
    AL = Modem status register

Error Codes: Refer to *Table 5-3, Line Status Register Values* (page 5-40), and *Table 5-4, Modem Status Register Values* (page 5-40), for a list of the bit values.

INT 14h, **Function AH = 02h**; **Receive a Character**

BIOS

Receives a character through the specified communications port, which raises DTR and drops RTS. DSR is expected within a timeout period specified by an array at 40:7C. Once DSR is received, a character is expected in the receive buffer within the timeout period. If a timeout occurs, the timeout bit is put into register AH. If a character is received before the timeout, it is returned in register AL.

On Entry:
    AH = 02h
    DX = Communications port number

On Return:
    AH = Line status register
    AL = Character received

Error Codes: Refer to *Table 5-3, Line Status Register Values* (page 5-40), for a list of the bit values.

INT 14h, **Function AH = 03h**; **Read Communications Port Status**

BIOS

Returns current line and modem status for the specified communications port.

On Entry:
    AH = 03h
    DX = Communications port number

On Return:
    AH = Line status register
    AL = Modem status register

Error Codes: Refer to *Table 5-3, Line Status Register Values* (page 5-40), and *Table 5-4, Modem Status Register Values* (page 5-40), for a list of the bit values.

INT 14h, Function AH = 04h; **Extended Initialize**

BIOS

Programs the Baud Rate Register and Byte Format Register for the selected port, and returns values of the Line Status Register and Modem Status Register. It enables UART power on the intercept, then passes the call on to the BIOS interrupt 14h entry point for initialization.

On Entry:

AH = 04h
DX = Communications port number
   (available port numbers: 0, 1, 2)
BH = Parity
   00h = None
   01h = Odd
   02h = Even
BL = Stop Bits
   00h = One
   01h = Two
CH = Word Length
   02h = 7 bits
   03h = 8 bits
CL = Baud Rate *(this parameter is used as an index into the following table)*
   00h = 110
   01h = 150
   02h = 300
   03h = 600
   04h = 1200
   05h = 2400
   06h = 4800
   07h = 9600
   08h = 19,200
   09h = 38,400
   0Ah = 57,600
   0Bh = 115,200

On Return:

AH = Line status register
AL = Modem status register

Error Codes: Refer to *Table 5-3, Line Status Register Values* (page 5-40), and *Table 5-4, Modem Status Register Values* (page 5-40), for a list of the bit values.

" **NOTE:**   *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 AX = 0500h; **Read Modem Control Port**

BIOS

Reads information from the modem control register.

On Entry:

AH = 05h

AL = 00h   (read modem control register)

DX = Communications port number
        (available port numbers: 0, 1, 2)

On Return:

BL = Modem control information

   bit 7: Ignore modem control signals, COM1 only

   bit 1: Request To Send (RTS)

   bit 0: Data Terminal Ready (DTR)

AH = Line status register

AL = Modem status register

Error Codes: Refer to *Table 5-3, Line Status Register Values* (page 5-40), and *Table 5-4, Modem Status Register Values* (page 5-40), for a list of the bit values.

INT 14h, Function AX = 0501h; **Write Modem Control Port**

BIOS

Writes the specified information to the modem control register.

On Entry:

AH = 05h

AL = 01h (write to modem control register)

DX = Communications port number
        (available port numbers: 0, 1, 2)

BL = Modem control register value

   bit 7: Ignore modem control signals, COM1 only

   bit 1: Request To Send (RTS)

   bit 0: Data Terminal Ready (DTR)

On Return:

AH = Line status register

AL = Modem status register

Error Codes: Refer to *Table 5-3, Line Status Register Values* (page 5-40), and *Table 5-4, Modem Status Register Values* (page 5-40), for a list of the bit values.

" **NOTE:**   *This is a nonstandard PC function. It may not be available in every PC BIOS.*
              Use caution where portability is a concern.

Bit Values for Status Registers

Table 5-3
Line Status Register Values

Bit	Description
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

Table 5-4
Modem Status Register Values

Bit	Description (Signal)
7	Carrier Detect (CD)
6	Ring Indicator (RI)
5	Data Set Ready (DSR)
4	Clear To Send (CTR)
3	Delta Carrier Detect (DCD)
2	Delta Trailing Edge Ring Indicator
1	Delta Data Set Ready
0	Delta Clear To Send

INT 15h, System BIOS Services

INT 15h, Function AX = 010Ch; Scanner Task Switch

SCANBIOS

Switches control between the application thread and the scanner handler thread. The scanner handler thread is created and deleted by the SCANBIOS *Scanner On* and *Scanner Off* functions.

On Entry:
> AX = 010Ch
> CX = 00h
> DX = 00h

On Return:
> None

INT 15h, Function AX = 01FAh: Beep the Buzzer

BIOS

Makes sounding device emit a beep.

On Entry:
> AH = 01h
> AL = FAh
> BL = Volume level of sound
>> (range: 0 = quiet to 255 = loud)
> CX = Duration of sound, in milliseconds (approximately)
>> DX = Frequency of sound, in Hz

On Return:
> None

" **NOTE:**   *This interrupt is a PC-like system service, but not necessarily PC compatible.*

INT 15h, Function AX = 2403h: Query A20 Gate Support

BIOS

This function ... TBD.

On Entry:
> AH = 24h
> AL = 03h

On Return:
> if successful
>> Carry = 0
>> AH = 00h
>> BX = Status of A20 gate support
>>> bit 0:   supported on keyboard controller
>>> bit 1:   supported with bit 1 of I/O port 92h
>>> bit 15:   additional data is available (location not yet defined)
> if NOT successful:
>> Carry = 1
>> AH = Status
>>> 01h = Keyboard controller is in secure mode
>>> 86h = Function not supported

INT 15h, Function AX = 42FFh: Request Low Power Sleep

MISCTSR

Puts in a requests for the system to go into a low power sleep. When the timeout expires, or when a system event (keyboard, scanner, radio receive, etc.) occurs, the actual time the system was asleep is returned.

On Entry:
> AH = 42h
> AL = FFh
> DX:CX = Number of milliseconds you want the processor to sleep

On Return:
> DX:CX = Actual number of milliseconds the processor was asleep

---

INT 15h, Function AH = 4Fh: Keyboard Intercept

BIOS

Called by INT 09h each time a key has been pressed. You can use it to create alternate keyboard layouts. The intercept routine can substitute or discard scan codes. The default BIOS routine returns the input scan code (unmodified) and carry (set).

On Entry:
> AH = 4Fh
> AL = Scan code

On Return:
> AL = Scan code
> Carry = Status
>> 0 = Scan code is processed. Do not buffer scan code
>> 1 = Scan code is *not* processed. Buffer the scan code

Note: Your application should call INT 15h, function C0h to determine whether this feature is supported by the BIOS.

" **NOTE:**   *This interrupt is a PC-like system service, but not necessarily PC compatible.*

INT 15h, Function AX = 5300h: APM Installation Check

ELANAPM

Determines whether the system's BIOS supports the APM functionality; and which specification version supported. The APM version number returned indicates the highest level of APM supported by the APM BIOS.

On Entry:
> AH = 53h
> AL = 00h
> BX = 0000h (indicating APM BIOS)

On Return:
> if successful
>> Carry = 0 (APM is supported by BIOS)
>> AH = 01h – APM major version number (in BCD format)
>> AL = 01h – APM minor version number (in BCD format)
>> BH = ASCII "P" character
>> BL = ASCII "M" character
>> CX = APM flags
>>> <u>*Bit*</u> <u>Value/Description</u>
>>> 0 = 1 (16-bit protected mode interface supported *(required in APM 1.2)*
>>> 1 = 1 (32-bit protected mode interface supported *(required in APM 1.2)*
>>> 2 = 1 (*CPU Idle* call slows processor clock speed. (If this bit is set, the APM driver must call the CPU Busy function to ensure the system is restored to normal processing clock speed after calling the CPU Idle function to slow down the processor clock speed).
>>> 2 = 0 (*CPU Idle* call stops the clock. (If this bit is cleared, the APM driver does not need to call the CPU Busy function).
>>> 3 = 1 (APM BIOS Power Management disabled)
>>> 4 = 1 (APM BIOS Power Management disengaged)
>>> Other bits reserved
> if NOT successful:
>> Carry = 1
>> AH = Error code
>>> 09h   Unrecognized device ID
>>> 86h   APM not present

Refer to the *Error Codes* paragraph in the *Power Management* section, in this book).

INT 15h, Function AX = 5301h: APM Real Mode Interface Connect

ELANAPM

Establishes the cooperative interface between the APM driver (caller) and the APM BIOS. This function provides compatibility with the Microsoft APM 1.01 specification. Use of the call is not required for the APM functions to succeed, other than the Disconnect function.

On Entry:
    AH = 53h
    AL = 01h
    BX = 0000h

On Return:
    if successful
        Carry = 0
    if NOT successful:
        Carry = 1
        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

Refer to the *Error Codes* paragraph in the *Power Management* section, in this book).

INT 15h, Function AX = 5304h: APM Interface Disconnect

ELANAPM

Provides compatibility with the Microsoft APM 1.01 specification. Use does not prevent the APM functions from succeeding or failing, except for the Connect function. The function has no effect upon the power management configuration.

On Entry:
    AH = 53h
    AL = 04h
    BX = 0000h

On Return:
    if successful
        Carry = 0
    if NOT successful:
        Carry = 1
        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

Refer to the *Error Codes* paragraph in the *Power Management* section, in this book).

INT 15h, Function AX = 5305h: CPU Idle

ELANAPM

An APM driver or power-aware application uses 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:
>   if successful
>   Carry = 0
>   if NOT successful:
>   Carry = 1
>   AH = Error code
>   03h = Interface not connected
>   0Bh = Interface not engaged

Refer to the *Error Codes* paragraph in the *Power Management* section, in this book).

INT 15h, Function AH = 5306h: CPU Busy

ELANAPM

Informs the APM BIOS that system is busy, preventing the system from entering Idle or Suspend states and resetting the timers associated with these states. 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:
>   if successful
>   Carry = 0
>   if NOT successful:
>   Carry = 1
>   AH = Error code
>   03h   Interface not connected
>   0Bh   Interface not engaged

Refer to the *Error Codes* paragraph in the *Power Management* section, in this book).

INT 15h, Function AX = 5307h: Set Power State

ELANAPM

Sets the system or device specified in the 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 *(not available for 6400 Computer)*
> >
> > 04xxh   RS-232
> >
> > 05xxh   Network adapters
> >
> > 06xxh   PC Card slots *(not available for use on 6400 Computer)*
> >
> > 0Exxh   Devices defined by Intermec
> >
> > where xxh =   Unit number (0-based).
> >
> > > *(Unit number of FFh means all devices in this class)*
> > >
> > > (Refer to the *APM BIOS Functions* paragraph, in the *Power Management* section, for details of these devices defined by Intermec).
>
> CX = Power state
>
> > 0000h   Ready
> >
> > 0001h   Standby
> >
> > 0002h   Suspend
> >
> > 0003h   OFF
> >
> > 0004h   Last Request Processing Notification
> >
> > 0005h   Last Request Rejected *(Supported for Power Device ID 0001h)*

On Return:

> if successful
>
> > Carry = 0
>
> if NOT successful:
>
> > Carry = 1
> >
> > 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

Refer to the *Error Codes* paragraph in the *Power Management* section, in this book).

INT 15h, Function AX = 5308h: Enable/Disable Power Management

---

ELANAPM

Enables or disables automatic System Idle and System Suspend functions.

On Entry:
    AH = 3h
    AL = 8h
    BX = 001h
    CX = Function code
        0000h   Disable power management
        0001h   Enable power management

On Return:
    if successful
        Carry = 0
    if NOT successful:
        Carry = 1
        AH = Error code
            01h   Power management functionality disabled
            03h   Interface not connected
            09h   Unrecognized device ID
            0Ah   Parameter value out or range (function code)

Refer to the *Error Codes* paragraph in the *Power Management* section, in this publication, for additional information about error codes.

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:
    if successful:
        Carry = 0
        BH = AC line status
            00h   Off-line
            01h   On-line
            FFh   Unknown
            All other values reserved
        BL = Battery status
            00h   High
            01h   Low
            02h   Critical
            03h   Charging
            FFh   Unknown
            All other values reserved
        CH = Battery flag
            bit 0 = 1 High
            bit 1 = 1 Low

    bit 2 = 1 Critical

    bit 3 = 1 Charging

    bit 7 = 1 No system battery or powerfail occurred

    All other bits reserved

    FFh = Unknown

    All other values reserved

   CL = Remaining battery life

    0–100 = Percentage of full charge

    FFh = Unknown

    All other values reserved

   DX = Remaining battery life

    Bit 15 Time:

      0 = Time-out in seconds

      1 = Time-out in minutes

    Bits 14–0 Number of seconds or minutes:

      0–7FFFh Valid number of seconds

      0–7FFEh Valid number of minutes

      FFFFh   Unknown

  if NOT successful:

   Carry = 1

   AH = Error code

    09h Unrecognized device ID

Refer to the *Error Codes* paragraph in the *Power Management* section, in this publication, for additional information about error codes.

INT 15h, Function AX = 530Bh: Get PM Event

ELANAPM

This function returns the next pending PM event or indicates whether PM events are pending. Call this function until there are no more pending PM events.

On Entry:

 AH = 53h

 AL = 0Bh

 BX = PM event code

On Return:

 if successful:

  Carry = 0

  BX = PM event code

 if NOT successful:

  Carry = 1

  AH = Error code

   80h No power management events pending

Refer to the *Error Codes* paragraph in the *Power Management* section, in this publication, for additional information about error codes.

INT 15h, Function AX = 530Ch: Get Power State

ELANAPM

Returns the device power state only 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

        04xxh   RS232

        05xxh   Network adapters

        06xxh   PC Card slots *(not available for use on 6400 Computer)*

        0Exxh   Devices defined by Intermec

        where xxh =   Unit number (0-based).

                         (Unit number of FFh means all devices in this class)

                         (Refer to the *APM BIOS Functions* paragraph, in the

                         Power Management section, for details of these devices

                         defined by Intermec).

On Return:

    if successful:

        Carry = 0

        CX = Power state

            0000h   Ready

            0001h   Standby

            0002h   Suspend

            0003h   OFF

    if NOT successful:

        Carry = 1

        AH = Error code

            09h   Unrecognized device ID

Refer to the *Error Codes* paragraph in the *Power Management* section, in this publication, for additional information about error codes.

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 power-manage the device. The only automatic power management that the BIOS does with devices is during Suspend. If a device's power management is disabled, during Suspend the device's power state 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
        04xxh   RS232
        05xxh   Network adapters
        06xxh   PC Card slots *(not available for use on 6400 Computer)*
        0Exxh   Devices defined by Intermec
        where xxh =   Unit number (0-based).
                    (Unit number of FFh means all devices in this class)
                    (Refer to the *APM BIOS Functions* paragraph, in the
                    Power Management section, for details of these devices
                    defined by Intermec).
    CX = Function code (0000h = Disable or 0001h = Enable)

On Return:

    if successful
        Carry = 0
    if NOT successful:
        Carry = 1
        AH = Error code
            09h   Unrecognized device ID

Refer to the *Error Codes* paragraph in the *Power Management* section, in this publication, for additional information about error codes.

INT 15h, Function AX = 5380h: OEM Installation Check
      Subfunction BH = 7Fh

64APMOEM

Verifies if the APM BIOS supports OEM function extensions. This API is a required OEM extension.

On Entry:

    AH = 53h
    AL = 80h
    BH = 7Fh

On Return:

    Carry = Status flag
        0 (success)
        1 (sailure)

**INT 15h, Function AX = 5380h: Get System Information (OEM)
    Subfunction BH = C3h**

64APMOEM

Returns system information fields. Information fields are arbitrary 32-bit values and are accessed by ID. Not all fields are expected to be supported on every machine, so a method is provided for finding out which field IDs are valid on a particular machine.

On Entry:
AH = 53h
AL = 80h
BH = C3h
BL = System information field ID to be returned
    Range: 0–254
    255 is reserved

On Return:
if successful:

    Carry = 0

    AX = Error codes
        C301h = Invalid field ID; valid IDs greater than this one.
        C302h = Invalid field ID; no valid IDs greater than this one.

if NOT successful:

    Carry = 1

    DX:AX = 32 bits of system information field data.
        (format is field-specific)

    BX:CX = 0 (32 bits of unused field flags.
        (these could be implemented inthe future)

INT 15h, Function AH = 80h; Device Open

BIOS

Opens a device. Hook this function as a multitasker needing to keep track of device ownership.

On Entry:
AH = 80h
BX = Device ID
CX = Process ID

On Return:
if successful:

    Carry = 0

    AH = 00h

if NOT successful:

    Carry = 1

    AH = Status
        80h   Invalid command
        86h   Function not supported

See Also: Functions AH=81h, AH=82h

INT 15h, Function AH = 81h; **Device Close**

BIOS

Closes a device. Hook this function as a multitasker needing to keep track of device ownership.

On Entry:
    AH = 81h
    BX = Device ID
    CX = Process ID

On Return:
    if successful:
        Carry = 0
        AH = 00h
    if NOT successful:
        Carry = 1
        AH = Status
            80h   Invalid command
            86h   Function not supported

See Also: Functions AH=80h, AH=82h

INT 15h, Function AH = 82h; **Program Termination**

BIOS

Closes all devices, opened by the given process ID with function 80h. Hook this function as a multitasker needing to keep track of device ownership.

On Entry:
    AH = 82h
    BX = Process ID

On Return:
    if successful:
        Carry = 0
        AH = 00h
    if NOT successful:
        Carry = 1
        AH = Status
            80h   Invalid command
            86h   Function not supported

See Also: Functions AH=80h, AH=81h

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 to by ES:BX is set. The duration of the wait interval is a multiple of 976 microseconds, because most BIOSs use 1/1024 second test interrupt from AT real-time clock chip (available on INT 70).

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 calling program's memory with bit 7 set
            when the interval expires.

On Return:

    AH = 00h
    if successful:
        Carry = 0
        AL = Value written to CMOS RAM register B
    if NOT successful:
        Carry = 1
        AL = 00h   (Function busy)

See Also: Functions AH=41h, AH=86h, and INT 70h

INT 15h, Function AX = 8301h; **Cancel Event Wait Interval**

BIOS

Cancels the event wait interval.

On Entry:

    AH = 83h
    AL = 01h

On Return:

    None

INT 15h, Function AH = 84h; **Read Joystick Input Settings**

BIOS

Reads the joystick input settings. Setting register pair DX to 00h reads the current switch settings. Setting register pair DX to 01h reads the current resistive input settings.

On Entry:

    AH = 84h
    DX = 00h (Read current switch settings)
        01h (Read resistive inputs)

On Return:

    AL = Switch settings (bits 7–4) (if DX = Read current switch settings)
        AX = A(x) Value   (if DX = Read resistive inputs (positions)
        BX = A(y) Value   (if DX = Read resistive inputs (positions)
        CX = B(x) Value   (if DX = Read resistive inputs (positions)
        DX = B(y) Value   (if DX = Read resistive inputs (positions)

INT 15h, Function AH = 85h; **System Request (SysReq) Key Pressed/Released**

BIOS

Called by INT 09h ISR when the SysReq key is pressed. Application programs
that must monitor the SysReq key must hook this call, since the default handler
returns successful.

On Entry:

AH = 85h
AL = 00h   (Key pressed)
   01h   (Key released)

On Return:

if successful
   Carry = 0
   AH = 00h
if NOT successful:
   Carry = 1
   AH = 80h   (Invalid command)
       86h   (Function not supported)

See Also: Function INT 09h

INT 15h, Function AH = 86h; **Wait (Elapsed Time)**

BIOS

Suspends calling program for specified time period. The register CX and DX
values are in microseconds, but are rounded down to the nearest multiple of 976.

On Entry:

AH = 86h
CX = High byte of time interval to wait (in microseconds)
DX = Low byte of time interval to wait (in microseconds)

On Return:

Carry = Status flag
   0 (successful) after the time interval has concluded
   1 (NOT successful) returns upon error or wait 83h in progress

---

INT 15h, Function AH = 87h; Copy Extended Memory Block

BIOS

Copies a block of memory from anywhere in the system address space to anywhere else in the system address space. Memory space that is not generally available to real-mode programs is accessible through this function.

The copy is performed in protected mode with interrupts disabled by the default BIOS handler. Many 386 memory managers perform the copy with interrupts enabled. This function is incompatible with the OS/2 compatibility box.

On Entry:
    AH = 87h
    CX = Number of 16-bit words to copy (max 8000h)
    ES:SI = Pointer to 48-byte GDT table, allocated by calling program

On Return:
    if successful
        Carry = 0
        AH = 00h   (source successfully copied to destination)
    if NOT successful
        Carry = 1   (returned immediately upon error)
        AH = Error Status
            01 = RAM parity error occurred
            02 = Interrupt error
            03 = Address line 20 gating failed

Global Descriptor Table

Offset	Description
00h	Uninitialized, Null descriptor (used by BIOS)
08h	Uninitialized, 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, Made into SS descriptor

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.

See Also: Functions AH=88h, AH=89h

INT 15h, Function AH = 88h; Read Extended Memory Size

BIOS

Determines the extended memory size by reading the amount of memory above 1 megabyte from the CMOS RAM locations 30h and 31h.

On Entry:
    AH = 88h

On Return:
    AX = Number of contiguous 1 kilobyte blocks of extended memory

---

INT 15h, Function AH = 89h; **Switch Processor to Protected Mode**

BIOS

The GDT describes the memory management environment that is in effect on return to the calling program. All table descriptors are initialized by the calling program, except the BIOS code segment descriptor. The "access rights" byte of the BIOS descriptor is 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.

On Entry:
AH = 89h
BH = Interrupt number for IRQ 8 (IRQ 9Fh use next 7 interrupts)
BL = Interrupt number for IRQ 0 (IRQ 17h use next 7 interrupts)
CX = Offset into protected mode CS to jump to
ES:SI = Pointer to GDT for protected mode

On Return:
if successful

Carry = 0

AH = 00h   (in protected mode at specified address)

if NOT successful

Carry = 1

AH = 0FFh   (error enabling address line 20)

Global Descriptor Table

Offset	Description
00h	Uninitialized, Null descriptor (used by BIOS)
08h	Uninitialized, Made into GDT descriptor
10h	Uninitialized, IDT descriptor
18h	Uninitialized, DS descriptor
20h	Uninitialized, DS descriptor
28h	Uninitialized, SS descriptor
30h	Uninitialized, CS descriptor
38h	Uninitialized, Builds descriptor for BIOS CS

The program that enters 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.

" **NOTE:**   *The protected mode CS must reference the same memory as the CS from which this function is called, because execution continues with the address following the interrupt call.*

BH and BL must be multiples of 8.

See Also: Functions AH=87h, AH=88h

 INT 15h, **Function AH = 90h**; **Device Busy Loop**

BIOS

Implements a time-out while waiting for the specified device to respond. Hook
this up as a multitasker to keep track of device ownership.

On Entry:
    AH = 90h
    AL = Device type
        00h = Fixed disk
        01h = Diskette
        02h = Keyboard
        03h = PS/2 pointing device
        80h = Network
        0FCh = Hard disk reset (PS/2)
        0FDh = Diskette driver motor start
        0FEh = Printer
    ES:BX = Pointer to request block (If device type 80h – BFh)

On Return:
    if successful:
        Carry = 1   ; wait was performed (I/O complete or time-out)
    if NOT successful:
        Carry = 0   if unsuccessful; no wait performed (driver must perform wait)

Type codes allocated as follows:
    00h–7Fh   Non-reentrant devices. OS must arbitrate access.
    80h–BFh   Reentrant devices. ES:BX points to a unique control block.
    C0h–FFh   Wait-only calls, no complementary INT 15, fn91h call.

See Also: Functions AH=91h, INT 13/AX=00h, INT 17/AH=00h, and INT 1A/AH=83h

INT 15h, Function AH = 91h; **Complete the Interrupt**

BIOS

Sets a device flag and completes the interrupt. Hook this as a multitasker with a requirement to keep track of device ownership.

On Entry:
    AH = 91h
    AL = Device type
        00h = Fixed disk
        01h = Diskette
        02h = Keyboard
        03h = PS/2 pointing device
        80h = Network
        0FCh = Hard disk reset (PS/2)
        0FDh = Diskette driver motor start
        0FEh = Printer
    ES:BX = Pointer to request block (if device type 80h – BFh)

On Return:
    AH = 00h
    Carry = Status flag
    if successful:
        Carry = 1   ; wait was performed (I/O complete or time-out)
    if NOT successful:
        Carry = 0   if unsuccessful; no wait performed (driver must perform wait)

See Also: Function AH=99h

INT 15h, Function AH = C0h: Get System Configuration Parameters

BIOS

Returns the address of the system configuration parameters.

On Entry:
  AH = C0h

On Return:
  if successful
    Carry = 0
    AH = 00h   (system model determined)
    ES:BX = Pointer to ROM system descriptor table
  if NOT successful
    Carry = 1
    AH = 86h   (could not determine system model; may not support call)

Format of ROM Configuration Table

Offset	Size	Description
00h	dword	Number of bytes following this word in table (normally, this is 16 bytes).
02h	byte	Model (System ID byte)
03h	byte	Submodel (Distinguishes between models)
04h	byte	BIOS revision: (0=first release, 1=next 2,3,4, etc.)
05h	byte	Feature byte 1 (See table #402)
bit 7		DMA channel 3 used by hard disk BIOS
bit 6		Second 8259 installed
bit 5		Real-time clock installed
bit 4		INT 15h, fn 04h called upon INT 09h
bit 3		Wait for external even supported (INT 15 fn 41h) used on Convertible; reserved on PS/2 systems extended BIOS area allocated at 640K
bit 2		Extended BIOS area allocated at 640K
bit 1		Bus is Micro Channel instead of PC
bit 0		System has dual bus (Micro Channel + ISA)
06h	byte	Set to zero (reserved by IBM)
07h	byte	Set to zero (reserved by IBM)
08h	byte	Set to zero
09h	byte	Set to zero (Award copyright here)

INT 15h, Function AH = C1h: Return Extended BIOS Data Area Segment (EBDA)

BIOS
Returns the EBDA.

On Entry:
AH = C1h

On Return:
if successful

Carry = 0

AH = 00h

ES = Pointer to segment address of EBDA

if NOT successful

Carry = 1

AH = 86h   (no EBDA)

CL = Resolution   (see Function AX=C203h for details)

DL = Sample rate   (see Function AX=C202h for details)

AH = Status of operation   (see Function AX=C200h for details)

INT 16h, Keyboard BIOS Services

Access to the keyboard BIOS services is through Interrupt 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.

INT 16h, Function AH = 00h: Read Next ASCII Character from Keyboard

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.

On Entry:
   AH = 00h

On Return:
   AL = ASCII character (zero if non-ASCII)
   AH = IBM-compatible scan code (zero if ASCII)

Refer to a standard ASCII character code table for INT 16h, for more information on returned character codes.

INT 16h, Function AH = 01h: Get Keyboard Status

BIOS

Gets the keyboard status. 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 (with a requirement) style keyboard filter, masking duplicates and removing undefined 101 keys.

A command should be sent to the keyboard to update the LED settings, to ensure that the keyboard LEDs match the current flag settings.

" **NOTE:**   *For the 6400, this function applies to the annunciators, not LEDs.*

On Entry:
   AH = 01h
On Return:
   If key is available:
      AH = IBM-compatible scan code (zero if ASCII)
      AL = ASCII character (zero if non-ASCII)
   If keyboard buffer is empty:
      Zero flag = 1

Refer to a standard ASCII character code table for INT 16h, for more information on returned character codes.

INT 16h, Function AH = 02h;Get Shift Status

BIOS

Returns the shift status.

On Entry:

AH = 02h

On Return:

AL = Shift status

Bit	Shift Status Definition
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 = 04h: Turn Off Key Click
                   AL = 00h;   Buzzer Control;

BIOS

Turns off the Key Click.

On Entry:

AH = 04h
AL = 00h   (Turn key click off)

On Return:

None

INT 16h, Function   AH = 04h: Turn On Key Click
                   AL = 01h;   Buzzer Control;

BIOS

Turns off the Key Click.

On Entry:

AH = 04h
AL = 01h   (Turn key click on)

On Return:

None

INT 16h, Function   AH = 04h: Set Key Click Parameters
AL = 02h;   Buzzer Control

BIOS

Sets the Key Click parameters.

On Entry:
AH = 04h
AL = 02h (Set parameters)
BL = Volume (Not implemented)
CX = Duration in milliseconds
DX = Frequency in Hz

On Return:
None

INT 16h, Function   AH = 04h: Get Key Click Parameters
AL = 03h;   Buzzer Control

BIOS

Gets the Key Click parameters.

On Entry:
AH = 04h
AL = 03h   (Get parameters)

On Return:
AL = State
0 = Off
1 = On
BL = Volume (Not implemented)
CX = Duration in milliseconds
DX = Frequency in Hz

INT 16h, Function   AH = 04h: Set Error Beep Parameters
AL = 04h;   Buzzer Control

BIOS

Sets the Error Beep parameters.

On Entry:
AH = 04h
AL = 04h (Set parameters)
BL = Volume (Not implemented)
CX = Duration in milliseconds
DX = Frequency in Hz

On Return:
None

INT 16h, Function   AH = 04h: Get Error Beep Parameters
                  AL = 05h;   **Buzzer Control**

BIOS

Gets the Error Beep parameters.

On Entry:
AH = 04h
AL = 05h   (Get parameters)

On Return:
AL = State
    0 = Off
    1 = On
BL = Volume
    (not implemented)
CX = Duration in milliseconds
DX = Frequency in Hz

INT 16h, Function   AH = 04h: Sound Key Click
                  AL = 06h;   **Buzzer Control**

BIOS

Sounds the key click.

On Entry:
AH = 04h
AL = 06h (Make the sound)

On Return:
None

INT 16h, Function   AH = 04h: Sound Error Beep
                  AL = 07h;   **Buzzer Control**

BIOS

Sounds the error beep.

On Entry:
AH = 04h
AL = 07h (Make the sound)

On Return:
None

INT 16h, Function   AH = 04h: Select Buzzer
                  AL = 08h;   **Buzzer Control**

BIOS

Selects the specified buzzer.

On Entry:
AH = 04h
AL = 08h (Make the selection)
BL = Buzzer selection
    bit 0:   Internal buzzer
    bit 1:   External (RS-232) buzzer (Not implemented)

On Return:
None

INT 16h, Function	AH = 04h: Beep the Buzzer
	AL = 09h; Buzzer Control

BIOS

Beep the buzzer, as per the specified parameters.

On Entry:

AH = 04h

AL = 09h (Sound the beep)

BL = Volume (Not implemented)

CX = Duration in milliseconds

DX = Frequency in Hertz

On Return:

None

INT 1Ah, Timer and Real-Time Clock Services

Unsupported functions return with the carry flag set. The carry flag is also set if the real-time clock (RTC) is not operational. Individual functions are selected by the content of the AH register when INT 1Ah is invoked.

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 (1800B0h clock ticks per 24 hours) 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 register AL, resets 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 midnight has not passed since last read)
   1   (If midnight has passed since last read)

" **NOTE:**   *Since the timer overflow flag is cleared, in the event an application calls this function after midnight, before DOS does, DOS does not recognize that another day has passed and does not advance the DOS date.*

See Also: Functions AH=01h, AH=02h

INT 1Ah, Function   AH = 01h: Write System Timer Ticks

BIOS

Sets the BIOS data values located at 40:6Ch and 40:6Eh to the data contained in registers CX and DX, respectively. It thereby sets the system timer tick count. Executing 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

See Also: Functions AH=03h

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. This function verifies that the update-in-progress bit is zero and disable interrupts before accessing the CMOS data.

On Entry:
    AH = 02h
    DS = Internal registers

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 = off   *(Standard Time)*
      1 = on
    Carry = Status Flag
      0   (success)
      1   (failure)

See Also: Functions AH=03h, AH=04h

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. This function waits for the update-in-progress bit to become zero and disable interrupts before accessing the CMOS data.

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 = off   *(Standard Time)*
      1 = on   *(Daylight Savings Time)*

On Return:
    Carry = Status Flag
      0   (success)
      1   (failure)

See Also: Functions AH=01h, AH=02h, AH=05h

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. This function waits for the update-in-progress bit to become zero and disable interrupts before accessing the CMOS data.

On Entry:
    AH = 04h
    DS = Internal registers

On Return:
    If successful:
        Carry = 0
        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)
    If NOT successful:
        Carry = 1

See Also: Functions AH=02h, AH=05h

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. This function waits for the update-in-progress bit to become zero and disable interrupts before accessing the CMOS data.

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:
    Carry = Status Flag
        0   (success) *(Clock is operating)*
        1   (failure)

See Also: Function AH=04h

INT 1Ah, Function   AH = 06h: Set the Real-Time Clock Alarm

BIOS

If the alarm is currently enabled, this 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 waits for the update-in-progress bit to become zero and disable interrupts before accessing the 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.

On Entry:
  AH = 06h
  CH = Hours in BCD (0–23)
  CL = Minutes in BCD (0–59)
  DH = Seconds in BCD (0–59)
  DS = Internal registers

On Return:
  Carry = Status Flag
    0   (success) *(Clock is operating)*
    1   (failure)

See Also: Function AH=07h

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
  DS = Internal registers

On Return:
  None

See Also: Function AH=06h

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 waits for the update-in-progress bit to become zero and disable interrupts before accessing the CMOS data.

On Entry:

    AH = 09h
    DS = Internal registers

On Return:

    CH = Hours in BCD (0–23)
    CL = Minutes in BCD (0–59)
    DH = Seconds in BCD (0–59)
    DL = Alarm status
        00h = Alarm not enabled
        01h = Alarm enabled, but does *not* power-on system (not used)
        02h = Alarm enabled, and does power-on system
    Carry = Status Flag
        0   (success) or
        1   (failure)

See Also: Function AH=08h

INT 1Ah, Function   AH = FFh: Load the Real-Time Clock to Internal Clock

---

BIOS

On Entry:

    AH = 0FFh

On Return:

    Carry = 0   (success)
             1   (failure)

INT 7Ah, Scanner Services: 6400IKPS.EXE

INT 7Ah, Function AH = 80h: Scanner On

SCANBIOS

Turns the scanner on. The scanning system must be on in order to scan and decode bar codes.

On Entry:
>   AH = 80h
>
>   AL = 0FFh =   On decode, the API puts the *SCAN_Key* into the BIOS key buffer,
>   then the value is read from the keyboard by the application.
>
>   BX = SCAN_Key value. The application recognizes that bar code data should
>   be available when this value is read from the keyboard.

On Return:
>   None

INT 7Ah, Function AH = 81h: Scanner Off

SCANBIOS

Turns the scanner off. The scanning system must be off when the *Set Scanner Type* function is called.

On Entry:
>   AH = 81h

On Return:
>   None

INT 7Ah, Function AH = 82h: Set Scanner Type

SCANBIOS

Establishes the scanner type. It must be issued while the scanner is off. Using this function while the scanner is on, has no effect.

On Entry:
>   AH = 82h
>
>   DL = Scanning type
>>   02h   Laser
>>   10h   Auto_Scan causes the scanner to scan and decode once, when the
>>   *Scanner On* function is called. To do an Auto_Scan, calls are
>>   required to be made to the *Scanner Off* function,
>>   to the *Set Scanner Type* function (optional),
>>   and to the *Scanner On* function.

On Return:
>   None

INT 7Ah, Function AH = 83h: Set Bar Code Options

SCANBIOS

Tells the scanning system which symbologies to enable, as well as which options. All symbologies are disabled; then those symbology types, passed in the *Symbology_Type* member of the *Barcode_Options* structure, are enabled.

Some of the following options are validated by the internal engine and some are validated by software. An explanation of these validations is included in the *Internal Laser Engine Symbology Validations* paragraph and the *Tethered Laser Symbology Validation*s paragraph, in the *Supported DOS Applications* section of this publication.

This function can be called anytime to set the bar code options.

On Entry:
> AH = 83h
> DS:DX = Segment:Offset pointer to array of *Barcode_Options* structures,
>   with *Symbology_Type* of last structure set to 00h.

On Return:
> None

Definitions of structures, values, and bits:

> **Barcode_Options structure**
> > byte Symbology_Type
> > byte Minimum_length
> > byte Maximum_length
> > byte Generic_Options
> > byte Symbology_Options
> > byte Drop_Leading_Trailing
> > byte Reserved_1
> > byte Fixed_Length_1
> > byte Fixed_Length_2
> > byte Fixed_Length_3
> > byte Fixed_Length_4

> **Symbology_Type values**

> | Hex | ASCII | Symbology |
> |------|---------|--|
> | 0x00 | \<NUL\> | last Barcode_Option structure in array |
> | 0x01 | \<SOH\> | ignore this Barcode_Option structure |
> | 0x08 | \<BS\> | MSI Plessey |
> | 0x10 | \<DLE\> | UPC |
> | 0x11 | \<DC1\> | EAN ★ |
> | 0x20 | \<space\> | Codabar |
> | 0x28 | '(' | Code 39 |
> | 0x32 | '2' | Interleaved 2 of 5 |
> | 0x40 | '@' | Code 128 |

★ The enabling *Symbology_Type*, for EAN-8 with addon 5, is different from the *Decoded_Symbology_Type* (see the **Read Bar Code Data** function), to differentiate it from the ambiguous EAN-13 with no addon.

Minimum_Length value
Maximum_Length value
> The number of decoded bar code data characters must be between the value of *Minimum_Length* and the *Maximum_Length*, inclusive. *Minimum_Length* must not be greater than *Maximum_Length*. If any of the fixed lengths are non-zero, then these two parameters are ignored. If all six parameters are zero, then any length bar code is decoded.
>
> The *Minimum_Length* and *Maximum_Length* values are not used for *Interleaved 2 of 5* bar codes. Use the *Fixed_Length_1* and *Fixed_Length_2* parameters.

Generic_Options bits
> bit 0   Enable redundancy for this symbology. If any Barcode_Options structure has this bit set, then the engine redundancy is enabled for all symbologies.

Symbology_Options bits

MSI Plessey Symbology_Options bits
> bit 1   Enable validation application check digit as modulus 10 type,
>
> bit 2   Enable validation application check digit as modulus 11 type:
> Bits 1 and 2 are mutually exclusive. If both are on, then both mod check types are executed, causing a validation failure, resulting in no decode. The check character is returned as part of the decoded data.

UPC and/or EAN Symbology_Options bits
> bit 0   Enable decode of addon 2, if detectable,
> bit 1   Enable decode of addon 5, if detectable,
> bit 3   Enable decode of UPC-E (short UPC),
> bit 4   Enable expanding UPC-E format to UPC-A:
> > UPC-E bar codes are expanded to the UPC-A format by insertion of the suppressed zeros. Expansion reverses the UPC-specified process for suppressing the zeros from the original bar code.
>
> bit 5   Enable expanding EAN-8 format to EAN-13:
> > EAN-8 bar codes are expanded to the full EAN-13 format by insertion of five leading zeros, and calculation of a new check digit.
>
> Decode formats for UPC/EAN are as follows:

Bar Code	Format
UPC-E	nddddddC
EAN-8	fnddddddC
UPC-A	ndddddddddddC
EAN-13	fnddddddddddddC *
UPC-E Addon 2	nddddddCaa
EAN-8 Addon 2	fnddddddCaa
UPC-A Addon 2	ndddddddddddCaa
EAN-13 Addon 2	fnddddddddddddCaa
UPC-E Addon 5	nddddddCaaaaa
EAN-8 Addon 5	fnddddddCaaaaa *
UPC-A Addon 5	ndddddddddddCaaaaa
EAN-13 Addon 5	fnddddddddddddCaaaaa

where:
- n   a number system digit
- d   a bar code data digit
- C   a check digit
- f   an EAN flag 1
- a   an addon digit

* The *Decoded_Symbology_Type* (see the **Read Bar Code Data** function) is different from the enabling *Symbology_Type* for EAN-8 with addon 5 to differentiate it from the ambiguous EAN-13 with no addon.

Code 39 Symbology_Options bits

bit 0   Enable mod check validation:
The last encoded character is validated as a modulus 43 check character, as specified by Automatic Identification Manufacturers (AIM) appendix E.3. The check character is returned as part of the decoded data.

bit 1   Enable concatenation:
Each Code 39 bar code beginning with a space character causes the next bar code to be appended to the prior decodes as specified by AIM (appendix E.1). The leading space character is not included in the decode data. Refer to the **Read Bar Code Data** function output values for CX and DX, for a description of the concatenation process.

bit 2   Enable translation of full ASCII escape sequences:
Encoded escape sequences are translated to ASCII characters according to the full ASCII mode of AIM (appendix E.2). An invalid Code 39 full ASCII sequence causes a no-decode.

Interleaved 2 of 5 Symbology_Options bits

bit 0   Enable validation of I2/5 check digit:
The last digit of I2/5 bar codes is validated as a check digit as specified by AIM (appendix E.1). The check digit is returned as part of the decoded data.

Code 128 Symbology_Options bits

bit 0   Enable UCC/EAN-128 format:
UCC/EAN Func1 support is provided with minor distinction from the specification. Embedded Func1 characters are decoded as the ASCII sequence, "]C1", instead of <GS> (0x1D), because <GS> is not displayable.

If the UCC/EAN-128 standard support is enabled and a Func1 character immediately follows the Start character, then "]C1" is prepended to the decoded bar code data, and each subsequent Func1 in the bar code will be decoded as "]C1".

If the UCC/EAN-128 standard support is enabled and a Func1 character does NOT immediately follow the start character then "]C0" is prepended to the decoded bar code data and each subsequent Func1 is dropped from the decoded bar code data.

If the UCC/EAN-128 standard support is not enabled, then Func1 characters in the bar code are dropped from the decoded data.

Func2 concatenation is always enabled.

Func3 is used by the physical scanner device; and the decoded data is not returned.

Func4 is ignored and dropped from the decoded data.

Drop_Leading_Trailing Byte value

The high 4 bits represent the number of characters to remove from the leading end of the bar code data. The low 4 bits represent the number of characters to remove from the trailing end of the bar code data. The drop trailing option drops characters from the trailing end of UPC and EAN addon supplements. This option cannot be used to remove the check digit character from UPC or EAN bar codes when they have an addon supplement.

Fixed_Length_1 value
Fixed_Length_2 value
Fixed_Length_3 value
Fixed_Length_4 value

If any *Fixed_Length* is non-zero, then only bar codes of those lengths are valid; and *Minimum_Length* and *Maximum_Length* are ignored. *Fixed_Length* values of zero are ignored. Use *Fixed_Length*s to increase the reliability of symbologies that are less secure.

For Interleaved 2 of 5:

Fixed_Length_3 and *Fixed_Length_4* are ignored. *Fixed_Length_1* must be set to the actual number of barcode digits. A second barcode length can be set to *Fixed_length_2.* If only one length of barcode is to be scanned, set *Fixed_Length_2* to zero.

INT 7Ah, Function AH = 84h: Read Bar Code Data

SCANBIOS

Initiates the reading of bar code data, then returns with a value that determines if bar code data is available at the location specified by the input pointer.

On Entry:

    AH = 84h

    DS:DX = Segment:Offset pointer to buffer where bar code is to be stored

    CX = Length of buffer

On Return:

    AL = Scanner data indicator

        0 = Bar code data available

        1 = No bar code data

    BH = Decode type (Decoded_Symbology_Type)

HEX	ASCII	Symbology
0x08	\<BS\>	MSI Plessey
0x10	\<DLE\>	UPC-A with or without addon 2 or 5
		UPC-E with or without addon 2 or 5
0x11	\<DC1\>	EAN-13 with or without addon 2 or 5
		EAN-8 with or without addon 2
		EAN-8 without addon 5
0x12	\<DC2\>	EAN-8 with addon 5 (ambiguous with EAN-13 with no addon) ★
0x20	\<space\>	Codabar
0x28	'('	Code 39
0x32	'2'	Interleaved 2 of 5
0x40	'@'	Code 128

    ★ The *Decoded_Symbology_Type* for EAN-8 with addon 5, is different from the enabling *Symbology_Type* (see the **Set Bar Code Options** function) to differentiate it from the ambiguous EAN-13 with on addon.

    BL = Direction of scan

        0 = Reverse – right-to-left

        1 = Forward – left-to-right

    CX = Length of data read (number of bar code characters)

    This includes previously concatenated characters for those symbologies that have concatenation enabled.

    DX = Previously concatenated characters.

    This is the number of characters that were concatenated (prepended) to this bar code from previous scan(s), excluding this scan. The high bit of the DX register is set, if this decode indicates a need to concatenate the next scan. Concatenation is terminated when one of the following is true:

            1 a non-concatenating bar code is scanned

            2 a different symbology is decoded

                (where the *Decoded_Symbology_Type* changes)

            3 the **Flush Scanner Concatenation Buffer** function is called

---

INT 7Ah, Function AH = FEh: Flush Scanner Concatenation Buffer

---

SCANBIOS

The concatenation buffer is flushed automatically, if a non-concatenating bar code is scanned or the decoded symbology type changes. The application may need to flush the buffer, if it recognizes that the last bar code scanned was concatenated.

On Entry:
   AH = 0FEh

On Return:
   None

Section 6

Reference,
System Information

.. ..

Introduction

This section contains system information, intended for supporting application programming efforts on the PEN*KEY**R** 6400 Computer. This includes information about hardware ports, messages, error codes, I/O and IRQ maps, system variables.

Topic Summary

List of Tables

ROM DOS 5

The ROM DOS 5 product, shipped in the 6400 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 the code can be loaded. In the 6400 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, the following default shell statement is used:

```
SHELL=C:\COMMAND.COM /P C:\
```

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 applications. However, since ELANUMP.SYS cannot support expanded memory, EMM386.EXE is still the only method for providing expanded (EMS) memory.

COMMAND.COM as Command Processor

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 6400 Computer, as described later in this section. If you prefer to use the unmodified version of COMMAND.COM, you can get it from the full DOS option, available on diskette. Copy the new COMMAND.COM over the old one on the C:\ drive.

CONFIG.SYS Processing

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.

COMMAND.COM Changes — Specific to Norand

All of the messages displayed by COMMAND.COM have been shortened or split into multiple lines so they will fit the 20-column screen on the 6400 Computer.

" Error messages are displayed as numbers.

" The critical error handler displays a menu that includes numeric prompts that the user can respond to without using an alpha keyboard.

Example:
```
“Abort, Retry, Fail?”
```

be
```
2) Abort,
1) Retry,
3) Fail?
```

The reason the numbers are not always in a natural order is that each line is displayed independently, depending on the options that are available. If the numbers were to be rearranged to look correct for one condition, they would look wrong for another.

" The Echo command was modified to allow “ECHO,” to display a blank line instead of “Cannot load program.” In a related change, comma (,), equals (=), and quote (”) now terminate a keyword comparison.

" A bug with the GOTO was eliminated so that it now works correctly with labels of two or more characters.

Boot Process

The following paragraphs describe the various methods used to boot the 6400 Computer. Also described are the options that are possible during the boot process, which you exercise by holding a key down during the boot phase or by reconfiguring a CMOS setting.

Four Key Reset

The 6400 Computer can be reset by pressing four keys simultaneously for approximately one second:

 " BLUE shift plane key
 " GOLD shift plane key
 " ENT (left enter)
 " ENT (right enter)

Cold Booting

A cold boot cycle normally should never be performed by a customer. A *cold boot* is defined as a boot cycle in which power is applied for the first time (when the 6400 Computer contains no backup battery or main battery, and no external power is being applied). A cold boot is normally done only at the factory. The only time you would need to be perform a cold boot, would be after a situation where the main battery pack has drained, no backup battery was installed, *and* no external charge is applied. The following paragraphs describe some of 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 (POST) are Run

A series of tests is performed on the hardware to ensure that the system is functioning properly. For the most part, these tests are performed without any 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, refer to the *Audible Error Codes* paragraph, in the section entitled, *Reference.*

Video BIOS is Enabled

Early in the Power-On Self-Test (POST) phase, the video BIOS is given control to initialize the screen. From that point, 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. 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 that is maintained by a backup battery. In contrast, all memory in the 6400 Computer is maintained at all times, with a fully charged backup battery. This memory can be maintained for approximately 100 hours in the event of a main battery failure. The CMOS RAM is special only because of the PC-compatible software that expects the CMOS RAM to exist and to contain certain configuration values.

Invalid RAM Drive Message

One of the first tests involving extended memory is to verify that the RAM drive, if it exists, has not been corrupted by a power failure. If the RAM drive signature has been corrupted, the RAM drive is removed and the default boot drive is set to drive C. The *Invalid RamDrive* message is displayed and a single beep is issued.

Testing XMS Memory Message

This message displays while the power-on test of extended memory is performed. The initials XMS refer to the eXtended Memory Specification 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 will 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 scans through system memory in the address range of C800:0 through F000:0, 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 ROMs or shadow RAM. A special signature precedes each BIOS extension. After that

signature has been validated, the corresponding BIOS extension is made active (or posted). Currently the 6400 Computer includes the following BIOS extensions: the *Datalight, Inc., Cardtrick Flash File System* and ROM DOS 5 (ROMDOSLO.BIN).

The Flash File System is already located and executed prior to this stage of the boot process since it resides in the address range of D000:0 through D7FF:F. ROM DOS 5 itself is located at E000:0. That leaves 32 kilobytes of shadow RAM between C800:0 and CFFF:F and 32 kilobytes between D800:0 and DFFF: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, this 32 kilobytes of shadow RAM is scanned for BIOS extensions and remains resident as write-protected RAM.

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 ROM DOS 5 product. The primary difference is that much of ROM DOS executes directly from shadow RAM instead of from conventional memory. All the DOS variables and a significant amount of code as well 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 simply 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, used only for 2.4GHz, 902MHz, and UHF radio cards.
>
> Drive 1 = B: Not available.

A DOS 5 embedded device driver uses Interrupt 15h, Function 87h (move extended memory), to access the RAM drive (drive D:) and the flash drive data (drive C:, accessed through flash file system Int 13h). ROM DOS 5 obtains the starting address for each drive by reading values stored in the BIOS data area of system memory.

CONFIG.SYS Is Loaded and Processed

Once ROM DOS has initialized, it reads and processes the CONFIG.SYS file on the default drive. The default drive is chosen by now, 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 sample `CONFIG. SYS` files and notes on booting.

COMMAND.COM Is Processed

The default command shell for ROM DOS 5 is:

```
SHELL=C:\COMMAND.COM /P C:\
```

After `COMMAND.COM` is loaded by this SHELL statement, AUTOEXEC.BAT is processed.

Warm Booting (or Resetting)

A *warm boot* is only slightly different from a *cold boot.* Since the two forms of booting are so similar, this publication may refer to booting without specifying warm or cold. Such references would imply that the form of boot does not matter.

For *warm booting,* the 6400 Computer must have been previously booted and still have power applied to it from either batteries or a charger. To warm boot, press the sequence CTL, ALT, and DEL. POST procedures are re-executed and ROM DOS restarts.

If a RAM drive has been 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 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 0 KB of extended memory is reported.

Boot Drive Selection

The 6400 Computer always boots DOS 5 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.

To obtain the ROM DOS bootup menu, press and release the [ALT] key, after resetting the unit.

You can use the "Start from" menu to override the CMOS setting by manually selecting the drive that is to be the default drive. 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 (C:) is used instead of the selected drive. Also, you can choose to skip the reading of the CONFIG.SYS and AUTOEXEC.BAT files, and boot directly to the C:> prompt. This can be very useful in case there is a problem in either of those two files. See the *Setup of PEN*KEY 6400 for PC Development* paragraph, in the *Getting Started* section, for more information on the "Start from" menu.

System Messages

Table 6-1
System Messages

Display Message	Meaning	Response
PEN*KEY BIOS Version: {n.nn}	Display of BIOS version	None
Testing XMS Memory	Performs extended memory tests	None
{nnnn} KB Extended	Reports number of extended memory KB tested	
{nnn} KB FLASH	Reports number of flash memory KB found	
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 6400 Computer, including RAM drive, is cleared.	Application download is required unless the application *was not stored* on the RAM drive.
Invalid RAM Drive	Power failure corrupted RAM drive but left CMOS intact. This occurs when swapping batteries and there is either a bad or missing backup battery.	
Hot Int. Failure	Hot interrupt failure.	Contact Customer Support Center
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.	
{nnnn} Keyboard Failure	Keyboard controller failure.	Error. Contact Customer Support Center
{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 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 identifies the code. The secondary set of beep codes consists of at least one long, one medium long, and one short beep. The following table shows the beep codes for POSTs.

Table 6-2
Audible POST Error Codes

Duration and # Beeps

Long	Medium	Short	Meaning
1	1	2	Unable to read clocks for peripherals.
1	1	4	ROM BIOS checksum failed.
1	3	1	RAM Refresh is not active.
1	3	3	First 64K of system RAM is not operating correctly.
3	4	3	LCD cannot be enabled.
4	2	3	Cannot enable A20 using port 64h (standard keyboard port)
4	2	4	Cannot enable A20 using port 92h (IBM fast A20 port).
4	2	5	Cannot enable A20 using port 0EEh (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

Hardware Ports

The following is a list of communication ports and the devices supported through those ports:

Table 6-3
Hardware Ports

COM Port	Device
COM1	Regular wired port, accessed through dock, or communications adaptor
COM2	Internal bar code scanner
COM3	IrDA port
COM4	Not Applicable

IRQ and Other Hardware Interrupts

The system supports the following hardware interrupts:

Table 6-4
Hardware Interrupts

Vector	Type	Usage
00h	processor	Divide error
01h	processor	Debug exceptions, such as 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
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) internal scanner
	processor	Segment not present
0Ch	IRQ4	Serial port 1 (COM1)
	processor	Stack fault
0Dh	IRQ5	COM3 (IrDA port)
	processor	General protection error
0Eh	processor	Page fault
0Fh	IRQ7	BATTERY DATA
10h	processor	Coprocessor error
70h	IRQ8	Real-time clock periodic interrupt
	IRQxx	Proxim 2.4 GHz Radio Card (Set up in NET.CFG) defaults to IRQ15
SMI		System Management Interrupt

I/O Map

Table 6-5
I/O Addresses 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	Proxim 2.4GHz card radio
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, such as LPT1)*
280–2E7	*Reserved for PC Card general use*
2E8–2EF	*Reserved (PC Card and COM)*
2F0–2F7	*Reserved for PC Card general use*
2F8–2FF	COM 2 (scanner)
320–35F	*Reserved for PC card general use*
378–37F	*Reserved (PC Card and Parallel port)*
380–38F	*Reserved (PC Card and SDLC)*
390–39F	*Reserved for PC Card general use*
3A0–3AF	*Reserved (PC Card and SDLC)*
3B0–3BF	*Reserved (MDA, EGA, and VGA)*
3C0–3DF	VGA and CGA
3E8–3EF	COM 3 (IrDA port)
3F6–3F7	*Reserved for PC Card (primary ATA and IDE disk controller)*
3F8–3FF	COM 1

BIOS/CMOS System Variables

The ROM BIOS is the system software at the lowest layer of a PC operating environment. The 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. For that reason, the following lists name the data areas on which the ROM BIOS operates.

" **NOTE:**   *A full description of each variable used by BIOS is beyond the scope of this publication.*

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 that a ROM BIOS must operate on. In the following table, the entries that are unique to the NORAND BIOS are ***bold and italicized***. The rest are defined in accordance with the industry standard.

Table 6-6
BIOS Data in System RAM

Address	Size	Description
40:00	4 words	I/O address of up to four asynchronous communications adaptors (COM1–COM4)
40:08	3 words	Reserved: I/O address of up to three printer adaptors (LPT1–LPT3)
40:0E	1 word	Segment address of extended BIOS Data Area
40:10	1 word	Equipment status word (returned by interrupt 11h)
40:12	1 byte	Reserved: POST status
40:13	1 word	Conventional memory size in kilobytes (returned by interrupt 12h)
40:15	***1 word***	***Pass or Fail indicators for RAM POST test***
40:17	1 word	Keyboard shift flags
40:19	1 byte	Alt-keypad accumulator
40:1A	1 word	Pointer to next keycode in keyboard buffer
40:1C	1 word	Pointer to next available location to save a keycode in the keyboard buffer
40:1E	16 words	Circular keyboard buffer
40:3E	1 byte	Reserved: Diskette recalibrate status
40:3F	1 byte	Reserved: Diskette motor status
40:40	1 byte	Reserved: Diskette master timeout control
40:41	1 byte	Diskette status return code — status of last RAM card access
40:42	7 bytes	Reserved: Diskette controller status bytes
40:49	1 byte	Video mode setting
40:4A	1 word	Number of video columns in memory (not all columns are represented onscreen)
40:4C	1 word	Current page size in bytes
40:4E	1 word	Current page address
40:50	8 words	Cursor position for each page
40:60	1 word	Cursor display mode (start and end line)

Table 6-6 (continued)
BIOS Data in System RAM

Address	Size	Description
40:62	1 byte	Current display page
40:63	1 word	I/O address of display hardware
40:65	1 byte	Current mode select register
40:66	1 byte	Current palette value
40:67	1 dword	Segmented address of video option ROM
40:6B	1 byte	Reserved: Last interrupt that occurred
40:6C	1 word	Time of day counter, low word
40:6E	1 word	Time of day counter, high word
40:70	1 byte	Timer overflow indicator
40:71	1 byte	System break flag
40:72	1 word	System reset flag
40:74	1 byte	Reserved: Last hard disk status
40:75	1 byte	Number of fixed drives
40:76	1 byte	Reserved: Fixed disk control byte
40:77	1 byte	Reserved: Fixed disk port offset
40:78	4 bytes	Reserved: Printer timeout table
40:7C	4 bytes	Serial timeout table
40:80	1 word	Offset to start of keyboard buffer
40:82	1 word	Offset to end of keyboard buffer
40:84	*1 byte*	*Number of rows onscreen (24/25)*
40:85	*1 word*	*Character height (bytes and characters)*
40:87	*1 byte*	*Video control bit flags*
40:88	*1 byte*	*Video feature bit flags*
40:89	*1 byte*	*VGA control bit flags*
40:8A	*1 byte*	*Display combination code table index*
40:8B	1 byte	Reserved: Diskette data rate information
40:8C	1 byte	Reserved: Fixed disk status register
40:8D	1 byte	Reserved: Fixed disk error register
40:8E	1 byte	Reserved: Fixed disk interrupt flag
40:8F	1 byte	Reserved: Diskette controller information
40:90	1 byte	Media type for drive 0
40:91	1 byte	Media type for drive 1
40:92	1 byte	Media type for drive 2
40:93	1 byte	Media type for drive 3
40:94	2 bytes	Reserved: Current track for each drive
40:96	1 byte	Keyboard status byte
40:97	1 byte	Keyboard LED status byte
40:98	1 dword	Segmented pointer to user wait flag
40:9C	1 dword	Wait timeout data word
40:A0	1 byte	Wait active flag
40:A1	39 bytes	Reserved
40:C8	10 bytes	Reserved for real mode restart stack, *used by RESTART*
40:D2	1 word	RS-232 timeout counter, used by interrupt 14h
40:D4	*4 bytes*	*Reserved, used by POST*
40:D8	*20 words*	*Reserved*

CMOS Registers

Registers in CMOS RAM are provided as part of a real-time clock interface. You access the registers by writing a register number to port 70h and 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, register definitions specific to Intermec are ***bold and italicized***. The remainder are defined by PC-compatibility standards.

Table 6-7
CMOS Register Assignments

Registers	Assignments
RTC_SECONDS	0
RTC_SECONDS_ALARM	1
RTC_MINUTES	2
RTC_MINUTES_ALARM	3
RTC_HOURS	4
RTC_HOURS_ALARM	5
RTC_WEEKDAY	6
RTC_DAY	7
RTC_MONTH	8
RTC_YEAR	9
RTC_REG_A	10
RTC_REG_B	11
RTC_REG_C	12
RTC_REG_D	13
RTC_CENTURY	032h
CMOS_DIAGS	0Eh
CMOS_SHUTDOWN	0Fh
SD_NORMAL_RESET	0
SD_REAL_MODE_ENTRY	1
SD_BOOTSTRAP	4
SD_EOI_JMP	5
SD_FAR_JMP	6
SD_BLOCK_MOVE1	7
SD_MEM_TEST	8
SD_BLOCK_MOVE2	9
SD_FAR_JMP2	10
SD_IRET	11
SD_FAR_RET	12
CMOS_DISKETTE1_TYPE	10h
CMOS_DISKETTE2_TYPE	11h
CMOS_FDISK_TYPE	12h
CMOS_EQUIPMENT	14h
CMOS_BASE_MEMLO	15h; Conventional memory size
CMOS_BASE_MEMHI	16h
CMOS_XTD_MEMLO	17h; Extended memory size
CMOS_XTD_MEMHI	18h
CMOS_FDISK1_TYPE	19h
CMOS_FDISK2_TYPE	1Ah

Table 6-7 (continued)
CMOS Register Assignments

Registers	Assignments
CMOS_CHKSUM_HI	2Eh; Checksum for 10h–2Dh
CMOS_CHKSUM_LO	2Fh
CMOS_POST_XTD_MEMLO	30h; Posted extended memory size
CMOS_POST_XTD_MEMHI	31h
CMOS_DOS_FLAGS	3Fh; ROM DOS boot flags
bMasterModeBoot	01h; Master Mode boot bit
bDefaultDrive	06h; 00=A:, 01=B:, 10=C:, 11=D:
CMOS_FLASH_INFORMATION	40h
FLASH_1_512K	02h
FLASH_2_512K	04h
FLASH_1_2MEG	08h
FLASH_2_2MEG	10h
FLASH_3_2MEG	18h
FLASH_4_2MEG	20h
bINTEL	40h; Set only for Intel flash type
bAMD	80h; Set only for AMD flash type

Reference,
Open Systems Publications

This section contains reference information, consisting of publications that were referenced from previous sections, as well as other recommended publications for application development activity.

" **NOTE:**   *All Intermec publications can be ordered through your Intermec sales representative.*

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
ISBN: 241704-001

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

Ralf Brown's Interrupt List, release 39, last updated 2/6/94

This public domain on-line 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 to directory
     `/afs/cs.cmu.edu/user/ralf/pub`
and get the files
     `inter??a.zip through inter??d.zip.`

You *must* change directory first and 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 DOS and Unix are available in the "archivers" subdirectory.

If connected to AFS, you can simply perform standard Unix/VMS/whatever directory listing and file copies 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.

Alternate Distribution Points (the list typically is available here 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], and**
> **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 with the following automated mail servers:

> In the United States:

> > " 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

> > " Colombia   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 will find a fairly recent release in the various UUCP archives.

System BIOS for IBM PCs, Compatibles, and EISA Computers, Second Edition

Phoenix Technology
Addison-Wesley Publishing Company
ISBN 0-201-57760-7

The Undocumented PC

A programmer's guide to I/O, CPUs, and fixed memory areas.
Addison-Wesley Publishing Company
ISBN 0-201-62277-7

Writing TCOM Modules in PL/N for the HHC

Intermec Technologies Corporation
P/N: 541-002-523

DOS 5.0 API Publications

PC Interrupts

Ralph Brown and Jim Ryle
Addison-Wesley Publishing
ISBN 0-201-57797-6

Hardware Interface Publications

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

Proxim

295 North Bernardo Avenue
Mountain View, CA 94043
(415) 960-1630
(415) 964-5181 FAX
1-800-229-1630

Sample Configuration Files

‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟   ‟

This section contains sample configuration files that may be used on the PEN*KEY® 6400 Computer.

Topic Summary

Sample AUTOEXEC.BAT

The following code represents a typical example of an AUTOEXEC.BAT file.

″ **NOTE:** *With limited space between page margins, the "for %%i in (...) do if not exist ... goto PC" text has been split in this listing, which places part of the text on the next line, making the "for %%i in (...) do" statement appear to be incomplete. For a more true example, refer to the actual file in the Tool Kit.*

```
@echo off
Echo PEN*KEY 6400 FLASH
Bios
elancfg /V1 /R1 /T2 /D4 /L25 /H1
for %%i in (net.cfg cdsvc1a.exe lsl.com r12pcm.com 64scn7b.exe
                    wtppkt.exe cfglit.dat) DO if not exist %%i goto PC
if exist 6418bios.exe goto GOT18BIOS
if not exist 6400bios.exe goto PC
if not exist fwp640h0.exe goto PC
goto GOTEMUL

:GOT18BIOS
if not exist fwp640h4.exe goto PC
```

```
:GOTEMUL
    delay "Press 0 for C>" /500
    if not errorlevel 1 goto PC
    misctsr –A –K –L –T –P0
    if exist 6418bios.exe 6418bios
    if exist 6400bios.exe 6400bios
    64scn7b
    cdsvc1a
    lsl
    r12pcm
    wtppkt
    if exist fwp640h4.exe fwp640h4
    if exist fwp640h0.exe fwp640h0
    goto EXIT

:PC
    misctsr –A –K –L –T –P1
    dosgas
:EXIT
```

Sample CONFIG.SYS

The following code represents a typical example of a CONFIG.SYS file.

```
break=on
buffers=30
files=128
lastdrive=f
stacks=9,256

device=c:\delay.exe
device=c:\himem.sys /machine:2
dos=high
device=c:\elanump.sys /X=C000,D000, D400, E400, E800, EC00
dos=umb

devicehigh=c:\elanapm.exe
devicehigh=c:\nordospm.exe
devicehigh=c:\clock.exe
devicehigh=c:\64apmoem.exe

shell=c:\command.com c:\ /e:512 /p
```

At a minimum, the following statements should be included:

```
shell=c:\command.com c:\ /p
device=c:\elanapm.exe
device=c:\nordospm.exe
devicehigh=c:\64apmoem.exe
```

Sample KEYREMAP.ASM

The following code represents a typical example of a key remapping file that could be modified, using keys that you want remapped.

After you create your own, or modify the sample code below, to suit your remapping needs, then you would create an object file, using TASM, the Borland Turbo assembler, and convert it to an executable, using TLINK, the Borland Turbo linker.

If you are more familiar with another product, it may only require minor changes to the source code, to be compatible with your assembler program.

The example below was created for the Borland assembler program.

```
Page   70, 132
TITLE KBSV


;#######################################################
;------------------------------------------------------
; Copyright (c) 1997 NORAND International UK
;
; Program KEYREMAP.ASM
;
; Jörgen Westman 1997-02-27
;
; Translates US 6400 keys to Swedish Keys
;
; Example of how to translate keys on the 6400 keyboard
; to country specific keys
; This example is doing a US to Swedish translation
;
; | -> å
; { -> ä
; } -> ö
; \ -> Å
; [ -> Ä
; [ -> Ö
;
; Compile with TASM (Borland Turbo Assembler)
; Link with TLINK   (Borland Turbo Linker)
; Example:
;   tasm kbsv /la
;   tlink kbsv /t
;
;#######################################################

.MODEL    Small
.CODE
assume cs:_TEXT,ds:_TEXT
org    100h
```

```
        start:
                jmp     begin
         intchk db      0               ; Marker for recognition if installed
         temp   dw      0               ; Temporary storage
         extkb  db      00h             ; 'OR'ed with read mode for extended funcs
         jvec   dw      0               ; Vector to original interrupt for wedging
                dw      0               ;  into

        ; Keyboard translate table of scan code plus character,
        ;  followed by the new character

        ktrans:
            dw      2b7ch       ; | -> à
            db      86h
            dw      1a7bh       ; { -> ä
            db      84h
            dw      1b7dh       ; } -> ö
            db      94h
            dw      2b5ch       ; \ -> Å
            db      8fh
            dw      1a5bh       ; [ -> Ä
            db      8eh
            dw      1b5dh       ; ] -> Ö
            db      99h
            dw      0           ; end marker

        inter:
            push    bp          ; Save base
            push    bx          ; and bx registers
            push    dx          ;
            cmp     ah,11h      ; Read of keyboard
            jg      quit        ; If not a read, jump to keyboard routine
            mov     bl,ah       ;
            and     bl,0fh
            cmp     bl,1h       ; Read of keyboard or status ?
            jg      quit        ; If not a read, jump to keyboard routine
            mov     bp,sp       ; Get stack pointer
            add     bp,0ah
            mov     dx,[bp]     ; Get calling function's flag register
            mov     bh,ah       ; Save the current mode
            or      ah,cs:extkb ; Use extended keyboard version
            push    dx          ; Push new flags which aren't affected
                                ; AKA pushf
            call    DWORD PTR cs:jvec    ; Call keyboard routine as if it was
                                ; an int
            pushf
            pop     dx          ; Get the new flags
            mov     [bp],dx     ;  and put them back on the stack
            cmp     al,0e0h     ; The grey keys don't have the prefix 0,n
                                ;  instead they have 0eh,n
            jne     inter2
            cmp     ah,0        ; Alt[224] gives 0eh,0
            je      inter2      ;  pass this back unchanged
```

```
        and      bh,010h           ; If we were called as extended read
        or       bh,bh             ;  then pass this back unchanged
        jnz      inter2
        mov      al,0              ; Otherwise pass back as 0,n

inter2:
        mov      bx,offset ktrans             ; The address of the translate table

inter3:
        cmp      cs:[bx],ax                   ; Compare scan code against table
        jne      inter4                       ; If not the same then loop
        mov      al,cs:byte ptr[bx+2]         ; Alter the key code
        jmp      inter5                       ;  and exit

inter4:
        add      bx,3                         ; Increment pointer to table for next
        cmp      cs:word ptr[bx],0            ; If this is the last entry, then exit
        jne      inter3

inter5:
        pop      dx                ; Exit from the read and
        pop      bx                ;  and restore all registers
        pop      bp
        iret

quit:
        pop      dx                           ; Go to original read routine and
                                              ;  do not return back to here
        pop      bx                           ; Restore base
        pop      bp                           ;  and bx registers
        jmp      DWORD PTR cs:[jvec]          ; Jump to original interrupt

;All code beyond this point is not preserved after exit
;   and remains resident

begin:
        push     ds                ; Preserve data segment
        mov      ax,3516h          ; Interrupt address for read key routine
        int      21h               ; interrogate
        push     es
        pop      ds
        mov      di,offset intchk            ; Position of KBSV marker
        cmp      byte ptr [di],2            ; Is interrupt already present
        jne      intset                    ; If not install it
        pop      ds                ; Restore data segment
        ret                        ; and return to DOS
```

```
intset:
    pop     ds                  ; Restore data segment
    mov     jvec,bx             ; Preserve old interrupt
    mov     jvec+2,es           ;  vector
    mov     dx,offset inter
    mov     ax,2516h            ; Install new interrupt
    int     21h                 ;  vector
    mov     ax,3100h            ; Code for exit and remain resident
    mov     dx,offset begin          ; Length of interrupt routine
    shr     dx,1                ; Convert to
    shr     dx,1                ;  number
    shr     dx,1                ;  of
    shr     dx,1                ;  segments
    inc     dx                  ; Segments+1
    int     21h                 ; Exit and remain resident
    end     start
```

,,   ,,

This Index has been partitioned into three subsections:

General Index — This subsection contains a list of all of the indexes, except for those which are included in the other Index subsections.

Files Index — This subsection provides a good means of locating descriptions for application programs, batch files, drivers, etc., since it is a fairly comprehensive list of all file names described or mentioned in this publication.

Interrupt Index — This subsection contains a list of all of the interrupts defined in this publication, organized alphabetically by topic.

NOTE:

Any page number in this index, listed without a prefix, refers to a page in the introductory section of this publication (the Preface).

NUMBERS

4 key reset, 1-10, 1-12, 6-4

48xx printer drivers, 2-16
   See also NPCP Printing (DOS)

49xx communication controllers, 1-18

3x4x specific display modes, APIs, 4-24

6400 keyboard:
   definitions, 4-12
   redefinitions, 4-15

A

Activity monitors, 3-3

Adapter, communication, 1-4, 1-8, 1-9, 1-10, 1-12, 1-15

Advanced Power Management:
   description, 2-2
   installation, 2-2
   to control power usage, 3-2

Alternate:
   character set, 1-22, 4-24
   window key set, 1-22

Annunciators:
   characters used, 1-22, 4-23
   description, **4-19**
   disappearing, 4-19
   displayed, 2-23
   explanation of display, **4-29**
   interrupts, 4-24, 5-14, 5-16, 5-61
   switches for MISCTSR, 1-22

Anomalies, timeouts, 2-4

APIs:
   3x4x specific display modes, 4-24
   annunciator, 4-19
   backlight, 4-24, 5-21
   character and attribute, 4-24
   contrast, 4-25, 5-19
   cursor mode, 4-25
   display mode, 4-25
   fonts, 4-22, 4-23, 4-25
   graphics dot, 4-25
   pixel map, 4-25
   refresh screen, 4-25
   screen size, 4-22, 4-25, 5-16
   scrolling, 4-25
   video mode, 4-25
   viewport, 4-20, 4-21, 4-22, 4-25

APM BIOS:
   firmware error codes, 3-11
   functions
      overview, 3-9
      summary,
         See NPCP Printing (DOS)
   interface between application and, 2-2, 3-2, 3-6
   interface specification reference, 3-11
   interface specification, document, 7-1
   interrupt definitions, 5-36, 5-41, 5-43, 5-51, 5-66, 5-71
      See also Interrupt Index
   notified when system is busy, 2-2
   overview, 3-2
   PM events communicated from, 3-7
   register interfaces to, 3-9
   residing at lowest layer, 3-2
   software interface, 1-18, 3-6

APM busy, 5-45

APM interrupt extensions, providing, 1-19

Application development:
   equipment setup, 1-12
   minimum configuration, 1-8

Applications, locating, 4

Files Index

This subsection contains a comprehensive list of file names that are listed or defined in various sections in this publication, to help you easily locate information about application programs, drivers, and other files.

NOTE:

Any page number in this index, listed without a prefix, refers to a page in the introductory section of this publication (the Preface).

A highlighted page number indicates a description for the filename found on that page (at least an important tidbit of information).

„   „

This subsection contains a complete list of interrupts defined within this publication, and 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 have been listed only by noun.

NOTE:

Any page number in this index, listed without a prefix, refers to a page in the introductory section of this publication (the Preface).

NUMBERS

3x4x Select Specific Display Modes:
    Set Chase Mode, INT 10h, 5-17
    Set Fixed Mode, INT 10h, 5-17
    Set Graphics Mode, INT 10h, 5-18
    Set Setparms Mode, INT 10h, 5-18

A

A20:
    Query A20 Gate Support, INT 15h, 5-41
Absolute:
    Get Contrast (absolute, no compensated), INT 10h, 5-19
    Set Contrast (absolute, no compensated), INT 10h, 5-20
Access:
    Disable Access to RAM Drive Memory, INT 13h, 5-35
    Enable Access to RAM Drive Memory, INT 13h, 5-35
Active:
    Scroll Window Down (active page), INT 10h, 5-9
    Scroll Window Up (active page), INT 10h, 5-9
    Set Active Display Page, INT 10h, 5-8
Alarm:
    Read the Real-Time Clock Alarm, INT 1Ah, 5-70
    Reset the Real-Time Clock Alarm, INT 1Ah, 5-69
    Set the Real-Time Clock Alarm, INT 1Ah, 5-69
Alternate Functions:
    (see individual functions, by topic)
Annunciator:
    Enable/Disable Annunciator Panning, INT 10h, 5-14
    Reset Annunciator Line to Current Keyboard Settings, INT 10h, 5-14
    Write BIOS Annunciator Values (with attribute), INT 10h, 5-16
    Write BIOS Annunciator Values (without attribute), INT 10h, 5-16
APM:
    APM Installation Check, INT 15h, 5-43
    APM Interface Disconnect, INT 15h, 5-44

    APM Real Mode Interface Connect, INT 15h, 5-44
ASCII:
    Read Next ASCII Character from Keyboard, INT 16h, 5-61
Attribute:
    Read Character and Attribute at Current Cursor Position, INT 10h, 5-10
    Write BIOS Annunciator Values (with attribute), INT 10h, 5-16
    Write BIOS Annunciator Values (without attribute), INT 10h, 5-16
    Write Character (no attribute) at Current Cursor Position, INT 10h, 5-11
    Write Character and Attribute at Current Cursor Position, INT 10h, 5-10

B

Backlight:
    Read Backlight Intensity, INT 10h, 5-22
    Read Backlight Timer, INT 10h, 5-23
    Refresh Backlight, INT 10h, 5-23
    Set Backlight Intensity, INT 10h, 5-22
    Set Backlight Timer, INT 10h, 5-23
    Toggle Backlight, INT 10h, 5-22
    Turn Backlight Off, INT 10h, 5-21
    Turn Backlight On, INT 10h, 5-21
Bar Code:
    Read Bar Code Data, INT 7Ah, 5-76
    Set Bar Code Options, INT 7Ah, 5-72
Beep:
    Beep the Buzzer, INT 15h, 5-41
    Beep the Buzzer, INT 16h, 5-65
    Get Error Beep Parameters, INT 16h, 5-64
    Set Error Beep Parameters, INT 16h, 5-63
    Sound Error Beep, INT 16h, 5-64
BIOS:
    Keyboard BIOS Services, Interrupt 16h, 5-61
    Return Extended BIOS Data Area Segment (EDBA), INT 15h, 5-60
    System BIOS Services, Interrupt 15h, 5-40
    Write BIOS Annunciator Values (with attribute), INT 10h, 5-16
    Write BIOS Annunciator Values (without attribute), INT 10h, 5-16
Block:
    Copy Extended Memory Block, INT 15h, 5-55
Buffer: