phyCORE-Z500PT

Atom Z500-based Module

BIOS Dokumentation

Ein Produkt eines Unternehmens der PHYTEC Technologie Holding AG

In this manual are descriptions for copyrighted products that are not explicitly indicated as such. The absence of the trademark (™) and copyright (©) symbols does not imply that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual.

The information in this document has been carefully checked and is believed to be entirely reliable. However, PHYTEC Messtechnik GmbH assumes no responsibility for any inaccuracies. PHYTEC Messtechnik GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. PHYTEC Messtechnik GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages which might result.

Additionally, PHYTEC Messtechnik GmbH offers no guarantee nor accepts any liability for damages arising from the improper usage or improper installation of the hardware or software. PHYTEC Messtechnik GmbH further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

	EUROPA	NORD AMERIKA
Adresse:	PHYTEC Technologie Holding AG Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Angebots Hotline:	+49 (800) 0749832 order@phytec.de	+1 (800) 278-9913 sales@phytec.com
Technische Hotline:	+49 (6131) 9221-31 support@phytec.de	+1 (800) 278-9913 support@phytec.com
Fax:	+49 (6131) 9221-33	+1 (206) 780-9135
Web Seite:	http://www.phytec.de	http://www.phytec.com

1st Edition November 2009

Index of Figures

1 BIOS INTRODUCTION

1.1 Introducing Embedded BIOS for the phyCORE-Z500PT

To be truly compatible with today's PC architecture, a BIOS must offer all the external software interfaces and certain internal nuances of the first IBM PC, IBM PC/XT, and IBM PC/AT BIOS implementations, as well as support the industry initiatives and operating systems that were introduced since those early days of the Personal Computer. Embedded BIOS does this and more, building on a decade of support for these initiatives on the widest possible range of chipsets, CPUs, and board designs.

2 POWER ON SELF TEST

2.1 About POST

When your system is powered on, Embedded BIOS tests and initializes the hardware and programs the chipset and other peripheral components. This phase is called Power-On Self-Test, or POST for short. This chapter describes POST and how to interact with it as a user. Configuration of POST is described in detail in the System Configuration Utility chapter.

2.2 POST Codes

POST is the process that configures the system as quickly as possible to provide computing facilities to the user. During POST, thousands of CPU, chipset, Super I/O, and device registers must be configured and inspected with precise protocols. When the hardware fails to respond as expected, POST may not be able to continue; for example, if the memory controller or memory itself cannot be configured, the system cannot continue to initialize the display because the BIOS-level display driver (known as the Video BIOS, supplied by the silicon vendor) requires memory to work properly. Therefore there are cases where POST cannot continue, but has not performed all of the steps necessary to make the user interface (keyboard and screen) ready for use. Embedded BIOS provides support for these early POST failure conditions.

During POST, special 2-digit hexadecimal-encoded values are written to a special I/O port at address 80h. This I/O port may be monitored by the 4-digit display on the motherboard. If POST determines that your system is unhealthy, yet has not initialized the system sufficiently to be able to display messages to the screen, then these POST codes can be an effective way of determining the last action taken by POST (i.e., “where POST stops”.) If your system stops during POST, you can use a POST code monitor as a diagnostic tool to look up a potential reason for the system stop in Appendix A, which provides a list of POST codes and their meanings. Please note that

while Embedded BIOS POST codes are listed in this appendix, other devices in the system, such as add-in network cards, video option ROMs, and even some operating systems, may also output codes to your POST code display

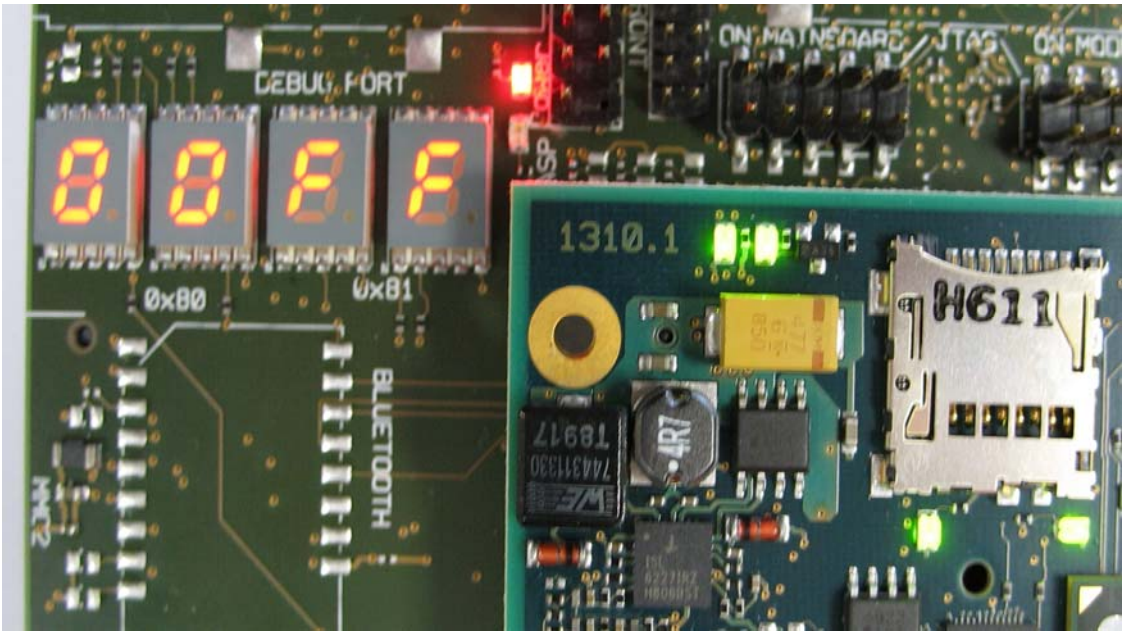

Figure 1: 4-digit POST code display

2.3 The POST User Interface

Depending on the kind of equipment being initialized by Embedded BIOS, POST may interact with the user in a variety of different ways. These are all configurable from the Preboot System Configuration Utility (Setup).

2.3.1 The POST User Interface

Depending on the kind of equipment being initialized by Embedded BIOS, POST may interact with the user in a variety of different ways. These are all configurable from the Preboot System Configuration Utility (Setup).

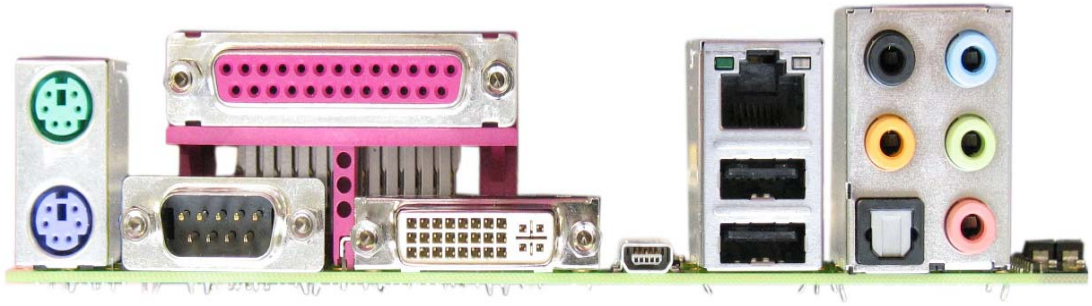

Figure 2: Connectors for PS/2 keyboard/mouse, video, serial port, parallel port, and USB.

POST comes up with a graphical splash screen as being shown by Figure 3

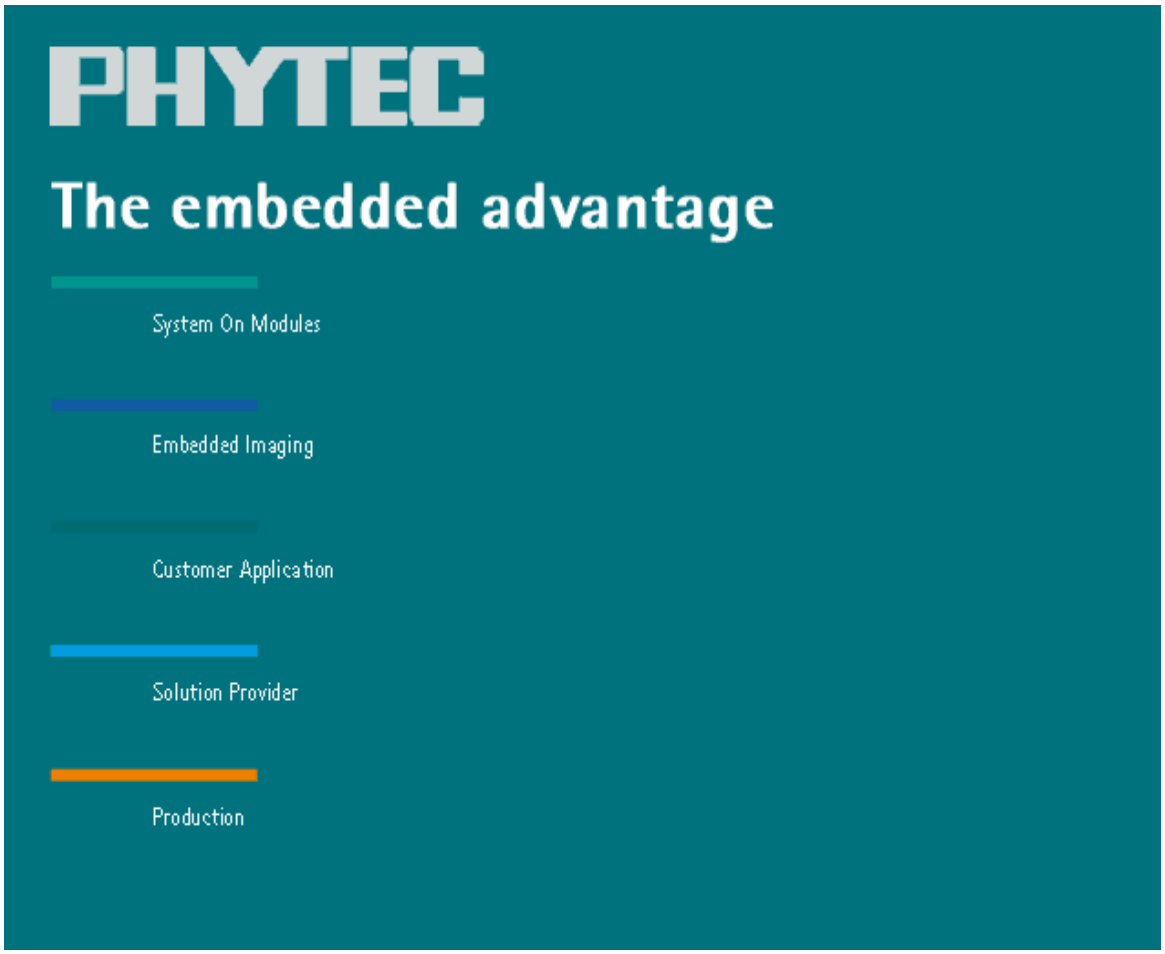

Figure 3: Graphical POST Showing Splash Screen.

2.3.2 Console Redirection

POST has a feature called Console Redirection, which causes text display to be redirected from the video screen to a serial port. The user connects a null modem serial cable between the system and another computer system (we'll call the latter, the "host" system), and uses a terminal emulation program such as Windows HyperTerminal, TeraTerm, DOS Navigator, Telix, PROCOMM, or any number of other programs widely available, to emulate the video display. Parameters are 115000 8N1.

Figure 4 shows a screen shot of Console Redirection being used to display the character-based POST.

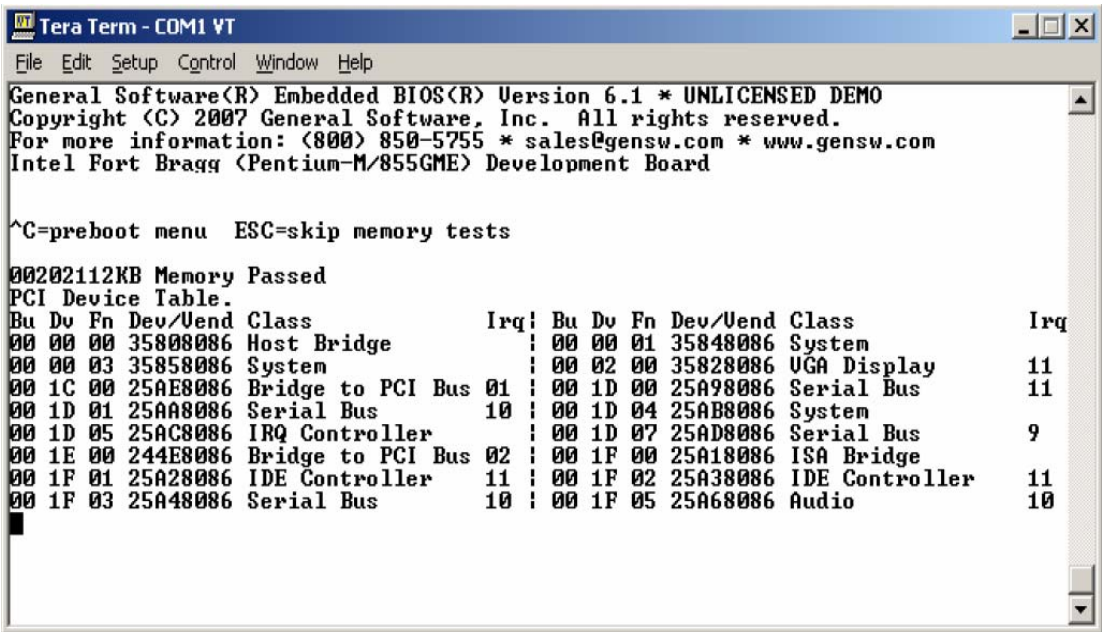

Figure 4: Terminal emulator being used on a host to act as a redirected console for POST.

With Console Redirection, the host's keyboard is used to type, and these keystrokes are sent by the terminal emulation program over the serial cable in the other direction to POST, which receives the keystrokes as though they came from the PS/2 or USB keyboard in the system.

Because POST's Console Redirection feature is configured to Automatic detection of a user on the configured serial port, POST begins using the serial port only if it detects a keystroke from the user through the terminal emulation program. Therefore, when Console Redirection is desired, press a key (any key, including an unrecognized one such as the space bar) to begin using the remote console link. If the character is one of the special POST characters, it is processed appropriately (see nect chapter).

2.4 POST User Intervention

With no typing on the part of the user, POST performs the activities as configured in the POST setup screen menu, and then begins processing the boot actions (typically, booting the operating system.)
While POST ignores unrecognized keystrokes, certain keystrokes have special meaning to POST and can direct it to perform special functions:

- B / ^B keys – Enter the BBS Boot Menu, allowing interactive selection of boot action to be taken, rather than the set of boot actions configured in the BOOT Setup menu.
- C / ^C / <Del> keys – Either of these keystrokes cause the Preboot Menu to be invoked (described later in this chapter.)
- T / ^T keys – Perform OEM-specific preboot tests on the system.

Thus in case of using Console Redirection, it is common for a serial port user to press ^C a few times in succession to get the attention of POST, to enter the Preboot Menu and gain access to its services over a serial port connection.

2.5 Configuring POST

When the system is powered on for the first time, you'll need to configure the system through the System Setup Utility (described in the next chapter) before peripherals, such as disk drives, are recognized by the BIOS. The information is written to non-volatile storage (sometimes Flash, or battery-backed CMOS RAM.)

The system's System Setup Utility is a comprehensive setup screen system that provides ways to configure POST so it performs the activities that the user needs every time the system boots. The most important menus to consider when configuring POST are given below. See the System Setup Utility chapter for details about how to configure each item.

- POST Menu
  - Configure memory test strength and whether the memory test clears memory.
  - Request the system to pause if POST finds errors so the user can see the messages.
  - Enable or disable POST messages by category, such as PnP or PCI tables.
  - Perform automated rebooting for factory burn-in Q/A cycles.
- BOOT Menu
  - Build list of boot devices for POST to configure and attempt to boot from.
  - Select floppy drive hardware types (1.44MB, 720KB, 1.2MB, 360KB, etc.)
  - Select PATA drive cable types (40-pin, 80-pin, autodetection.)
  - Select PATA UDMA modes authorized by user.
- Firmbase Technology Menu
  - Enable USB HID and USB Boot components supporting USB keyboards, mice, and disk drives.

3 SYSTEM SETUP UTILITY

3.1 The System Setup Utility

The system is configured with the System Setup Utility, accessible from the Preboot Menu. For information about how to enter the Preboot Menu from power on, read Chapter 2.

The System Setup Utility is highly configurable by the OEM, and may omit features shown here, or may include OEM-proprietary features not shown here. However, it is likely that your system will include the main setup screens having to do with basic system configuration.

Setup may be run from either the main keyboard and video display, or from a terminal emulator program running on a host computer connected to the system through a serial cable. See Chapter 5 for a description of how to use the Console Redirection feature.

3.2 Setup Menus

The standard Embedded BIOS setup menus are described below in the order they generally appear in the menuing system:

Main	Display main system components and allow editing of date and time.
Exit	Save changes and exit, discard changes and exit, or restore factory default settings.
Boot	Configure boot actions and boot devices.
POST	Configure POST.
PnP	Configure Plug-n-Play for non-ACPI OSes.
Features	Enable and disable system BIOS features like ACPI, APM, PnP, MP, quick boot, and the splash screen.
Firmbase	Configure Firmbase Technology and the features that use it, such as USB keyboard and mouse support (commonly, USB HID) and boot from USB (commonly, USB Boot).
Misc	Configure miscellaneous BIOS settings that do not fall into any other category.
Video	Configure display device parameters.
Chipset	Configure any chipset-specific parameters, such as memory, CPU, and bus timing, and availability of chipset-specific features such as TFT support. Highly platform-specific and entirely up to the OEM's implementation.
AdvancedCPU	Display CPU relevant informations.

3.3 Navigating Setup Menus and Fields

Navigation (moving your cursor around, selecting items, and changing them) is easy in the Setup system. The following chart is a helpful user reference:

UP key (also ^E)	Move the cursor to the line above, scrolling the window as necessary.
DOWN key (also ^X)	Move the cursor to the line below, scrolling the window as necessary.
LEFT key	Go back to the menu to the left of the currently-displayed menu in the menu bar.
RIGHT key	Go forward to the menu to the right of the currently-displayed menu in the menu bar.
PGUP key	Move the cursor up several lines (a full window's worth), scrolling the window as necessary.
PGDN key	Move the cursor down several lines (a full window's worth), scrolling the window as necessary.
HOME key	Move the cursor to the first configurable field in the current menu, scrolling the window as necessary.
END key	Move the cursor to the last configurable field in the current menu, scrolling the window as necessary.
ESC key	Exit the Setup system, discarding all changes (except date/time changes, which take place on-the-fly.)
TAB key	Move the cursor down to the next configurable field.
Shift-TAB key (backtab)	Move the cursor up to the last configurable field.
+ key	Toggle an Enable/Disable field, or increase a numeric field's value.
- key	Toggle an Enable/Disable field, or decrease a numeric field's value.
SPACE key	Toggle an Enable/Disable field.
BKSP key	Reset an Enable/Disable or multiple-choice field, or back-up in numeric or string fields.
Digits (0-9)	Used to enter numeric parameters.
Alphabetic (A-Z, a-z)	Used to enter text data on ASCII fields such as email addresses.
Special symbols (!@#$%^&*_-+={}[], etc.)	Used to enter special text on ASCII fields thatpermit these characters.

The basic idea when using the Setup system is to navigate to the menus containing fields you want to review, and change those fields as desired. When your settings are complete, navigate to the EXIT menu, and select “Save Settings and Restart”. This causes the settings to be stored in nonvolatile memory in the system, and the system will reboot so that POST can configure itself with the new settings.

After rebooting it may be desirable to reenter the Setup system as necessary to adjust settings as necessary.

Once the system boots, the Setup system cannot be entered; this is because the memory used by the BIOS configuration manager is deallocated by the system BIOS, so that it can be used by the OS when it boots. To reenter the Setup system after boot, simply reset the system or power off and power back on.

3.4 Main Setup Menu

The first menu always showing in the Setup system is the Main menu. This menu is shown in Figure 5 below.

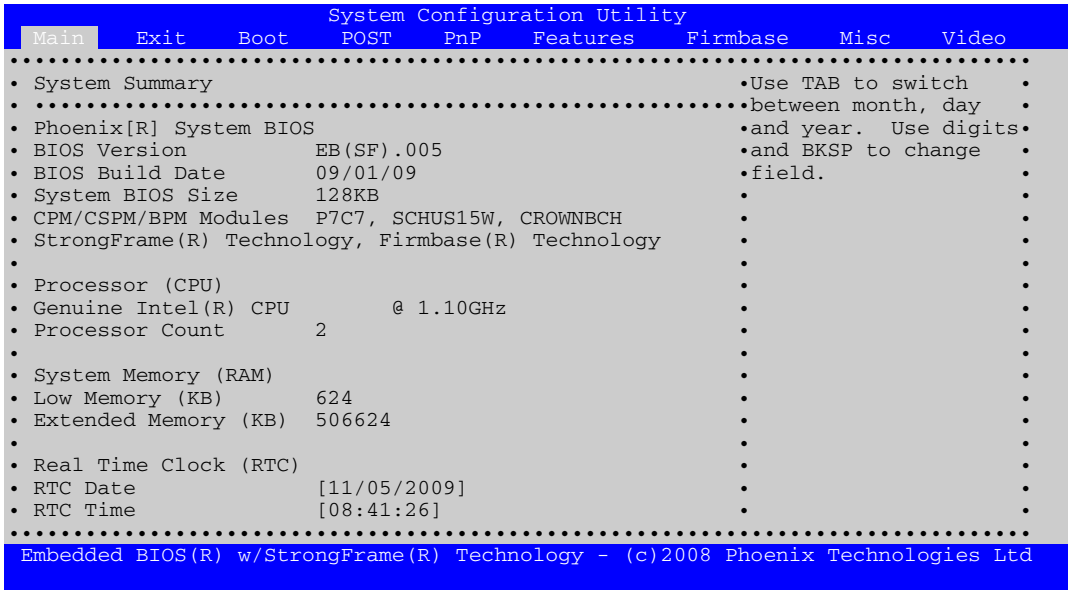

Figure 5: Main Setup Menu.

The Main menu provides a system summary about the BIOS, processor, system memory, date and time, and any other items configured by the OEM.

The BIOS information is obtained by Setup from the internal system BIOS build itself; this information is useful when obtaining support for your system.

BIOS Version	Indicates the major and minor core architecture versions (6.x, where x is a number from 0 to 999.)
BIOS Build Date	Date in MM/DD/YY format on which the OEM built the system BIOS binary file.
System BIOS Size	Size of BIOS exposed in low memory below the 1MB boundary. Commonly, 128KB would mean that the BIOS is visible in the address space from E000:0000 to F000:FFFF.
CPM/CSPM/BPM Modules	Indicates the names of the key architectural modules used to create the system BIOS binary file. The CPM module provides the CPU family support; the CSPM module provides the northbridge support; and the BPM module provides the board-level support.

The CPU information is normally obtained by querying the Processor Brand String in the CPU's MSRs; the method used to achieve this is beyond the scope of this document.

The system memory information does not describe physical RAM; rather it describes the RAM as configured, subtracting RAM used for System Management Mode, Shadowing, Video buffers, and other uses. This provides realistic values about how much memory is actually available to operating systems and applications.

The Real Time Clock fields are editable with keystrokes. To navigate through the MM/DD/YYYY and HH:MM:SS fields, use the TAB and BACKTAB keys. The hours are normally specified in military time; thus 13 means 1pm, or one hour after noon, whereas 01 means 1am, or one hour after midnight. When the cursor leaves RTC fields, they either affect the battery-backed RTC right away, allowing the system

to continue with your new settings, or they revert back to old values if the new values are not valid entries.

3.5 Exit Setup Menu

The Exit menu provides methods for saving changes made in other menus, discarding changes, or reloading the standard system settings. This menu is shown in Figure 6 below.

Figure 6: Exit Setup Menu.

To select any of these options, position the cursor over the option and press the ENTER key. Pressing the ESC key at any time within the Setup system is equivalent to requesting “Exit Setup Without Saving Changes.”

All three options request verification before performing the selected action, otherwise, the system configuration might be saved or lost by accident. Figure 7 illustrates the verification popup for saving and exiting; the other options are similar.

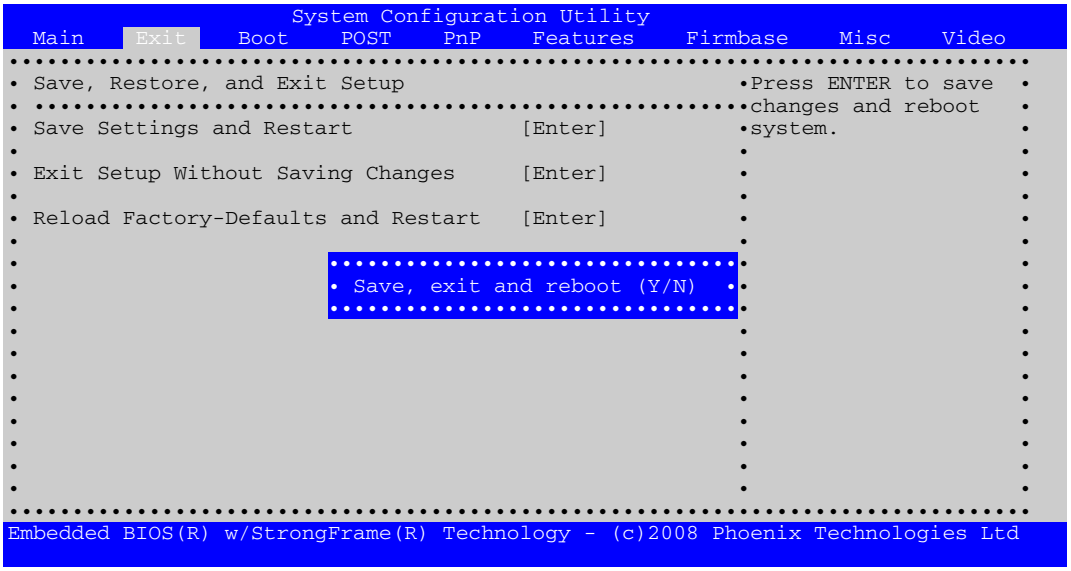

Figure 7: Popup requesting verification before exiting Setup system.

3.6 Boot Setup Menu

The Boot menu allows the system's boot actions and boot devices to be configured. This menu is shown in Figure 8.

The BBS portion of this menu lists the devices and activities to be performed in the order in which they appear in the list. When the BIOS completes POST, it follows this list, attempting to process each item. Some items are drives, such as an ATA/IDE drive, or a USB hard disk, or CDROM.

The ordering of the drives in the BBS list controls the BIOS in several ways. First, it is the list of drives that is scanned and assigned BIOS unit numbers for DOS (0, 1, 2 for floppy-type devices, and 80h, 81h, 83h, and so on for hard drives.) If a drive on the list is not plugged in or working properly, the BIOS moves on to the next drive, skipping the inoperative one.

Second, once the drives in the list have been verified, POST attempts to boot from them in that order as well. Drives without bootable partitions might be configured, but skipped over in the boot phase, so that other drives on the list become candidates for booting the OS.

The BBS list also contains other boot actions, such as boot from network cards and PCI slots. When deciding what boot action to do first and then next in succession, POST first scans all the drives in the list to verify they are present and operating properly (as described earlier in this section) and then goes down the list and tries to perform the actions in order. During this boot phase, if the list item is a drive, an attempt is made to boot from the boot record of that drive. If the list item is a device like a network card or PCI slot, an attempt is made to boot from that device. If the list item is a software item like "Boot Debugger", then it performs that action, and when that action completes, it moves on to the next item in the BBS list.

The table that follows lists the set of standard boot action items (which is entirely configurable by the OEM; you may see many more or fewer choices depending on how your system's BIOS has been adapted):

"drive name" – The system BIOS may list the drive's name in a generic sense (i.e., "USB Hard Drive") if the drive has not been detected yet, or the drive's full manufacturing name and serial number (if detected.)	Boot from the MBR/PBR of the named BIOSaware IPL drive (BAID). The drive may be Legacy Floppy, PATA, SATA, Compact Flash, or a USB drive.
IDE0/Primary Master	Primary Master PATA drive or SATA mapping by the chipset.
IDE1/Primary Slave	Primary Slave PATA drive or SATA mapping by the chipset.
IDE2/Secondary Master	Secondary Master PATA drive or SATA mapping by the chipset.

IDE3/Secondary Slave	Secondary Slave PATA drive or SATA mapping by the chipset.
IDE CDROM	First detected IDE CDROM.
USB Floppy Drive	First detected USB floppy drive.
USB Hard Drive	First detected USB hard drive.
USB CDROM Drive	First detected USB CDROM.
Enter BIOS Setup Screen	Invoke System Setup Utility in ROM.
Reboot System	Restart system.
Network	Boot from any network adapter.

Figure 8 below illustrates a common setup of the BBS list for desktop applications. In this example, the first and only boot device is the Seagate Technologies hard drive connected to the target as a Primary Master IDE drive. A second boot device, “None”, is a placeholder that is simply used to add more entries in the setup screen; “None” is not actually executed by POST as a boot action item.

Figure 8: Simple BBS configuration boots to a hard drive

In addition to the BBS boot device list, there is one more section in the BOOT menu; namely the IDE Drive Configuration sections. The IDE Drive Configuration section describes the drive geometry detection and the fastest access mode.

3.7 POST Setup Menu

The POST menu is used to configure POST. This menu is shown in Figure 9. Be sure to review the Features menu, where additional items can be configured, such as the Splash Screen and BIOS initiatives.

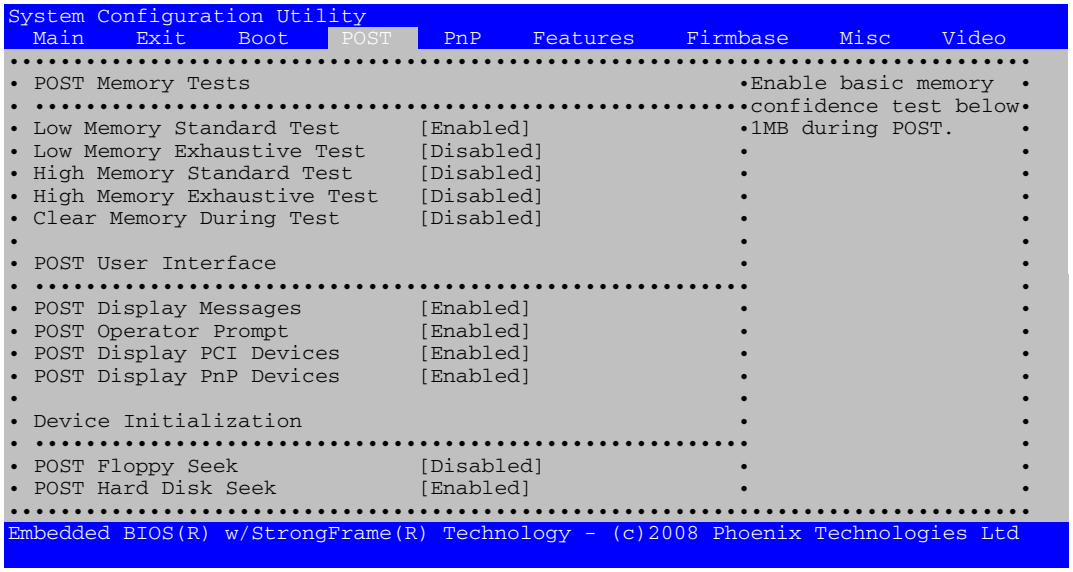

Figure 9: POST Setup menu.

The following table describes the settings associated with the POST setup menu's Memory Test section.

Low Memory Standard Test	Enable basic memory confidence test, of memory below 1MB address boundary (conventional memory, or memory normally used by DOS.)
Low Memory Exhaustive Test	Enable exhaustive memory confidence test of memory below 1MB address boundary.
High Memory Standard Test	Enable basic memory confidence test, of memory between 1MB and 4.2GB address boundaries (extended memory.)
High Memory Exhaustive Test	Enable exhaustive memory confidence test, of memory between 1MB and 4.2GB address boundaries.
Clear Memory During Test	Enable storing 0's in all memory locations tested. Only necessary when some legacy DOS programs are run, as they may rely on cleared memory to operate properly.

The following table describes the settings associated with the POST setup menu's POST User Interface section:

POST Display Messages	Enable display of text messages during POST. When disabled, POST is "quiet."
POST Operator Prompt	Enable operator prompts if POST is configured to ask interactive questions of the user about whether to load specific features; i.e., whether or not to load SMM.
POST Display PCI Devices	Enable display of PCI devices.
POST Display PnP Devices	Enable display of ISA PnP devices.

The following table describes the settings associated with the POST setup menu's Device Initialization section:

POST Floppy Seek	Enable head seek on each floppy drive configured in the system. Used to recalibrate the drive in some systems with older DOS operating systems.
POST Hard Disk Seek	Enable head seek on each hard drive configured in the system. This is a way of extending the standard testing performed on each drive during POST, by requesting that the drive actually move the head. Not available with all drives.

3.8 PnP Setup Menu

The PnP menu is used to configure Plug-n-Play, a legacy BIOS initiative used to support operating systems such as Windows95, Windows98, and WindowsNT. ACPI has largely replaced this feature; however, it is necessary for platforms to support older operating systems. Figure 10 shows the PnP Setup menu.

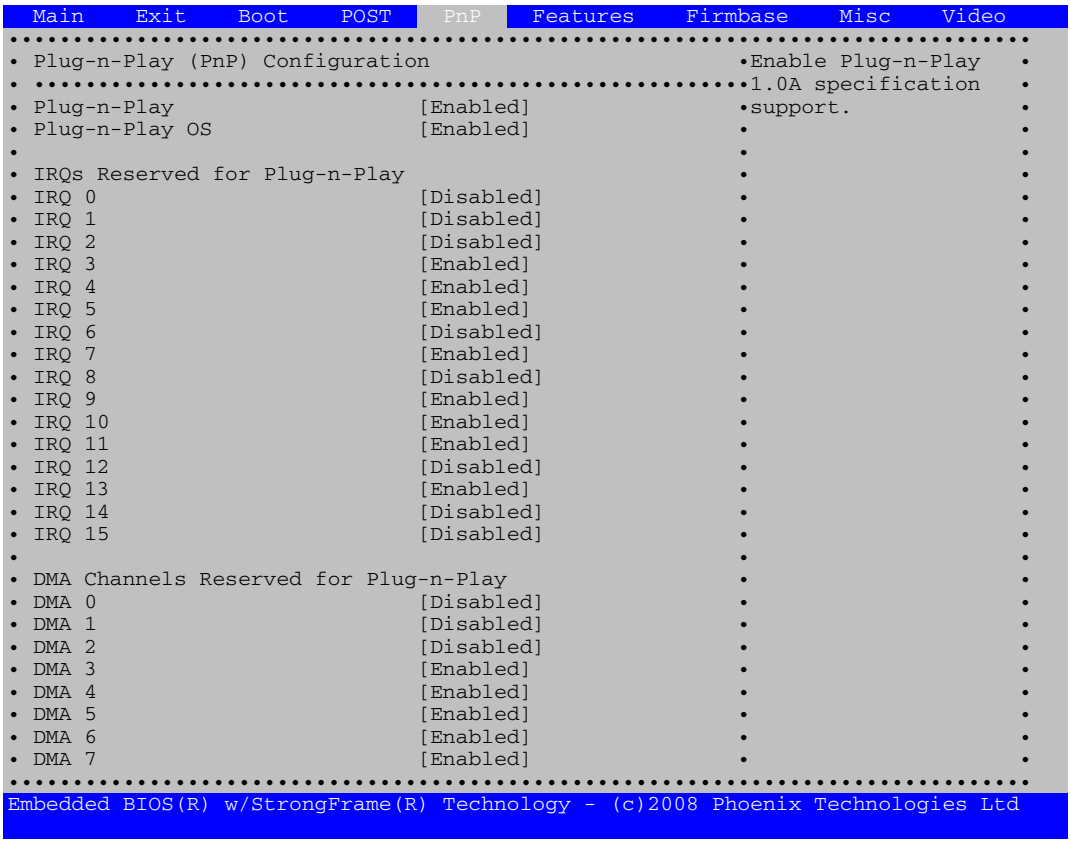

Figure 10: PnP Setup menu.

The PnP menu consists of two sections; basic configuration that enables Plug-n-Play and identifies if a PnP should perform configuration or let the OS do it; and then, another section that defines which system IRQs should be reserved for PnP's use, so that PCI doesn't use them.

The following table presents the fields in the PnP menu.

Plug-n-Play	Enable PnP feature. When disabled, a PnPaware OS will not find any PnP services in the BIOS, and all other configuration parameters in the menu will be greyed out. Enable to support legacy OSes like DOS, Windows95, Windows98, and WindowsNT. Disable for operating systems like WindowsXP or Windows Vista, or for Linux operating systems with ACPI support.
Plug-n-Play OS	Enable delay of configuration of PnP hardware and option ROMs. When enabled, BIOS will NOT configure the devices, and instead defer assignment of resources, such as DMA, I/O, memory, and IRQs, to the PnP OS. When disabled, the BIOS performs conflict detection and resolution, and assigns resources for the OS. Disable this parameter when running non-PnP OSes like DOS. Enable this parameter when running PnP OSes like Windows95, Windows98, and WindowsNT.
IRQ0	Enable exclusive use of IRQ0 by PnP.
IRQ1	Enable exclusive use of IRQ1 by PnP.
IRQ2	Enable exclusive use of IRQ2 by PnP.
IRQ3	Enable exclusive use of IRQ3 by PnP.
IRQ4	Enable exclusive use of IRQ4 by PnP.
IRQ5	Enable exclusive use of IRQ5 by PnP.
IRQ6	Enable exclusive use of IRQ6 by PnP.
IRQ7	Enable exclusive use of IRQ7 by PnP.
IRQ8	Enable exclusive use of IRQ8 by PnP.
IRQ9	Enable exclusive use of IRQ9 by PnP.
IRQ10	Enable exclusive use of IRQ10 by PnP.
IRQ11	Enable exclusive use of IRQ11 by PnP.
IRQ12	Enable exclusive use of IRQ12 by PnP.
IRQ13	Enable exclusive use of IRQ13 by PnP.
IRQ14	Enable exclusive use of IRQ14 by PnP.
IRQ15	Enable exclusive use of IRQ15 by PnP.
DMA 0	Enable exclusive use of DMA 0 by PnP.
DMA 1	Enable exclusive use of DMA 1 by PnP.
DMA 2	Enable exclusive use of DMA 2 by PnP.
DMA 3	Enable exclusive use of DMA 3 by PnP.
DMA 4	Enable exclusive use of DMA 4 by PnP.
DMA 5	Enable exclusive use of DMA 5 by PnP.
DMA 6	Enable exclusive use of DMA 6 by PnP.
DMA 7	Enable exclusive use of DMA 7 by PnP.

3.9 Features Setup Menu

The Features menu is used to configure the system BIOS' major features, including Quick Boot, APM, ACPI, PMM, SMBUS, SMBIOS, Manufacturing Mode, Splash Screen, Console Redirection, and others added by the OEM.

Figure 11 shows a typical Features Setup menu.

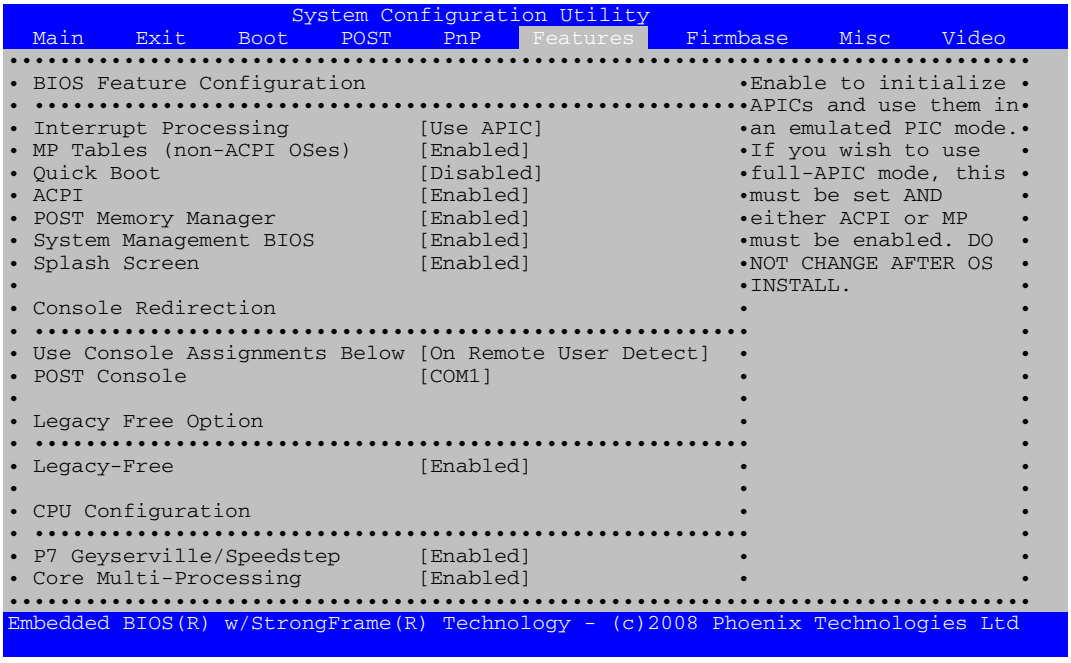

Figure 11: Features Setup menu.

The following table describes each setting in the Features menu (some may not be pictured in the screen shot, even though they may be enabled by the OEM.)

Interrupt Processing	APICs and use them in Interrupt Processing an emulated PIC mode. If you wish to use full-APIC mode, this must be set AND either ACPI or MP must be enabled. DO NOT CHANGE AFTER OS INSTALL.
MP Tables	Enable to provide OSes with APIC and processor info, according to the MultiProcessor Specification. This feature requires the use of APICs. DO NOT CHANGE AFTER OS INSTALL.
Quick Boot	Enable time-optimized POST, causing certain preconfigured OEM optimizations to be made when the system boots.
ACPI	Enable ACPI system description and power management (ACPI replaces PnP and APM.) Used with ACPI-aware OSes such as Linux kernels version 2.6 and above, Windows XP, and Windows Vista. Commonly also uses the SMM feature (see Firmbase) to operate properly.
POST Memory Manager (PMM)	Enable memory allocation services for option ROMs, especially network cards running PXE. Some option ROMs may use this interface incorrectly, causing system crashes. Other PXE option ROMs may not run if PXE is not supported. Because of the state of these option ROMs, the setting is provided as an option to the user.
System Management BIOS (SMBIOS)	Enable System Management BIOS interface specification support, exposing information about the type of hardware, including the chassis, motherboard layout, type of CPU and DRAM sticks, to applications such as WfM, which runs on PXE in the preboot environment.
Splash Screen	Enable graphical POST, including animation, sound, icons, advertisements, and other multimedia objects that may be configured by the OEM.
Console	Configure the console redirection feature over a

Redirection	serial port. Automatic – causes POST, the debugger, and the preboot environment to use the system's first serial port (COM1) when an RS232 cable is detected with DSR and CTS modem signals active, indicating a terminal emulation program is likely to be attached to the other end of the cable. Always – causes the BIOS to always use the serial port as the console, without testing for the presence of the terminal emulation program. Never – causes the BIOS to never invoke console redirection, but instead always use the main keyboard and video display. If there is no keyboard or video display, the system operates headless.
POST Console	Select redirected console for POST/DOS
Legacy Free	Enable Support for Microsoft Legacy-Free specification.
P7 Geyserville Speedstep	Enable to set Geyserville/Speedstep processors to full speed.
Core Multi-Processing	When disabled, the second execution core will not be visible to software and cannot be started via a SIPI message.

3.10 Firmbase Setup Menu

The Firmbase menu configures the Firmbase Technology component of the system BIOS, including all of the features enabled by it; i.e. boot from USB devices (see Figure 12 below).

This menu is highly configurable by the OEM who may elect to eliminate some of the Firmbase Technology tuning parameters in more fixed-function devices.

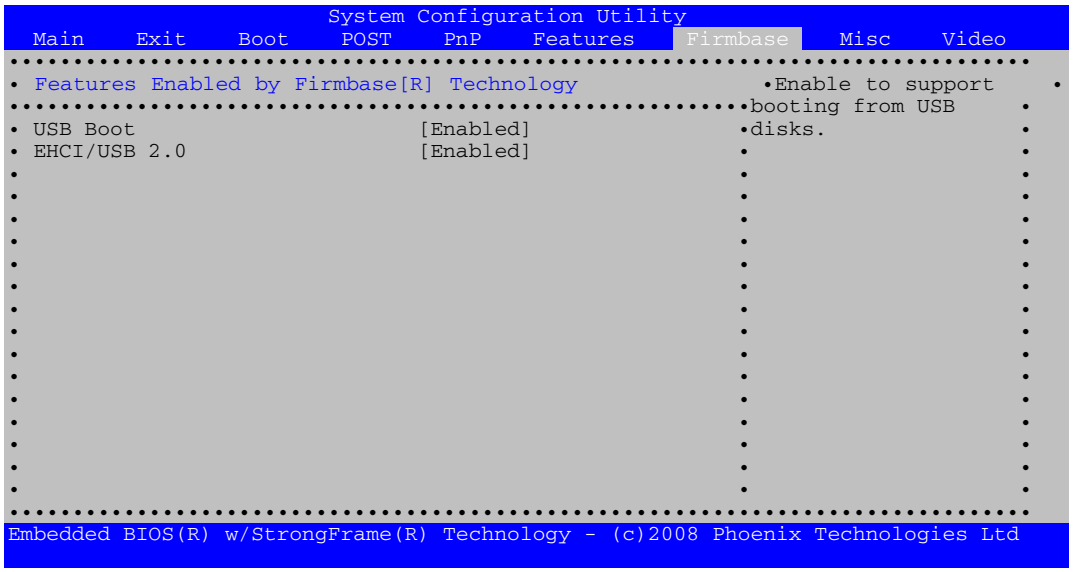

Figure 12: Firmbase Setup menu.

The following table presents the settings that enable high-level features enabled by Firmbase Technology (see Figure 15.)

USB Boot	Enables BIOS support for accessing USB mass storage devices and emulating legacy floppy, hard drive, and CDROM drive devices with them. Enable this option in order for USB devices to be supported in the BBS device list (see the BOOT menu.)
EHCI	Enable EHCI (USB 2.0) driver.

3.11 Miscellaneous Setup Menu

The Misc menu provides for configuration of BIOS settings that don't easily fit in any other category. They include Keyboard Control and Debugger Settings. Figure 13 shows the Misc Setup menu.

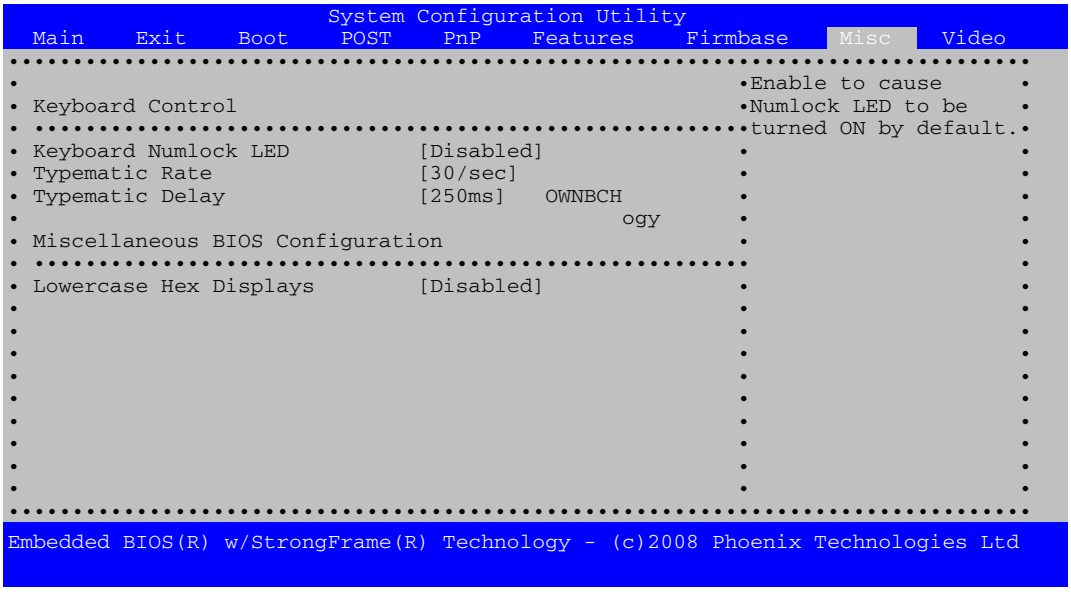

Figure 13: Miscellaneous Setup Menu

The following table presents the settings in the Misc Setup menu.

Keyboard Numlock LED	Enables the Numlock key when POST initializes the PS/2 keyboard. Typematic Rate Specify the rate at which the PS/2 keyboard controller repeats characters when most keys are pressed down. USB typematic is automatic and does not use this parameter. Typematic Delay Specifies the amount of time a repeating key may be pressed on a PS/2 keyboard until the key repeat feature begins repeating the keystroke. USB typematic is automatic and does not use this parameter.
Typematic Rate	Select keyboard repeat rate.
Typematic Delay	Select time BIOS waits to start repeating after keypress.
Lowercase Hex Displays	Enables the display of hexadecimal numbers in the debugger with lowercase letters instead of uppercase letters (ie, 2f8ah instead of 2F8AH.)

3.12 Video Setup Menu

The Video Setup menu configures graphics settings for the attached Display.
Figure 14 shows the Video Setup menu.

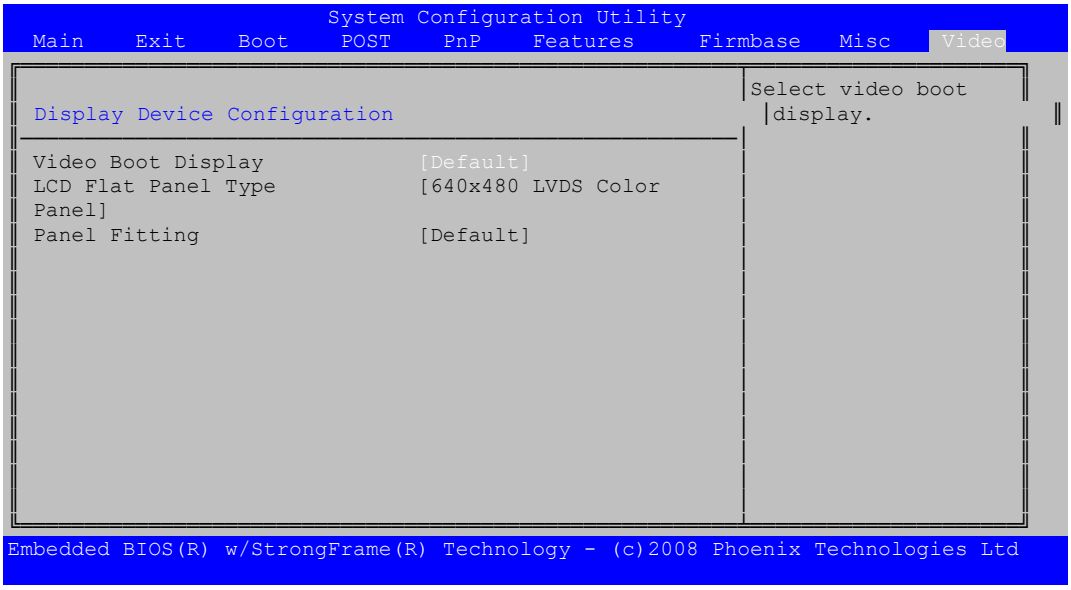

Figure 14: Video Setup Menu

The following table presents the settings in the Video Setup menu.

Video Boot Display	Select video boot display.
LCD Flat Panel Type	Select the flat panel type.
Panel Fitting	Select how to handle a displayed image that is smaller than the native panel resolution.

3.13 Chipset Setup Menu

The Chipset Setup menu enables the user to configure chipset-specific parameters. This menu is completely proprietary and may contain any number of parameters that, in the OEM's belief, are necessary for the successful operation of the system by its users. On desktop systems, these parameters commonly allow advanced users to overclock the system beyond its published limits on an AS-IS basis. In some cases, these parameters are supplied so as to provide support personnel with the tools necessary to diagnose user problems in the field over the telephone.

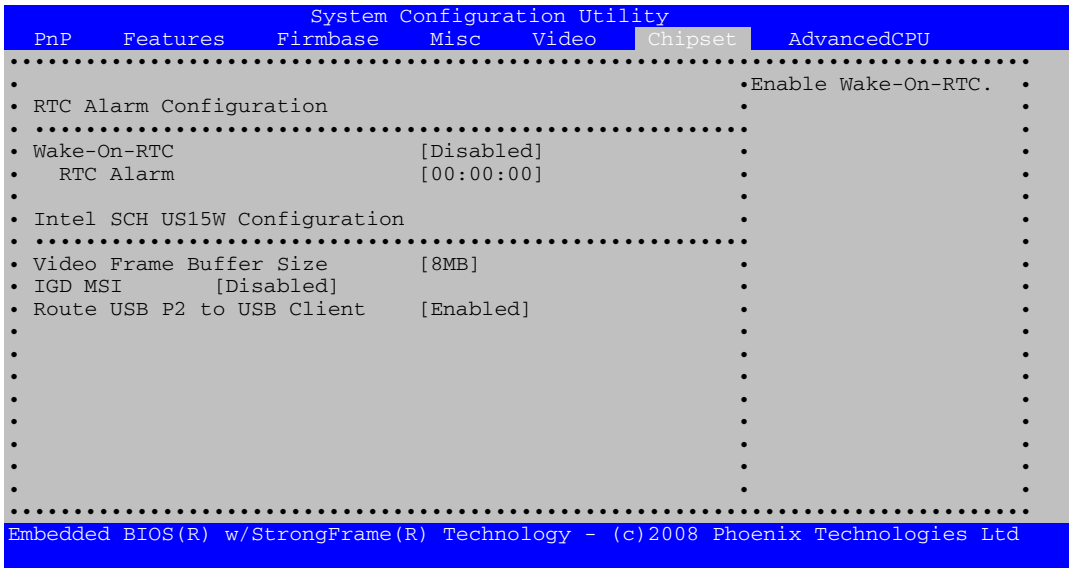

Figure 15: Chipset Setup Menu

The following table presents the settings in the Video Setup menu.

Wake-On-RTC	Enable Wake-On-RTC.
RTC Alarm	Use TAB to switch between hours, minutes and seconds. Use digits and BKSP to change field.
Video Frame Buffer Size	Select the size of the video frame buffer.
IGD MSI	IGD Message Signalled Interrupts control.
Route USB P2 to USB Client	USB port 2 can be routed to Client or host controller.

3.14 Advanced CPU Setup Menu

The Advanced CPU Setup menu shows status information of the CPU.

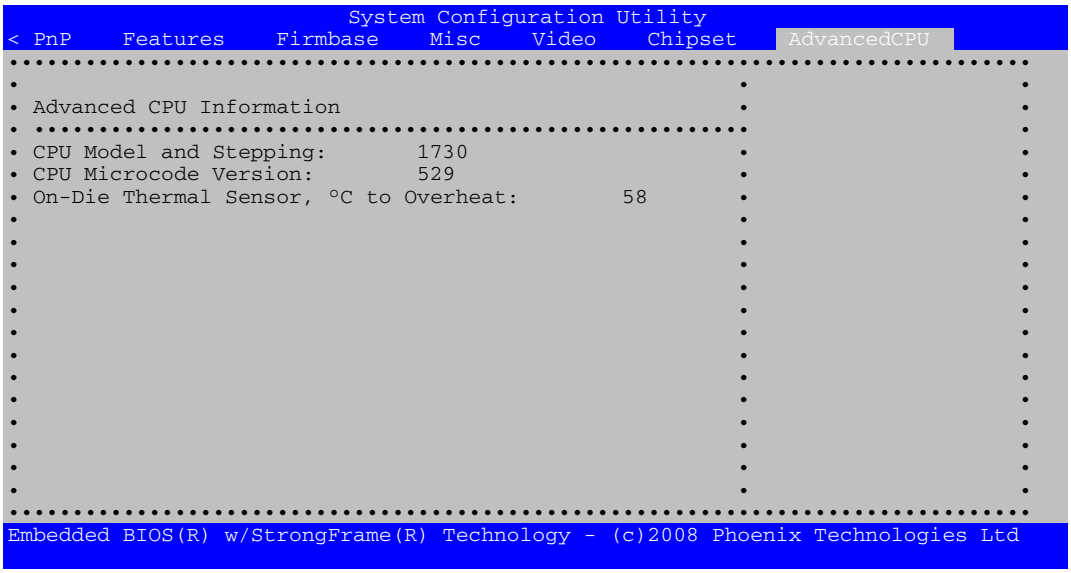

Figure 16: Advanced CPU Setup

The following table presents the items in the Advanced CPU Setup menu.

CPU Model and Stepping	Shows the CPU model and stepping.
CPU Microcode Version	Shows the version of the installed microcode update.
On-Die Thermal Sensor, °C to Overheat	Show the temperature reserve to overhead.

4 REFLASH

There are two BIOS flash ROMs, which can be used alternately, selectable per jumper (see the Hardware section of this manual). The one flash is at the CPU module and the other at the baseboard.

- We take a look on the "*BIOS SEL*." Jumper on the base board. Open this jumper for booting using BIOS on the CPU module or close it for booting using BIOS on the baseboard.

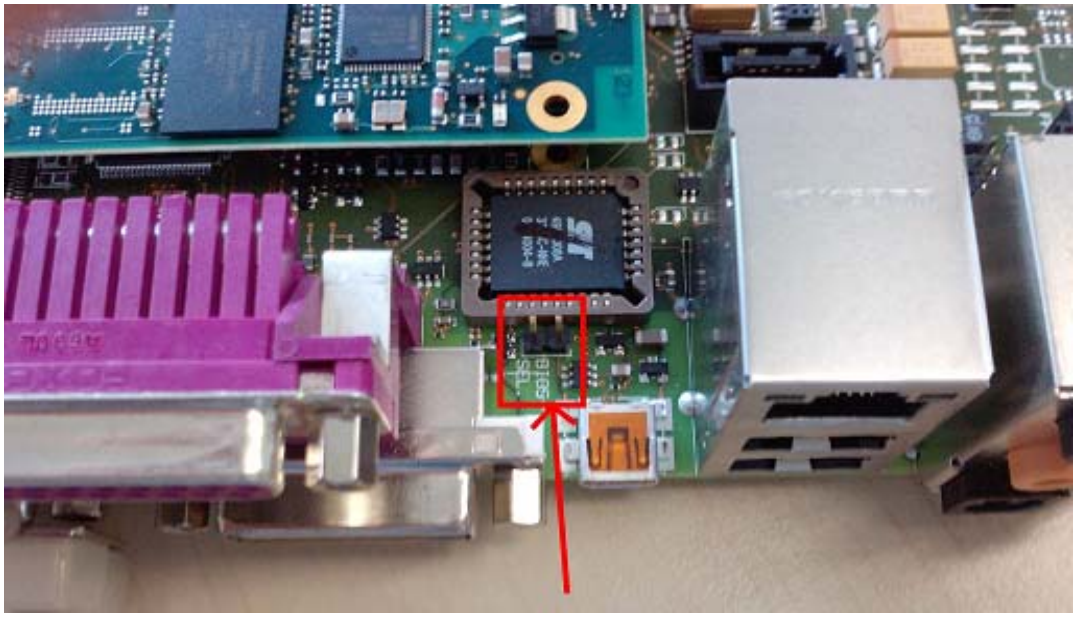

Figure 17: BIOS SEL Jumper location

For updating the BIOS first of all creation of an USB-stick with FreeDOS is needed. Therefor you need to download the following files:

Please visit http://www.freedos.org and download *fdbasews.iso*.
Please visit http://advancemame.sourceforge.net/boot-download.html and download tool **makebootfat** for windows.
Please visit http://syslinux.zytor.com and download *syslinux-3.84.zip*.

Open archive *fdbasews.iso* and extract *kernels.zip* out of subdirectory *freedos\packages\src_base*.

Now you should create a working directory on your system in order to assemble all necessary files. These are:

- *makebootfat.exe* out of the **makebootfat** package.

- mbr.bin out of subdirectory *mbr* from the **syslinux** package.
  (This is because *mbrfat.bin* within the **makebootfat** package sometimes doesn't work.)

- *fat12.bin*, *fat16.bin* and *fat32lba.bin* out of subdirectory *source\ukernel\boot* from archive *kernels.zip*.
  Rename these three files from *.bin* to *.bss*.

- Create a subdirectory *root*.

- Within this subdirectory put *command.com* and *kernel.sys* out of subdirectory freedos\setup\odin from archive fdbasews.iso.

After that, format the USB-stick from the windows command line with "format e: /fs:fat32" (assuming *E:* as the corresponding drive letter). In case of trouble later on, you can try "format e: /fs:fat" instead. Some BIOSs don't support booting from FAT32.

Remove all other USB drives (hard drives as well as flash drives) from the computer. Otherwise FreeDOS might be installed to one of them unintentionally and your data is lost!

Enter your working directory and install bootable FreeDOS to your USB-stick with command
makebootfat -o usb -E 255 -b fat32lba.bss -m mbr.bin root

4.1 Updating the BIOS Flash ROM on the CPU Modul

Please visit ftp://ftp.phytec.de/pub/Products/phyCORE-Z500PT/Tools, download *spiprog.exe* as well as *CRO301a.bin* and copy them additionally to the files on your FreeDOS bootable USB-stick.

Savely remove the USB-stick from your computer and plug it into the board.

Connect the board with the power supply and boot it.

Start the *SPIprog* tool by typing:
spiprog cro301a.bin

Remove the USB-stick, open the BIOS SEL jumper if not already open and restart your system.
You will see the new BIOS starting.

4.2 Updating the BIOS Flash ROM on the Baseboard

Please visit ftp://ftp.phytec.de/pub/Products/phyCORE-Z500PT/Tools, download *reflash.exe* as well as *CRO301a.bin* and copy them additionally to the files on your FreeDOS bootable USB-stick.

Savely remove the USB-stick from your computer and plug it into the board. Connect the board with the power supply and boot it.

Reflash is a simple DOS-based utility that loads a valid Embedded BIOS image, and uses the media driver from the BIOS within that image to reflash the BIOS. It may be used to update any area of Flash supported by the BIOS build However, it was specifically designed to reflash the BIOS itself. There are two modes for this utility that determine what media layer (Flash update code) is used to write to Flash parts. In local mode, the utility finds the INT 15h entry point in a new BIOS image specified in the script. That entry point is then used to access the media layer within the new BIOS image, so that there are no dependencies on the current system BIOS. This means that as long as a General Software BIOS is loaded as the Flash driver, it does not matter what BIOS is on the target system or if that BIOS

has a working media layer as long as that BIOS does not actively protect the Flash part from being updated. If local mode is being used and there are problems updating the Flash, the error must be in the system BIOS image being used as a Flash driver. Always check the configuration of the media layer and the board module code to support write enable in the new BIOS image before altering this utility.

When using manufacturing mode, the BIOS on the far side of the link must be Embedded BIOS and must already be in manufacturing mode. Note that at this time, only parallel and serial modes are supported. If other link types are added to manufacturing mode, this code may need to be updated to support those modes.

Failures in remote mode are typically related to the link or to the BIOS running on the target system. If either of these is compromised, the BIOS cannot be updated remotely. Note that running the Firmbase console (or any other software) on the same COM port as a manufacturing mode link can cause problems with remote update as the two will interfere with each other. Note that failures in both modes can be caused by inappropriate update scripts. For example, if the wrong media region is specified for erase or write cycles, the utility will fail to work because the media region in the system BIOS will be unable to compensate for the incorrect address. Also note that some media drivers will fail if the wrong block size is specified for erase instructions.

4.2.1 Command Line Options

Reflash works normally without needing any command line parameters; simply type REFLASH at the DOS prompt, and it will automatically determine how to update the BIOS from a file located in the current working directory. However, command line parameters may be supplied to override the default (automatic) operation. The command-line syntax for Reflash is formally the following:

```
REFLASH [/REDIR=COMx]
[/CLE[ARSCREEN]]
[/Q[UIET]]
[/CMDFILE=filename]
[/REB[OOT]]
[/PAR[ALLEL]]
[/SER[IAL]]
[/PORT=COMx]
[/BAUD=rate]
[/D sym=value]
[/?]
```

/REDIR=COMx	Specifies the COM port that will be used for console redirection.
/CLE /CLEARSCREEN	Specifes that Reflash will clear the screen prior to displaying its banner.
/Q /QUIET	Specifies that Reflash will clear not display progress reports, such as thermometer-style screen updates, during operation.
/CMDFILE=filename	Specifies the name of a file containing the script to read in order to perform the reflashing operation. By default, this filename is REFLASH.CMD.
/REB /REBOOT	Specifies that REFLASH will attempt to reboot the system after it has completed its work.
/PAR /PARALLEL	Specifies that Reflash will use the parallel port (ECP/EPP) link to access manufacturing mode protocol with the target.

/SER /SERIAL	Specifies that REFLASH will use the serial port (RS-232) link to access manufacturing mode protocol with the target.
/PORT=COMx	Specifies the serial communications port to be used when /SERIAL is specified. 'x' can take on the value 1, 2, 3, or 4. By default, COM1 is assumed.
/BAUD=rate	Specifies the serial communications baud rate to be used when using Reflash as a client on a HOST machine (not the target itself), to connect to the target using Manufacturing Mode. 'x' can take on the values AUTO, 9600, 19K, 28K, 38K, 56K, or 115K. By default, AUTO is assumed, meaning REFLASH will automatically determine the baud rate established by the target.
/D sym=value	Specifies that a symbol will be defined, just as it is in the environment, so that the script may inspect its value and make decisions based on it.
/?	Requests REFLASH to display its help screen.

4.2.2 Running Reflash on the System

Close the BIOS SEL jumper if not already closed .Start the *Reflash* tool by typing:
reflash

Remove the USB-stick and restart your system.
You will see the new BIOS starting.

Note: If you modify your FreeDOS in a case, in which it uses an extended memory manager (i.e., EMM386.EXE), this will mask the true physical address space of the system and defeat REFLASH.EXE's ability to switch to protected mode and access the Flash directly. So just don't do that!

If you're a BIOS developer, and not an end user, you may wish to copy the build log file containing MD5 hashes and descriptions of all

valid binaries built for the system onto the disk, allowing Reflash to verify that these are correct before proceeding.
Next, boot the system with the DOS-bootable disk, and wait until the DOS prompt ("A:>" or "C:>" appears.)

Finally, type REFLASH at the command prompt, optionally specifying the name of the new BIOS image as a parameter. Figure 18 shows a typical screen shot that shows Reflash in action, updating the ROM and displaying the status of the update with a progress thermometer on the display.

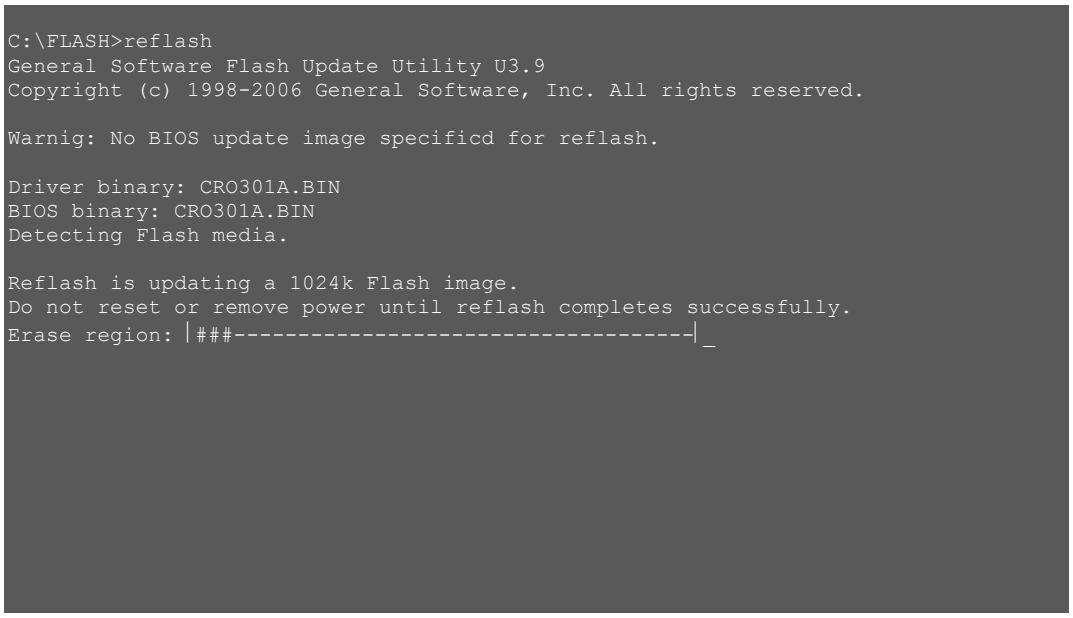

Figure 18: Running the Reflash utility from DOS

Finally, if Reflash indicates that the operation was successful, then reboot the system with a COLD BOOT before performing other activities.

WARNING: If the operation did not complete successfully, DO NOT reboot your system, because the BIOS image is not correctly installed; instead, the Flash may be partially programmed, or partially erased, etc.

Figure 19 shows a typical case where Reflash detects a problem and warns the user to not reboot, but instead to try alternatives, to avoid having to remove the Flash ROM from the board.

```
C:\FLASH>reflash
General Software Flash Update Utility U3.9
Copyright (c) 1998-2006 General Software, Inc. All rights reserved.

Warnig: No BIOS update image specificd for reflash.

Driver binary: CRO301A.BIN
BIOS binary: CRO301A.BIN
Detecting Flash media.

Reflash is updating a 1024k Flash image.
Do not reset or remove power until reflash completes successfully.
Erase set zero: |----------------------------------------|
Unable to write 0 to FFF00000.

Reflash has failed to update the image but the contents of the Flash part
Remains unchanged.  The Flash contents should be intact, so it may be safe
To reset the machine.

However, you should still make a backup copy of the original flash contents
To restore the original image from.

You can store the original Flash contents to CRO301A.BIN.
Do you wish to store the original Flash contents to CRO301A.BAK (Y/N)?
```

Figure 19: Reflash can warn the user when the Flash isn't reprogramming correctly

Please read this final note about reflashing; it can't be interrupted because the entire ROM must be programmed as a whole.

WARNING: The Reflash program MUST NOT BE INTERRUPTED! A power outage may be fatal to any unit with a soldered-on Flash part. No recovery method is provided, since this version of Embedded BIOS does not support an intrinsic boot block recovery structure.

4.2.3 Notes and Cautions

Make certain that the BIOS and any part of the Flash that is being overwritten is shadowed in RAM. This is very important. If the current BIOS is running directly from ROM, and you attempt to re-write a video BIOS that is running directly from ROM or you try to overwrite any portion of the Flash that is currently being used by the operating system, the system may hang in the middle of a reflash and be left in an unbootable state. Reflash automatically checks to make sure memory managers such as EMM386.EXE or Windows are not running. Any paging mechanism would cause Reflash to not correctly access the actual physical media. Reflash can and does take advantage of HIMEM.SYS to buffer its data if it is available.

If you are using a silicon vendor reference design from a company such as AMD, Intel, VIA, or others, you may need to change a jumper or two on the board to cause it to supply the correct programming voltage for Reflash to work. Return the jumper to the original position after you have successfully completed reflashing. If your system has jumpers that need to be set for this purpose, it will be explained in your system's hardware operations manual; otherwise, assume there is no jumper needed, and give it a try. If Reflash runs and a jumper is needed for programming, Reflash won't be able to modify the Flash at all, so the part will be safe.

If you are having problems with getting Reflash to run, there may be a paging conflict with your current BIOS. If this is the case, then you may have to recompile Embedded BIOS so that it uses a different memory page for writing to the Flash.

Finally, be sure to re-boot the machine when you are done running Reflash. Although the target seems to still be functioning, it is doing so by running the BIOS from shadow RAM, and is not running the new BIOS.

5 CONSOLE REDIRECTION

5.1 Using Console Redirection

Systems supporting the Console Redirection feature can operate either with a PS/2 or USB keyboard and video display, or with a special emulation of a console over an RS232 cable connected to a host computer running a terminal emulation program.

There are actually several different ways the serial port can be used, because there are several different components of the BIOS that use the console. The most basic use of the serial port is to redirect what the BIOS normally displays on the screen during POST, together with the output that a simple OS like MS-DOS displays. In this role, Console Redirection simply emulates the keyboard and video display with a strictly character-based monochrome approach.

Of course, the Preboot environment, including the preboot menu and its applications, is another user of the keyboard and video display, and can be redirected just like POST can be. The preboot environment uses color and a special screen painting algorithm that makes it seem more drawing-oriented rather than scrolling text.

A third component of the BIOS is the Integrated Debugger, which normally uses the keyboard and video display to interact with the user in a scrolling text manner, just like POST (actually, rather like the DEBUG program in MS-DOS.) This component can be redirected to the serial port as well.

These three components (POST/DOS, Preboot, and Debugger) are all configurable “channels” for the BIOS Console Redirection feature. If the redirection feature is on, then the user can select which device (standard keyboard and video display, COM1, COM2, COM3, COM4) will be used as the console for each of these three major “channels”.

In addition to the standard BIOS components that can use serial ports for consoles, Firmbase can also use a serial port for several purposes; namely, the System Console, the Debug Console, and a User Shell. The use of the serial ports for these purposes is separate from the Console Redirection feature, although it is a reuse of the same idea. To learn more about how Firmbase can be configured to use the serial ports, see Chapter 3. The remainder of this chapter assumes you're going to redirect POST/DOS, the Preboot environment, or the Integrated Debugger features.

5.1.1 Setting up the Host Computer

To use the Console Redirection feature, you'll need a host computer running a program that emulates an ASCII terminal. Both Windows and Linux have such programs; consider using TeraTerm, PROCOMM, DOS Navigator, or Hyperterminal (although Hyperterminal sometimes crashes or causes the host system to become compute-bound without warning.)

Connect the system running Embedded BIOS to your host computer with a null modem RS232 cable. A so-called straight-through cable will not work.

Next, set the communications parameters of the host's terminal program to 115Kbaud, unless otherwise suggested by the OEM supplier of your system; other baud rates are possible but normally this feature uses the fastest standard baud rate. Other parameters are 8-bit, no parity, and one stop bit. Do not enable XON/XOFF or hardware flow control.

5.1.2 Configuring Console Redirection on the Target

With this link set up, power-on the system and from the host's terminal program, press ^C a few times as the system boots. POST will redirect to the serial console, and after it has completed its early stages, it will start the preboot menu, a full-screen color application from which all the preboot services may be accessed.

If no output appears in the terminal emulation program, use the standard console to enter the System Configuration Utility from the

preboot menu, and configure Console Redirection in the Features menu; Console Redirection may be disabled by default. Settings include “Never”, which means Console Redirection is completely disabled; “Always”, which means Console Redirection always works and the main keyboard and screen are never used; and “Auto”, which means Console Redirection defaults to disabled and then looks for reasons why it should enable itself.

Another Setup menu to consider is SIO, which is where Super I/O components are typically configured to enable the serial ports on a system. Console Redirection normally uses the serial port at 3f8h (and does not use interrupts.) This I/O address can be changed by the OEM but is normally not changed.

5.1.3 Conditional Access Via Console Redirection

If the user presses one of the POST control keys, such as ESC, ^B, ^C, ^D, ^T, or ENTER, then POST begins console redirection, and uses the character as it would on the main console.

CAUTION: HyperTerminal’s default setting is to use flow control, which will render the console inoperative. To change this, create a new session, change the flow control setting to [none], save the session, and exit HyperTerminal. Then reinvoke HyperTerminal with the session, and it will operate with the new flow control setting.

5.2 POST/DOS Channel

The most common use of the Console Redirection feature is to redirect the keyboard and video display for POST and DOS to a serial port. This is common on headless devices that would not otherwise be thought of as containing a PC-style computer that “boots”.

Examples might include settop boxes or network appliances like routers or gateways. Often these devices have an RJ11 connector on the back that allows a user to configure the device with a terminal program running on a laptop, using a special cable with an RJ11 plug on one end and a DB9 connector on the other.

Operating systems other than DOS are supported; however, they must use BIOS calls (INT 10h and INT 16h for video and keyboard, respectively) in order to access the console. Any attempts by programs to “paint” the video controller’s display memory, or interact directly with the keyboard controller (8042) itself will bypass the Console Redirection feature, and will not allow the user to reap the benefits thereof.

An example of such an OS is Linux, which requires kernel configuration in order to redirect its console to a serial port; this is not handled by the BIOS itself, but actually by Linux directly; therefore, for seamless operation, it is necessary for the system developer to ensure that the Linux OS’ UART operating parameters (I/O address, and communications parameters) match those associated with the BIOS. Users who attempt to load a Linux operating system of their own may need to configure it in order to match the settings made by the BIOS to ensure seamless operation.

6 UNIVERSAL SERIAL BUS

6.1 USB Features

The system BIOS incorporates the industry-standard 32-bit Firmbase Technology V2, which provides comprehensive support for USB mass storage and human interface devices, including the ability to boot operating systems from USB devices, and emulate a PS/2 keyboard and mouse with their USB equivalents.

USB is actually a sophisticated hierarchical bus that supports multiple keyboards, mice, floppy disks, hard disks, CDROM/DVD/CDR and other devices, such as printers, scanners, cameras, and other equipment. Of these, only bootable mass storage devices are supported by the USB boot feature, and only those USB devices that support the industry standard USB Boot HID protocol are supported (virtually all keyboards and mice, but not strictly all.) In general, standard USB keyboards and mice purchased at a computer store will work with the BIOS USB feature.

Not all BIOSes have the depth of USB support that Firmbase V2 provides. Your OEM may configure the stack to support USB 1.0, 1.1, and/or 2.0 devices, and through any subset of the USB host controller's root hub ports, optionally leaving some ports unsupported by firmware so as to not interfere with specialized applications.

As well, the Firmbase V2 USB stack may be configured by the OEM to support plug-in USB host controller cards natively, eliminating the need for third party drivers to configure them. The USB stack features the industry-standard handoff, as defined by Microsoft documentation, allowing the preboot environment to configure and use the USB devices up to exactly the point at which the OS wishes to take over the devices' operation. When the OS wishes to hand-back the devices, the USB stack follows the industry-standard handback protocol to resume control over the devices.

Operating systems not following the industry-standard protocols for handoff and handback may have limitations when used with this full-featured USB stack. Some operating systems, like DOS and Windows NT, do not have built-in USB device support, and so never execute a handoff or handback protocol; instead, these operating systems and their application programs believe they are communicating with real PS/2 devices and real “INT 13h” disk drives in the system.

The USB services support arbitrarily complex bus topologies incorporating USB hubs, up to eight USB keyboards and mice simultaneously, and a large variety of USB mass storage devices including floppy disk drives, hard disk drives, CD/DVD ROM drives, CompactFlash readers, Zip disk drives, DiskOnKey devices and much more. Firmbase runs in System Management Mode (SMM) and must be enabled within the System Setup Utility’s Firmbase menu, as described in Chapter 3. On some systems, SMM is required for normal system operation, and is always enabled, even without an option in the System Setup Utility.

WARNING: USB hubs and other devices may be powered or unpowered. It is up to the user to connect devices in such a configuration that each hub and device has sufficient power to operate properly. When this configuration is incorrect, devices may not have enough power to function, and as a result may operate erratically or not at all.
Two system setup utility menus are relevant to the operation of USB; the first is the Boot menu, which deals with bootable actions, and if configured by the OEM, may include bootable USB devices in the boot action list. This menu is shown in Figure 20.

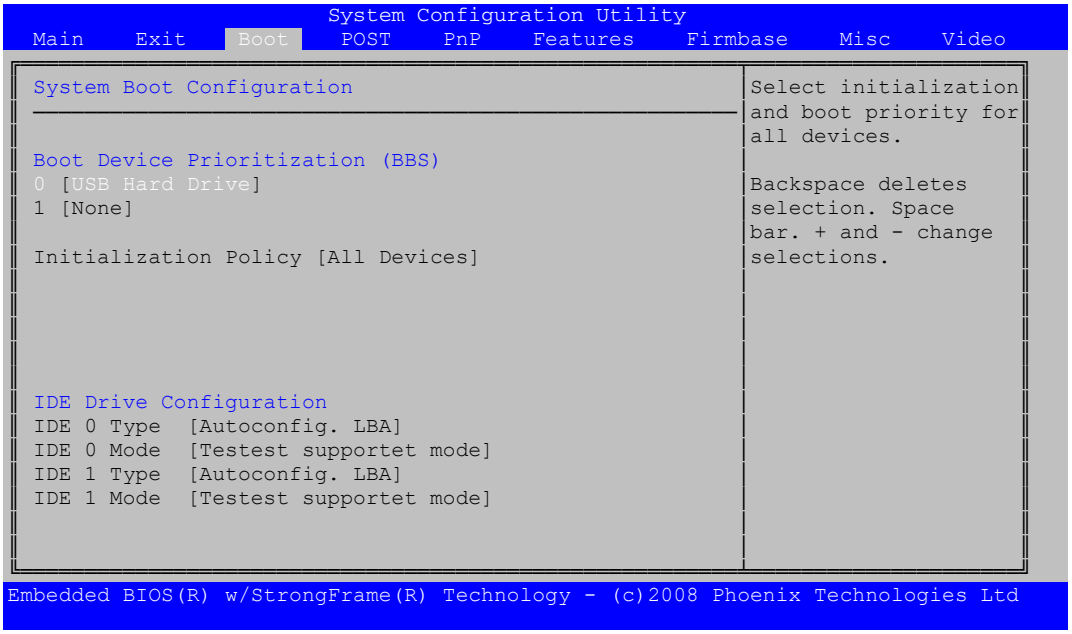

Figure 20: System Configuration Utility's Boot Menu showing enabled USB drive

The second system setup utility menu relating to USB is the Firmbase menu; it enables the technology components used by the USB stack needed to support legacy USB and boot from USB. This menu is shown in Figure 21 below.

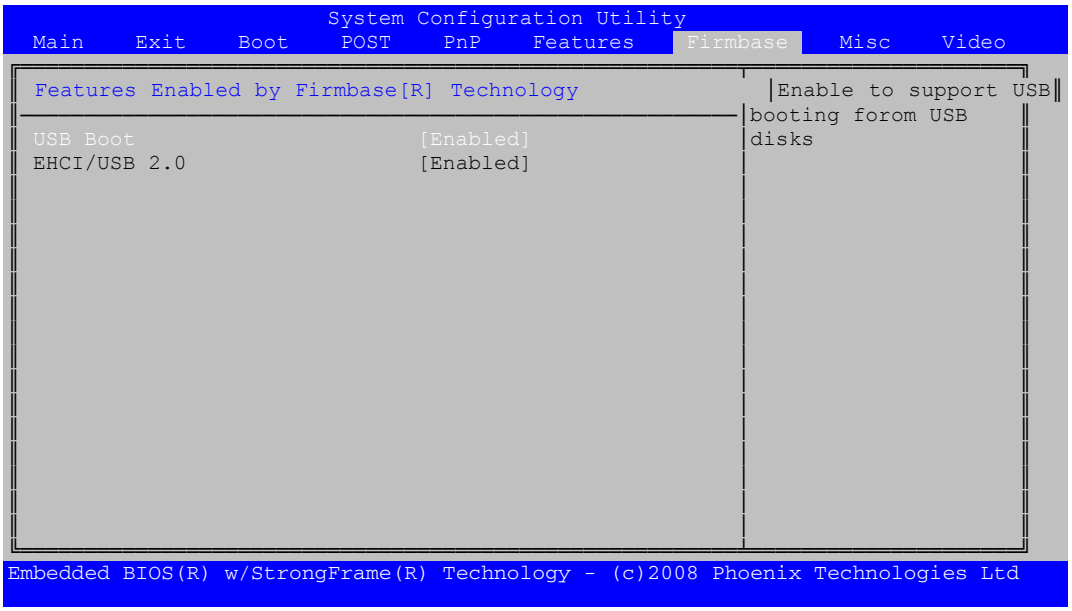

Figure 21: System Configuration Utility's Firmbase Menu showing USB Boot and EHCI/USB 2.0 features enabled.

6.2 Legacy USB Keyboard and Mouse

The system supports USB keyboards with or without integrated hubs. The devices must be connected either directly to the platform's USB ports or through a USB hub prior to power-up in order to make sure they are detected and configured properly. The USB keyboard is also available during POST like a traditional PS/2 device so it can be utilized for BIOS configuration and with the integrated debugger.

USB mice must also be connected to the TARGET prior to power-up so they too can be properly detected and configured during POST. The operating system level driver, for example MOUSE.COM in MS-DOS, will need to be installed to enable complete functionality.

WARNING: Both USB and PS/2 devices may be connected to the system simultaneously; however, due to hardware limitations inherent in the PC keyboard controller architecture, it is possible that when using the USB devices at the same time as the PS/2 devices, the keyboard controller may confuse the data streams and intermix them inappropriately, causing the mouse buttons and mouse movements to become erratic. It is advisable to not actively use PS/2 and USB devices at the very same instant to avoid this limitation. To use legacy USB devices, make sure it is configured in the System Setup Utility's Firmbase menu. It is not necessary to enable USB 2.0 as well unless USB 2.0 devices will also be supported (see USB Mass Storage below), as USB HID devices do not use USB 2.0 protocols.

6.3 USB Mass Storage

If configured by the OEM, the system's USB boot feature enables the user to boot the system from a variety of USB storage devices that use no emulation, floppy disk emulation, and hard disk emulation. Some of the USB devices supported include:

- Hard disk drives
- Floppy disk drives
- CD/DVD ROM drives
- CompactFlash readers
- DiskOnKey or memory stick devices
- Zip disk drives

Many of these devices are not configured from the factory to be used as boot devices, therefore it may be necessary to run FDISK /MBR on the device to rewrite the MBR and make it bootable. Any device with a capacity greater then 1.44MB, other than CD/DVD ROM drives, will be automatically treated as a hard disk drive.

6.3.1 Adding USB Drives to the BBS Boot List

The Boot menu enables the user to select which drives will be supported by the system, and also what order they will be scanned to find bootable operating systems. To boot from a USB device, place it early in the system's boot device prioritization list. If a bootable device precedes the USB device, it will still be accessible by the OS booted from the drive that appears earlier on in the list; allowing the USB device to be a data drive or a backup OS boot drive.

When a USB drive is first configured in the BBS list, it will appear with a generic name such as "USB Hard Drive" or "USB CDROM Drive", etc. If, after saving and rebooting the target, the user enters setup again, the system will have automatically scanned the specified generic drive type and obtained the actual name of the drive, such as "Corsair 1GB USB stick SerNo D2961F42R-X", etc.

6.3.2 Enabling USB in the Configuration Utility's Firmbase Menu

In addition to adding a drive to the Boot menu, the user must enable the USB Boot option from the Firmbase menu. The "USB Disk I/O" is a different feature and has nothing to do with USB Boot; do not enable it to support USB Boot.

However, it is desirable whenever possible to enable USB 2.0, as it allows a target's EHCI (USB 2.0) controller to communicate with your USB peripherals at USB High Speed (480Mb/sec), a substantial improvement over USB 1.x speeds.

Appendix A List of POST codes

Symbol definitions associated with POST progress reporting through I/O port 80h.
A few codes are overloaded, and they appear here in chronological, not numerical, order.

POST_STATUS_START	00h	;POST beginning.
POST_STATUS_CPUTEST	01h	;CPU register test about to start.
POST_STATUS_DELAY	02h	;NMIs are disabled; delay starts.
POST_STATUS_DELAYDONE	03h	;power-on delay finished.
POST_STATUS_KBDBATRDY	04h	;kbd BAT done; reading kbd SYS bit.
POST_STATUS_DISABSHADOW	05h	;disabling shadowing & cache.
POST_STATUS_CALCCKSUM	06h	;calcing ROM cksum, wait kbd ctrlr.
POST_STATUS_CKSUMGOOD	07h	;cksum okay, kbd ctrllr free.
POST_STATUS_BATVRFY	08h	;verifying BAT cmd to kbd ctrllr.
POST_STATUS_KBDCMD	09h	;issuing kbd ctrllr cmd byte.
POST_STATUS_KBDDATA	0ah	;issuing kbd ctrllr data byte.
POST_STATUS_BLKUNBLK	0bh	;issuing pin 23,24 blocking & unblocking.
POST_STATUS_KBDNOP	0ch	;issuing kbd ctrllr NOP cmd next.
POST_STATUS_SHUTTEST	0dh	;testing CMOS RAM shutdown register.
POST_STATUS_CMOSDIAG	0eh	;checking CMOS cksum, updating DIAG byte.
POST_STATUS_CMOSINIT	0fh	;initializing CMOS (if req'd every boot).
POST_STATUS_CMOSSTATUS	10h	;init CMOS status reg for date/time.
POST_STATUS_DISABDMAINT	11h	;disabling DMA, interrupt ctrllrs.

POST_STATUS_DISABPORTB	12h	;disabling Port B, disabling video display.
POST_STATUS_BOARD	13h	;init board, start auto-mem detect.
POST_STATUS_TESTTIMER	14h	;starting timer tests.
POST_STATUS_TESTTIMER2	15h	;testing 8254 T2, for spkr, part B
POST_STATUS_TESTTIMER1	16h	;testing 8254 T1, for refresh.
POST_STATUS_TESTTIMER0	17h	;testing 8254 T0, for 18.2Hz.
POST_STATUS_MEMREFRESH	18h	;starting memory refresh.
POST_STATUS_TESTREFRESH	19h	;testing memory refresh.
POST_STATUS_TEST15US	1ah	;testing 15usec refresh ON/OFF time.
POST_STATUS_TEST64KB	1bh	;testing base 64KB memory.
POST_STATUS_TESTDATA	1ch	;testing data lines.
POST_STATUS_TESTADDR	20h	;testing address lines.
POST_STATUS_TESTPARITY	21h	;testing parity (toggling).
POST_STATUS_TESTMEMRDWR	22h	;base 64KB mem read/write test.

Now we have memory, so we can use a stack to use Pcall, not Rcall.

POST_STATUS_SYSINIT	23h	;system init before vector table init.
POST_STATUS_INITVECTORS	24h	;init vector table.
POST_STATUS_8042TURBO	25h	;reading 8042 for turbo switch setting.
POST_STATUS_POSTTURBO	26h	;initializing turbo data.
POST_STATUS_POSTVECTORS	27h	;any init after vector table init is next.
POST_STATUS_MONOMODE	28h	;setting monochrome mode.
POST_STATUS_COLORMODE	29h	;setting color mode.
POST_STATUS_TOGGLEPARITY	2ah	;toggle parity before optional video ROM test.
POST_STATUS_INITBEFOREVIDEO	2bh	;init before video ROM check.
POST_STATUS_VIDEOROM	2ch	;control passed to video ROM.
POST_STATUS_POSTVIDEO	2dh	;video ROM returned control.
POST_STATUS_CHECKEGAVGA	2eh	;checking for EGA/VGA adapter found.

POST_STATUS_ TESTVIDEOMEMORY	2fh	;no EGA/VGA found, r/w test of video memory.
POST_STATUS_RETRACE	30h	;looking for video retrace signal.
POST_STATUS_ALTDISPLAY	31	;retrace failed, checking alt. display.
POST_STATUS_ALTRETRACE	32h	;alt found, checking video retrace signal.
POST_STATUS_VRFYSWADAPTER	33h	;compare switches w/actual adapter type.
POST_STATUS_SETDISPMODE	34h	;setting display mode.

Now we have a display. All code that outputs codes at 35h and above can use INT 10h to display messages.

POST_STATUS_CHECKSEG40A	35h	;check ROM BIOS data area at seg 40h.
POST_STATUS_SETCURSOR	36h	;setting cursor for power-on msg.
POST_STATUS_PWRONDISPLAY	37h	;displaying power-on message.
POST_STATUS_SAVECURSOR	38h	;save cursor position.
POST_STATUS_BIOSIDENT	39h	;display BIOS ident. string.
POST_STATUS_HITDEL	3ah	;display "Hit <DEL> to ..." msg.
POST_STATUS_VIRTUAL	40h	;preparing vm test. vrfy from display memory.
POST_STATUS_DESCR	41h	;preparing descriptor tables.
POST_STATUS_ENTERVM	42h	;enter virtual mode for memory test.
POST_STATUS_ENABINT	43h	;enable ints for diagnostics mode.
POST_STATUS_CHECKWRAP1	44h	;init data for checking wraparound at 0:0.
POST_STATUS_CHECKWRAP2	45h	;checking for wrap, find total memory size.
POST_STATUS_HIGHPATTERNS	46h	;write extended memory test patterns.
POST_STATUS_LOWPATTERNS	47h	;write conventional memory test patterns.

POST_STATUS_FINDLOWMEM	48h	;finding low memory size from patterns.
POST_STATUS_FINDHIMEM	49h	;finding high memory size from patterns.
POST_STATUS_CHECKSEG40B	4ah	;check ROM BIOS data area again.
POST_STATUS_CHECKDEL	4bh	;check for <DEL>, clear low mem for soft reset.
POST_STATUS_CLREXTMEM	4ch	;clearing ext mem for soft reset.
POST_STATUS_SAVEMEMSIZE	4dh	;saving memory size.
POST_STATUS_COLD64TEST	4eh	;on cold boot, display 1st 64KB memtest.
POST_STATUS_COLDLOWTEST	4fh	;on cold boot, test all of low memory.
POST_STATUS_ADJUSTLOW	50h	;adjust memsize for 1K usage.
POST_STATUS_COLDHITEST	51h	;on cold boot, test high memory.
POST_STATUS_REALMODETEST	52h	;prepare for shutdown to real-mode.
POST_STATUS_ENTERREAL	53h	;saved regs & memsize, entering real-mode.
POST_STATUS_SHUTDOWN	54h	;shutdown successful, restoring codepath.
POST_STATUS_DISABA20	55h	;disabling A20 line.
POST_STATUS_CHECKSEG40C	56h	;checking ROM BIOS data area again.
POST_STATUS_CHECKSEG40D	57h	;checking ROM BIOS data area some more.
POST_STATUS_CLRHITDEL	58h	;clear the "Hit <DEL>" message.
POST_STATUS_TESTDMAPAGE	59h	;test DMA page register.
POST_STATUS_VRFYDISPMEM	60h	;verify from display memory (???).
POST_STATUS_TESTDMA0BASE	61h	;test DMA0 base register.
POST_STATUS_TESTDMA1BASE	62h	;test DMA1 base register.
POST_STATUS_CHECKSEG40E	63h	;checking ROM BIOS data area again.
POST_STATUS_CHECKSEG40F	64h	;checking ROM BIOS data area some more.

POST_STATUS_PROGDMA	65h	;programming DMA ctrllrs 0 & 1.
POST_STATUS_INITINTCTRL	66h	;initializing INT ctrllrs 0 & 1.
POST_STATUS_STARTKBDTEST	67h	;starting keyboard test.
POST_STATUS_KBDRESET	80h	;issuing reset cmd & clring output buffer.
POST_STATUS_CHECKSTUCKKEYS	81h	;check for stuck keys & issue test cmd.
POST_STATUS_INITCIRCBUFFER	82h	;initializing circular buffer.
POST_STATUS_CHECKLOCKEDKEYS	83h	;check for locked keys.
POST_STATUS_ CHECKMEMSIZEMISMATCH	84h	;check for memsize mismatch (CMOS/BIOSDATA).
POST_STATUS_PASSWORD	85h;	check for pswd or bypass setup.
POST_STATUS_BEFORESETUP	86h	;pswd checked. do pgming before setup.
POST_STATUS_CALLSETUP	87h	;call the setup module.
POST_STATUS_POSTSETUP	88h	;back from setup, clr screen.
POST_STATUS_DISPPWRON	89h	;display power-on screen message.
POST_STATUS_DISPWAIT	8ah	;display "Wait..." message.
POST_STATUS_ENABSHADOW	8bh	;do system & video BIOS shadowing.
POST_STATUS_STDCMOSSETUP	8ch	;load standard setup params into BIOSDATA.
POST_STATUS_MOUSE	8dh	;check and initialize mouse.
POST_STATUS_FLOPPY	8eh	;check floppy disks.
POST_STATUS_CONFIGFLOPPY	8fh	;configure floppy drives.
POST_STATUS_IDE	90h	;check hard disks.
POST_STATUS_CONFIGIDE	91h	;configure IDE drives.
POST_STATUS_CHECKSEG40G	92h	;checking ROM BIOS data area again.
POST_STATUS_CHECKSEG40H	93h	;checking ROM BIOS data area some more.
POST_STATUS_SETMEMSIZE	94h	;setting base & ext mem sizes.
POST_STATUS_SIZEADJUST	95h	;memsize adjusted for 1K, verifying disp mem.

Initialize any ROM BIOS extensions.

POST_STATUS_INITBEFOREC8000	96h	;initialization before calling C800h.
POST_STATUS_CALLC8000	97h	;call ROM BIOS extension at C800h.
POST_STATUS_POSTC8000	98h	;processing after extension returns.

Setup serial ports, parallel ports, NPX, keyboard, cache, wait states.

POST_STATUS_TIMERPRNBASE	99h	;configuring timer data area, printer base addr.
POST_STATUS_SERIALBASE	9ah	;configuring serial port base addrs.
POST_STATUS_INITBEFORENPX	9bh	;initialization before coprocessor test.
POST_STATUS_INITNPX	9ch	;initializing the coprocessor.
POST_STATUS_POSTNPX	9dh	;processing after coprocessor initialized.
POST_STATUS_CHECKLOCKS	9eh	;check ext kbd, kbdID, numlock settings.
POST_STATUS_ISSUEKBDID	9fh	;issue keyboard ID command next.
POST_STATUS_RESETID	a0h	;kbd ID flag reset.
POST_STATUS_TESTCACHE	a1h	;do cache memory test.
POST_STATUS_DISPSOFTERR	a2h	;display any soft errors.
POST_STATUS_TYPEMATIC	a3h	;set keyboard typematic rate.
POST_STATUS_MEMWAIT	a4h	;program memory wait states.
POST_STATUS_CLRSCR	a5h	;clear screen.
POST_STATUS_ENABPTYNMI	a6h	;enable parity and NMIs.

Initialize ROM BASIC if available.

POST_STATUS_INITBEFOREE000	a7h	;initialization before calling E000h.
POST_STATUS_CALLE000	a8h	;call ROM BIOS extension at E000h.
POST_STATUS_POSTE000	a9h	;processing after extension returns.
POST_STATUS_TOPROTFAIL	aah	;mode switch to protected mode failed.

POST_STATUS_TOREALFAIL	abh	;mode switch to real mode failed.

Boot operating system.

POST_STATUS_DISPCONFIG	b0h	;display system config box.
POST_STATUS_INT19BOOT	00h	;call INT 19h bootstrap loader.

Additional paths.

POST_STATUS_LOWMEMEXH	b1h	;test low memory exhaustively.
POST_STATUS_EXTMEMEXH	b2h	;test extended memory exhaustively.
POST_STATUS_PCIENUM	b3h	;enumerate PCI space.
POST_STATUS_ADMGRINIT	b4h	;initialize address manager.
POST_STATUS_ADMGRBOOT	b5h	;preboot address manager callout.
POST_STATUS_HUGEMEMEXH	b6h	;test huge memory exhaustively.
POST_STATUS_SMBIOSINIT	b7h	;initialize SMBIOS structure table.
POST_STATUS_FBSIGNAL	b8h	;about to signal Firmbase.
POST_STATUS_MEMMGRINIT	b9h	;about to initialize low small memory manager.
POST_STATUS_INITDRIVER	bah	;about to initialize driver manager.
POST_STATUS_MPINIT	bbh	;about to start multiprocessor init.
POST_STATUS_A20_ERROR	bch	;A20 enable/disable error, system halted.

POST progress codes for the DRIVER MANAGER.

POST_SYSID_NONSTATE	000h	;The BIOS is at level 0.
POST_PREID_BOARD_0	001h	;BoardInit0 has been called.
POST_PREID_BOARD_PCCI	002h	;BoardPostCodeComInit called.
POST_PREID_BOARD_1	003h	;BoardInit1 has been called.
POST_PREID_NMI	004h	;PostMaskNmi has been called.

POST_PREID_NPX	005h	;PostResetNpx has been called.
POST_PREID_DELAY	006h	;PostDelay has been called.
POST_PREID_DISSHAD	007h	;BoardDisableShadow has been called.

BIOS driver managers and high level message handlers.

POST_DRVID_MEMMGR1	10h	;MemMgr1, Small Low Memory Manager.
POST_DRVID_RES	11h	;Res, Resource Manager.
POST_DRVID_POST	12h	;Post, POST.
POST_DRVID_INT	13h	;Int, Interrupt Manager.
POST_DRVID_STREAM	14h	;Stream, Stream Manager.
POST_DRVID_CON	15h	;Con, Console Manager.
POST_DRVID_DISK	16h	;Disk, Disk Manager Driver.
POST_DRVID_DEBUGMGR	17h	;DebugMgr, Debug Manager.
POST_DRVID_CONFIG	18h	;Config, Configuration Manager.
POST_DRVID_SOUND	19h	;Sound, Sound Manager Driver.
POST_DRVID_POINT	1ah	;Point, Pointer Manager Driver.
POST_DRVID_API	1bh	;Api, BIOS API Manager.
POST_DRVID_PNP	1ch	;Pnp, Plug-n-Play Manager.
POST_DRVID_USB	1dh	;Usb, USB Manager.
POST_DRVID_PCI	1eh	;Pci, PCI Manager.
POST_DRVID_UIMGR	1fh	;UiMgr, User Interface manager.
POST_DRVID_SMP	20h	;Smp, Multiprocessor Driver.
POST_DRVID_SIO	21h	;SioMgr, Super I/O Manager.
POST_DRVID_TPMDBG	22h	;TpmDbg, TPM Debugger Commands.

BIOS device drivers and low level message handlers.

POST_DRVID_TIME	30h	;Time, System Time Driver.
POST_DRVID_UIPC	31h	;UiPc, PC/AT User Interface.
POST_DRVID_CMOS	32h	;Cmos, PC/AT CMOS Driver.
POST_DRVID_PCATRTC	33h	;PcatRtc, PC/AT RTC Driver.

POST_DRVID_BOOT	34h	;Boot, PC/AT Boot Sequence.
POST_DRVID_DEMO	35h	;Demo, Demo Timeout Driver.
POST_DRVID_8259	36h	;8259, 8259 PIC Driver.
POST_DRVID_8042	37h	;8042, 8042 Keyboard Driver.
POST_DRVID_PORTB	38h	;PortB, ISA Port B Driver.
POST_DRVID_PORT92	39h	;Port92, Port 92h Driver.
POST_DRVID_CADULMON	3ah	;CadulMon, CAD-UL Monitor.
POST_DRVID_DEBUG	3bh	;Debug, BIOS Debugger.
POST_DRVID_MCL	3ch	;Mcl, Media Control Layer.
POST_DRVID_USBKBD	3dh	;UsbKbd, USB Keyboard Driver.
POST_DRVID_USBUHCI	3eh;	;UsbUhci, USB UHCI Driver.
POST_DRVID_PS2POINT	3fh	;Ps2Point, PS/2 Pointer Driver.
POST_DRVID_PCATSND	40h	;PcatSnd, PC/AT Sound Driver.
POST_DRVID_PCATSER	41h	;PcatSer, PC/AT Serial Driver.
POST_DRVID_PCATPAR	42h	;PcatPar, PC/AT Parallel Driver.
POST_DRVID_ANSICON	43h	;AnsiCon, ANSI Console Driver.
POST_DRVID_PCATCON	44h	;PcatCon, PC/AT Console Driver.
POST_DRVID_PCATKBD	45h	;PcatKbd, PC/AT Keyboard Driver.
POST_DRVID_PCATVID	46h	;PcatVid, PC/AT Video Driver.
POST_DRVID_FLOPPY	47h	;Floppy, Floppy Disk Driver.
POST_DRVID_UIGUI	48h	;UiGui, GUI User Interface.

Board, Chipset, CPU and other custom message handlers.

POST_DRVID_BOARD	60h	;Board, Board Personality Module.
POST_DRVID_CS	64h	;Cs, Chipset Personality Module.
POST_DRVID_CPU	68h	;Cpu, CPU Personality Module.

POST_DRVID_SUPERIO	6ch	;SuperIo, SuperIO driver.
POST_DRVID_REG	70h	;Reg, Registration driver.

Dokument: **phyCORE-Z500PT**
Dokumentnummer: **L-749e_1**

How would you improve this manual?

Did you find any mistakes in this manual? page

Submitted by:
Customer number: _____

Name: _____

Company: _____

Address: _____

_____

Return to:

PHYTEC Technologie Holding AG
Postfach 100403
D-55135 Mainz, Germany
Fax : +49 (6131) 9221-26

Published by

PHYTEC

Ordering No. L-749e_1
Printed in Germany