IB892

Intel ® Atom

EG20T Chipset

3.5” Disk Size SBC

USER’S MANUAL

Version 1.0

Acknowledgments

AMI BIOS is a trademark of American Megatrends Inc.
PS/2 is a trademark of International Business Machines Corporation.
Intel are registered trademarks of Intel Corporation.
Microsoft Windows is a registered trademark of Microsoft Corporation.
Winbond is a registered trademark of Winbond Electronics Corporation.
All other product names or trademarks are properties of their respective owners.

Table of Contents

Introduction 1

Product Description 1
Checklist 2
IB892 Specifications 3
Board Dimensions 4

Installations 5

Setting the Jumpers 6
Connectors on IB892 10

BIOS SETUP 17

Drivers Installation 33

Appendix 41

A. I/O Port Address Map 41
B. Interrupt Request Lines (IRQ) 42
C. Watchdog Timer Configuration 43
D. Digital I/O Sample Code 48

This page is intentionally left blank.

Introduction

Product Description

The IB892-10T (Atom E640T, 1.0GHz) and IB892-13T (Atom E660T 1.6GHz) are 3.5-inch disk size SBCs that support wide operating temperature [- 40 ~ +85 degree C]. They are based on the Intel® EG20T I/O Hub and come on board with 1GB DDRII-800 addressable memory

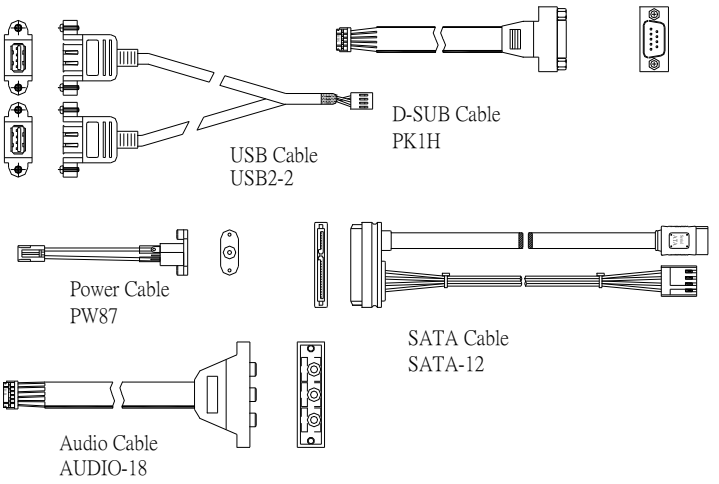
IB892-10T/13T SBC supports and integrated graphics controller with an LVDS interface (18/24-bit single channel). CRT output is available with the use of SDVO (Chrontel 7022A). The boards also have Gigabit LAN connectivity.

High speed communication and external connections are provided by 4 USB ports, high definition audio, Mini PCI-e, 4 COM ports - with COM1 supporting RS232/422/485 while others RS232, RJ45, CRT, Micro SD slot, CAN Bus, SATA and a CFast socket.

Checklist

Your IB892 package should include the items listed below.

- The IB892 3.5" disk-size SBC
- This User's Manual
- 1 CD containing chipset drivers and flash memory utility
- Options:
  - Cable kit (IB65: PW87, USB2-2, PK1H, SATA12, AUDIO-18)
  - Heatsink



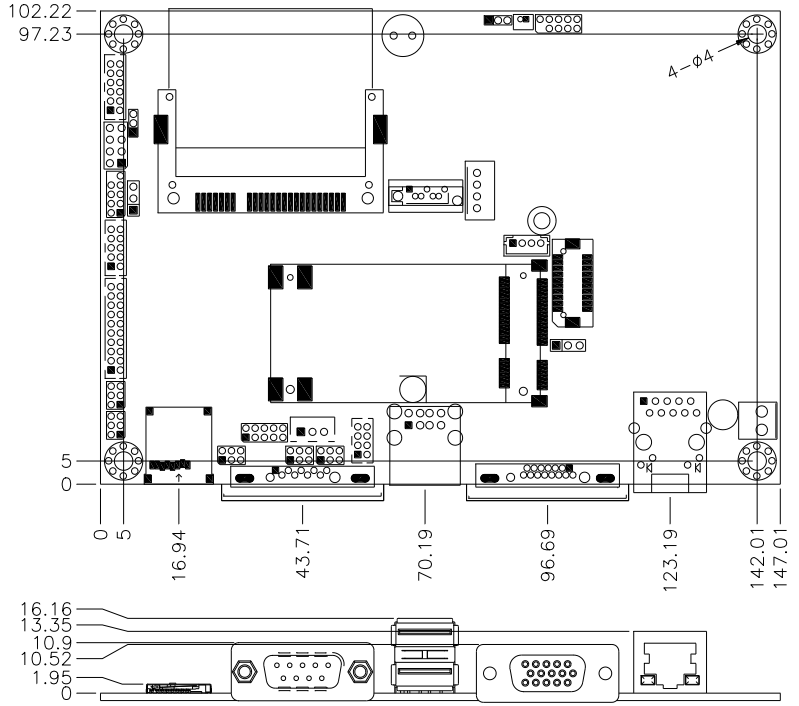
Optional Cables for IB892

IB892 Specifications

Product Name	IB892-10**T**(Atom™ E640T, 1.0GHz) IB892-13**T**(Atom™ E660T, 1.6GHz) ["T" means support Wide-Temp Operating- 40 ~ +85 degree C]
Form Factor	3.5" Disk Size
CPU Type	Intel® Atom™ E6xx series processor (45nm SC) FCBGA-676 balls (22 x 22 mm; TDP=2.7~3.3W)
CPU Speed	Intel® Atom™ E640**T** [1.0GHz(TDP=3.3W)] Intel® Atom™ E660 **T** [1.3GHz(TDP=3.6W)]
CPU FSB	800MHz
Cache	512KB
BIOS	AMI BIOS, supports ACPI Function
Chipset	Intel® **EG20T** I/O Hub (-40 to +85 degree C) PBGA-376 balls (23 x 23 mm; TDP = 2W)
Memory	DDRII-800 addressable memory **1GB** onboard (single channel) [128MBx8 SDRAM x **8** pcs]
Display	Integrated 2D/3D Graphics@320MHz (600MHz)@400MHz (1.3GHz) ● LVDS - 18/24-bit (Single channel) ● CRT - Thru SDVO by using Chrontel 7022A
LAN	Intel® PCI-e GbE LAN 82574IT x 1
USB	Intel® EG20T IOH built-in USB2.0 host controller with **4** ports
Audio	Intel® Atom™ E6xx series built-in HD Audio +Realtek ALC 662 5.1-Channel (Line-in, Line-out & MIC)
Expansion slot	Mini PCI-e x1 w/USB [Reserved one mounting hole only for half-sized]
SATA	Intel® EG20T IOH built-in SATAII x **1**
LPC I/O	Nuvoton NCT6627UD - COM1 (RS232/422/485), COM2(RS232), - COM3(RS232) & COM4(RS232) w/ pin-9 with power for 2 ports (500 mA for each port) - Hardware monitor (3 thermal inputs, 6 voltage monitor inputs)
RTC/CMOS	Intel® EG20T built-in with onboard Lithium Battery
Edge Connector	RJ45 x 1 for GbE DB15 connector x 1 for CRT Dual USB stack connector x1 for USB1,2 DB9 x 1 for COM 1
On Board Header / Connectors	2x4 pin header x **1** for 2* USB ports LVDS (DF13 x 1), 24-bit single channel Mini PCI-e connector x 1 (Reserved screw holes for Half-Mini type also) Micro SD slot x 1 2 x 6 pin box header x1 for Audio 2 x 4 pin DF11 x1 for KB/MS DF11-10 pin box header x 1 for COM2 DF11-20 pin box header x 1 for COM3, COM4 2x5 pin headers x 1 for LPC (80 port card debugging purpose) 3 pin headers x 1 for CAN Bus 2 pin headers x 1 for DC power-in 4-pin power connector x 1 for SATAII HDD CFast Socket x 1 (aligned with PCB edge) 4 pin box header x 1 for backlight/brightness control
Digital I/O	4 in/4 out
Watchdog Timer	Yes (256 segments, 0, 1, 2…255. sec/min)
Operating Temp.	-40 degree C to + 75 degree C for **Wide-Temp.**
Power Connector	+12V DC-in
Board Size	102x147mm (4"x5.8")

Others	1. Drivers for IB892 series will be special one, departed from current iBASE driver DVD 2. Due to the limitation from Intel driver, there will be below driver selections in drivers that iBASE can offering **[Under Windows XP]** 1. VGA+LVDS@ 640 x 480 resolution 2. VGA+LVDS@ 800 x 600 resolution 3. VGA+LVDS@ 1024 x 768 resolution 4. VGA+LVDS@ 1366 x 768 resolution **Linux driver will be available later** 3. Heatsink for IB892 will be built-in on the BOM, the height is 6mm

Board Dimensions



Installations

This section provides information on how to use the jumpers and connectors on the IB892 in order to set up a workable system. The topics covered are:

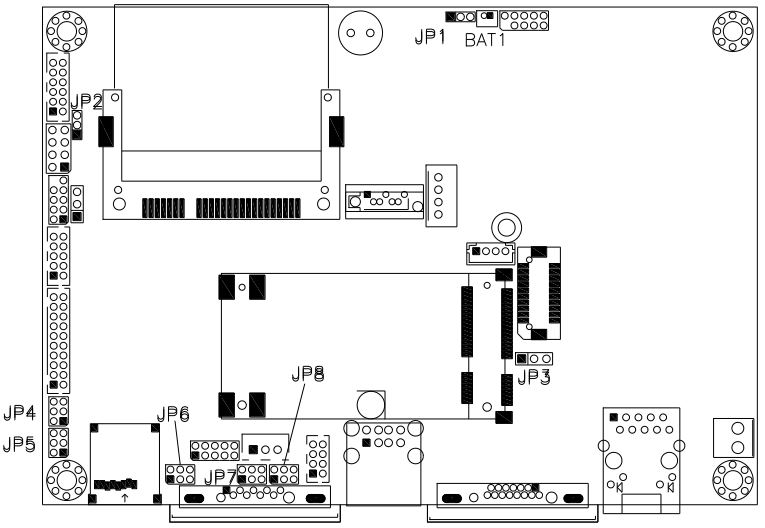
Setting the Jumpers 6
Connectors on IB892 10

Setting the Jumpers

Jumpers are used on IB892 to select various settings and features according to your needs and applications. Contact your supplier if you have doubts about the best configuration for your needs. The following lists the connectors on IB892 and their respective functions.

Jumper Locations on IB892 7
JP2: ATX or AT Power Selection 8
JP3: LCD Panel Power Selection 8
JP4, JP5: COM3,4 RS232 +5V/+12V Power Setting 8
JP6, JP7, JP8: RS232/422/485 (COM1) Selection 9

Jumper Locations on IB892



Jumpers on IB892.....Page
JP2: ATX or AT Power Selection..... 8
JP3: LCD Panel Power Selection..... 8
JP4, JP5: COM3.4 RS232 +5V/+12V Power Setting..... 8
JP6, JP7, JP8: RS232/422/485 (COM1) Selection..... 9

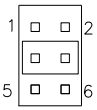
JP2: ATX or AT Power Selection

JP2	ATX Power
	ATX
	AT

JP3: LCD Panel Power Selection

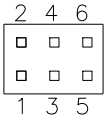
JP3	LCD Panel Power
	3.3V
	5V

JP4, JP5: COM3,4 RS232 +5V/+12V Power Setting

JP4, JP5	Setting	Function
	Pin 1-2 Short/Closed	+12V
	Pin 3-4 Short/Closed	RI
	Pin 5-6 Short/Closed	+5V

JP6, JP7, JP8: RS232/422/485 (COM1) Selection

COM2, COM3, COM4 are fixed for RS-232 use only.
COM1 is selectable for RS232, RS-422 and RS-485.
The following table describes the jumper settings for COM1 selection.

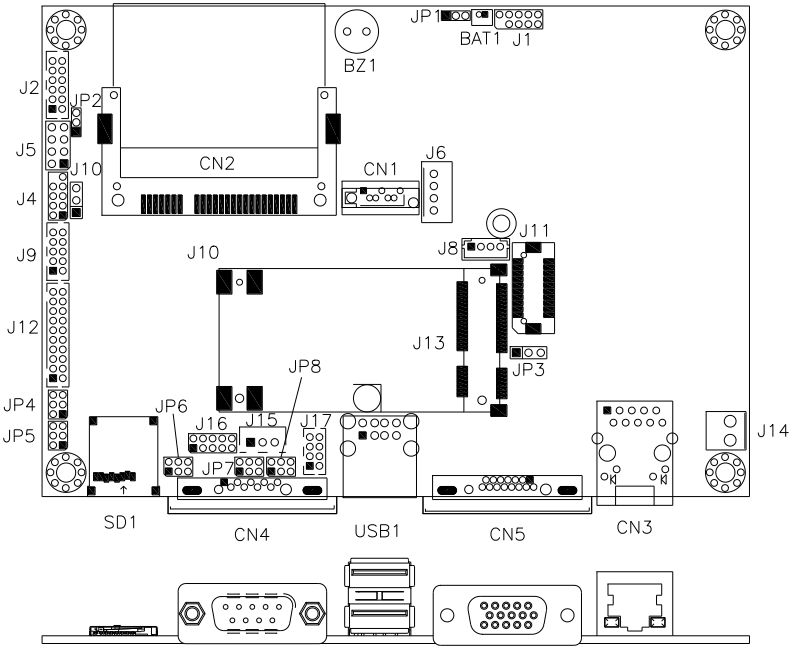


COM1 Function	RS-232	RS-422	RS-485
Jumper Setting (pin closed)	JP7: 1-2 JP6: 3-5 & 4-6 JP8: 3-5 & 4-6	JP7: 3-4 JP6: 1-3 & 2-4 JP8: 1-3 & 2-4	JP7: 5-6 JP6: 1-3 & 2-4 JP8: 1-3 & 2-4

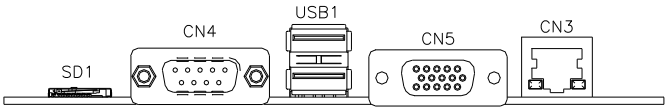
Connectors on IB892

Connector Locations on IB892 11
CN1: SATA Connector 12
CN2: CFAST Connector 12
CN3: Gigabit LAN RJ45 Connector 12
CN4: COM1 Serial Ports Connector 12
CN5: VGA Connector 12
SD1: Micro SD Connector 12
USB1: USB 0/1 Connector 12
J1: SPI Flash Connector (factory use only) 13
J2: Audio Connector 13
J4: LPC Connector (factory use only) 13
J5: System Function Connector 13
J6: HDD Power Connector 14
J8: LCD Backlight Connector 14
J9: COM2/RS232 Serial Port 14
J10: Power LED 14
J11: LVDS Connectors (DF13 Connector) 15
J12: COM3, COM4 Serial Port (DF11 Connector) 15
J13: Mini PCIE Connector 15
J14: DC-IN 12V Power Connector 15
J15: CAN BUS Connector 15
J16: Digital I/O Connector (4 in, 4 out) 16
J17: USB2/3 Connector 16

Connector Locations on IB892



Connectors on IB892 ..... Page
CN1: SATA Connector 12
CN2: CFAST Connector 12
CN3: Gigabit LAN RJ45 Connector 12
CN4: COM1 Serial Ports Connector 12
CN5: VGA Connector 12
SD1: Micro SD Connector 12
USB1: USB 0/1 Connector 12
J1: SPI Flash Connector (factory use only) 13
J2: Audio Connector 13
J4: LPC Connector (factory use only) 13
J5: System Function Connector 13
J6: HDD Power Connector 14
J8: LCD Backlight Connector 14
J9: COM2/RS232 Serial Port 14
J10: Power LED 14
J11: LVDS Connectors (DF13 Connector) 15
J12: COM3, COM4 Serial Port (DF11 Connector) 15
J13: Mini PCIE Connector 15
J14: DC-IN 12V Power Connector 15
J15: CAN BUS Connector 15
J16: Digital I/O Connector (4 in, 4 out) 16
J17: USB2/3 Connector 16

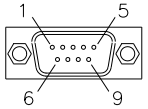


CN1: SATA Connectors

CN2: CFAST Connectors

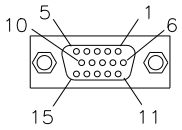
CN3: Gigabit LAN RJ45 Connector

CN4: COM1 Serial Ports Connector



Pin #	Signal Name		
	RS-232	RS-422	RS-485
1	DCD	TX-	DATA-
2	RX	TX+	DATA+
3	TX	RX+	NC
4	DTR	RX-	NC
5	Ground	Ground	Ground
6	DSR	NC	NC
7	RTS	NC	NC
8	CTS	NC	NC
9	RI	NC	NC

CN5: VGA Connector



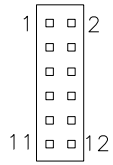
Signal Name	Pin #	Pin #	Signal Name
Red	1	2	Green
Blue	3	4	N.C.
GND	5	6	GND
GND	7	8	GND
VCC	9	10	GND
N.C.	11	12	DDCDATA
HSYNC	13	14	VSYNC
DDCCLK	15		

SD1: Micro SD Connector

USB1: USB 0/1 Connector

J1: SPI Flash Connector (factory use only)

J2: Audio Connector



Signal Name	Pin #	Pin #	Signal Name
LINE-OUT_L	1	2	LINE-OUT_R
JD-OUT	3	4	Ground
LINE-IN_L	5	6	LINE-IN R
JD-IN	7	8	Ground
Mic-In L	9	10	Mic-In R
JD-Mic	11	12	Ground

J4: LPC Connector (factory use only)

J5: System Function Connector

ATX Power ON Switch: Pins 1 and 2
This 2-pin connector is an "ATX Power Supply On/Off Switch" on the system that connects to the power switch on the case. When pressed, the power switch will force the system to power on. When pressed again, it will force the system to power off.

Hard Disk Drive LED Connector: Pins 3 and 4
This connector connects to the hard drive activity LED on control panel. This LED will flash when the HDD is being accessed.

Pin #	Signal Name
4	HDD Active
3	5V

Reset Switch: Pins 5 and 6
The reset switch allows the user to reset the system without turning the main power switch off and then on again. Orientation is not required when making a connection to this header.

+5V and 5VSB Signals: Pins 7 and 8

Pin #	Signal Name
7	+5V
8	+5VSB

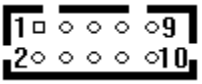
J6: HDD Power Connector

Pin #	Signal Name
1	+5V
2	Ground
3	Ground
4	+12V

J8: LCD Backlight Connector

Pin #	Signal Name
1	+12V
2	Backlight Enable
3	ADJ
4	Ground

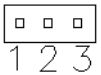
J9: COM2/RS232 Serial Port



Signal Name	Pin #	Pin #	Signal Name
DCD, Data carrier detect	1	2	RXD, Receive data
TXD, Transmit data	3	4	DTR, Data terminal ready
GND, ground	5	6	DSR, Data set ready
RTS, Request to send	7	8	CTS, Clear to send
RI, Ring indicator	9	10	Not Used

J10: Power LED

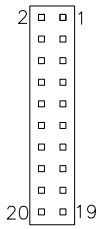
The power LED indicates the status of the main power switch.



Pin #	Signal Name
1	Power LED
2	No connect
3	Ground

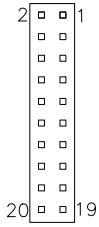
J11: LVDS Connectors (DF13 Connector)

The LVDS connector supports single-channel 18-bit or 24-bit displays.



Signal Name	Pin #	Pin #	Signal Name
TX0-	2	1	TX0+
Ground	4	3	Ground
TX1-	6	5	TX1+
5V/3.3V	8	7	Ground
TX3-	10	9	TX3+
TX2-	12	11	TX2+
Ground	14	13	Ground
TXC-	16	15	TXC+
5V/3.3V	18	17	ENABKL
+12V	20	19	+12V

J12: COM3, COM4 Serial Port (DF11 Connector)



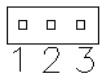
Signal Name	Pin #	Pin #	Signal Name
DSR3	2	1	DCD3
RTS3	4	3	RXD3
CTS3	6	5	TXD3
RI3	8	7	DTR3
NC	10	9	Ground
DSR4	12	11	DCD4
RTS4	14	13	RXD4
CTS4	16	15	TXD4
RI4	18	17	DTR4
NC	20	19	Ground

J13: Mini PCIE Connector

J14: DC-IN 12V Power Connector

Pin #	Signal Name
1	DC in (12V only)
2	Ground

J15: CAN BUS Connector

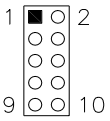


Pin #	Signal Name
1	CAN_H
2	CAN_L

3	Ground

J16: Digital I/O Connector (4 in, 4 out)

This 10-pin digital I/O connector supports TTL levels and is used to control external devices requiring ON/OFF circuitry.



Signal Name	Pin #	Pin #	Signal Name
Ground	1	2	+5V
Out3	3	4	Out1
Out2	5	6	Out0
IN3	7	8	IN1
IN2	9	10	IN0

J17: USB2/3 Connector

Signal Name	Pin	Pin	Signal Name
Vcc	1	2	Ground
USB2-	3	4	USB3+
USB2+	5	6	USB3-
Ground	7	8	Vcc

BIOS SETUP

This chapter describes the different settings available in the AMI (American Megatrends, Inc.) BIOS that comes with the board. The topics covered in this chapter are as follows:

BIOS Introduction 18
BIOS Setup 18
Main BIOS Setup 19
Advanced Settings 20
Chipset Settings 26
Audio Controller options 27
Boot Settings 28
Security Settings 29
Save & Exit Settings 30

BIOS Introduction

The BIOS (Basic Input/Output System) installed in your computer system's ROM supports Intel processors. The BIOS provides critical low-level support for a standard device such as disk drives, serial ports and parallel ports. It also password protection as well as special support for detailed fine-tuning of the chipset controlling the entire system.

BIOS Setup

The BIOS provides a Setup utility program for specifying the system configurations and settings. The BIOS ROM of the system stores the Setup utility. When you turn on the computer, the BIOS is immediately activated. Pressing the <Del> key immediately allows you to enter the Setup utility. If you are a little bit late pressing the <Del> key, POST (Power On Self Test) will continue with its test routines, thus preventing you from invoking the Setup. If you still wish to enter Setup, restart the system by pressing the "Reset" button or simultaneously pressing the <Ctrl>, <Alt> and <Delete> keys. You can also restart by turning the system Off and back On again. The following message will appear on the screen:

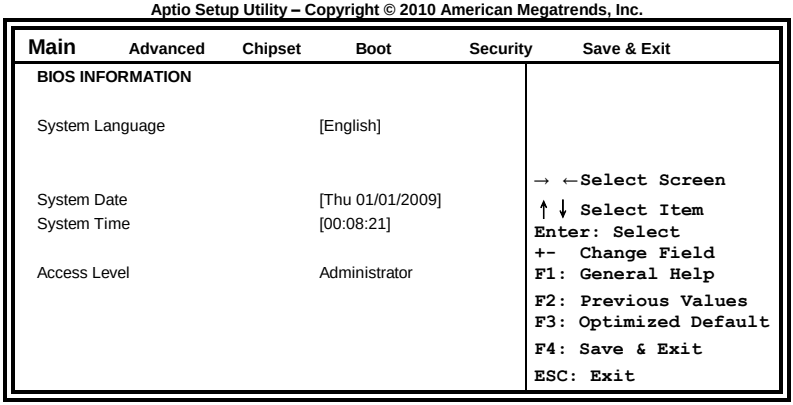
```
Press  <DEL> or <F2> to  Enter  Setup
```

In general, you press the arrow keys to highlight items, <Enter> to select, the <PgUp> and <PgDn> keys to change entries, <F1> for help and <Esc> to quit.

When you enter the Setup utility, the Main Menu screen will appear on the screen. The Main Menu allows you to select from various setup functions and exit choices.

Main BIOS Setup

This setup allows you to record some basic hardware configurations in your computer system and set the system clock.



***Note**: If the system cannot boot after making and saving system changes with Setup, the AMI BIOS supports an override to the CMOS settings that resets your system to its default.*

***Warning:** It is strongly recommended that you avoid making any changes to the chipset defaults. These defaults have been carefully chosen by both AMI and your system manufacturer to provide the absolute maximum performance and reliability. Changing the defaults could cause the system to become unstable and crash in some cases.*

System Language

Choose the system default language.

System Date

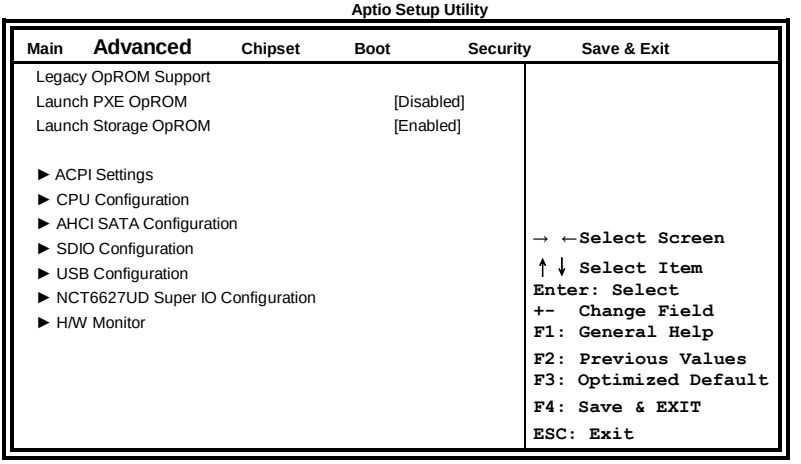
Set the Date. Use Tab to switch between Data elements.

System Time

Set the Time. Use Tab to switch between Data elements.

Advanced Settings

This section allows you to configure and improve your system and allows you to set up some system features according to your preference.



Launch PXE OpROM

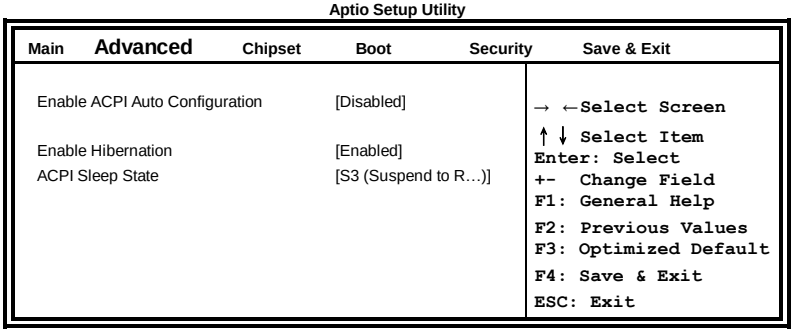
Enable or Disable Boot Option for Legacy Network Devices.

Launch Storage OpROM

Enable or Disable Boot Option for Legacy Mass Storage Devices with Option ROM.

ACPI Settings

System ACPI Parameters.



Enable ACPI Auto Configuration

Enables or Disables BIOS ACPI Auto Configuration.

Enable Hibernation

Enables or Disables System ability to Hibernate (OS/S4 Sleep State). This option may be not effective with some OS.

ACPI Sleep State

Select the highest ACPI sleep state the system will enter, when the SUSPEND button is pressed.

CPU Configuration

This section shows the CPU configuration parameters.

Aptio Setup Utility

Main	Advanced	Chipset	Boot	Security	Save & Exit
CPU Configuration					
Processor Type		Genuine Intel® CPU			
EMT64		Supported			
Processor Speed		1300 MHz			
System Bus Speed		400MHz			
Processor Stepping		20661			
Microcode Revision		260			
Processor Core		Single			
Hyper-Threading		Supported			
Intel SpeedStep		Enabled			
Hyper-Threading		All			
Execute Disable Bit		Disabled			
Limit CPUID Maximum		Enabled			
Intel Virtualization Technology		Enabled			
C-States		Enabled			
Enhanced C1		Disabled			
Enhanced C2		Enabled			
Enhanced C3		Disabled			
Enhanced C4		Disabled			

→ ←Select Screen
↑ ↓ Select Item
Enter: Select
+- Change Field
F1: General Help
F2: Previous Values
F3: Optimized Default
F4: Save & Exit
ESC: Exit

Intel SpeedStep

Enabled or Disable Intel® SpeedStep ™

Hyper-Threading

Enabled for Windows XP and Linux (OS optimized for Hyper-Threading Technology) and Disabled for other OS (OS not optimized for Hyper-Threading Technology). When Disabled, only one thread per enabled core is enabled.

Execute Disable Bit

XD can prevent certain classes of malicious buffer overflow attacks when combined with a supporting OS (Windows Server 2003 SP1, Windows XP SP2, SuSE Linux 9.2, RedHat Enterprise 3 Update 3.)

Limit CPUID Maximum

Disabled for Windows XP.

Intel Virtualization Technology

When enabled, a VMM can utilize the additional hardware capabilities provided by Vanderpool Technology.

C-States

Enable or Disable C2 and above

AHCI SATA Configuration

AHCI SATA Device Options Settings

Aptio Setup Utility

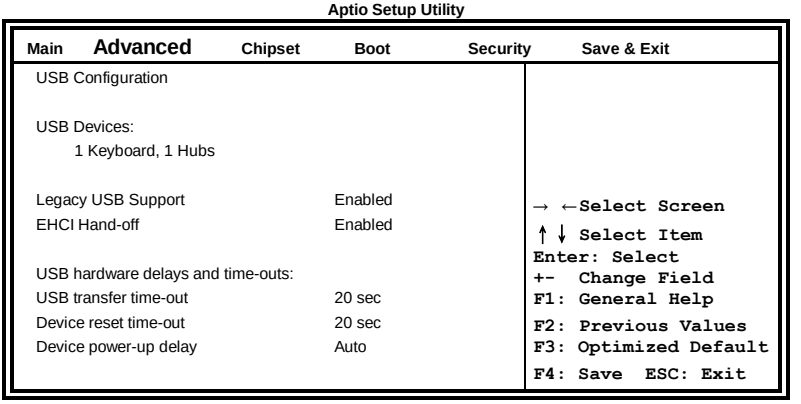
Main	Advanced	Chipset	Boot	Security	Save & Exit
AHCI SATA Configuration					
PORT 0		Enabled			Enable / Disable PORT 0 Set transfer mode programming
PORT 1		Enabled			
					→ ←Select Screen ↑ ↓ Select Item Enter: Select +- Change Field F1: General Help F2: Previous Values F3: Optimized Default F4: Save & Exit ESC: Exit

SDIO Configuration

Main	Advanced	Chipset	Boot	Security	Save & Exit
SDIO Access Mode			Auto		Auto option: Access SD device in DMA mode if controller supports it, otherwise in PIO mode. DMA option: Access SD device in DMA mode. PIO option: Access SD device in PIO mode

USB Configuration

USB Configuration Parameters.



Legacy USB Support

Enables Legacy USB support.
AUTO option disables legacy support if no USB devices are connected. DISABLE option will keep USB devices available only for EFI applications.

EHCI Hand-off

This is a workaround for OSes without EHCI hand-off support. The EHCI ownership change should be claimed by EHCI driver.

USB transfer time-out

The time-out value for Control, Bulk, and Interrupt transfers.

Device reset time-out

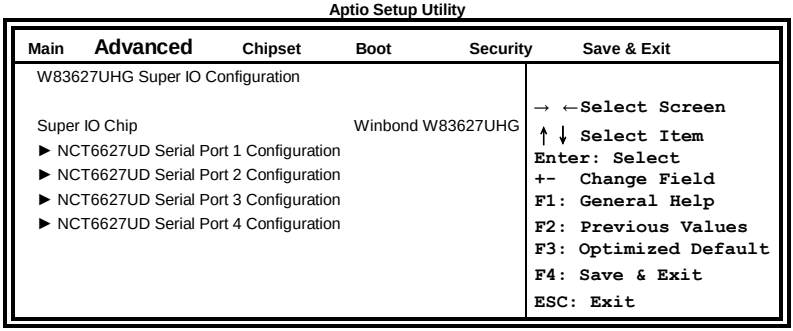
USB mass storage device Start Unit command time-out.

Device power-up delay

Maximum time the device will take before it properly reports itself to the Host Controller. 'Auto' uses default value: for a Root port it is 100 ms, for a hub port the delay is taken from Hub Descriptor.

NCT6627UD Super IO Configuration

System Super IO Chip Parameters.

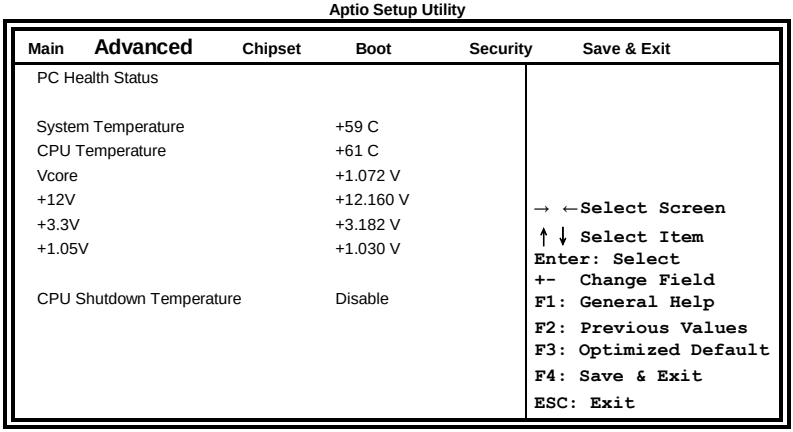


Serial Port Configuration

Set Parameters of Serial Ports. User can Enable/Disable the serial port and Select an optimal settings for the Super IO Device.

H/W Monitor

Monitor hardware status.



Temperatures/Voltages

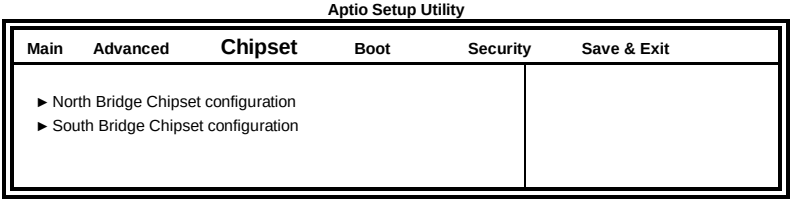
These fields are the parameters of the hardware monitoring function feature of the motherboard. The values are read-only values as monitored by the system and show the PC health status.

CPU Shutdown Temperature

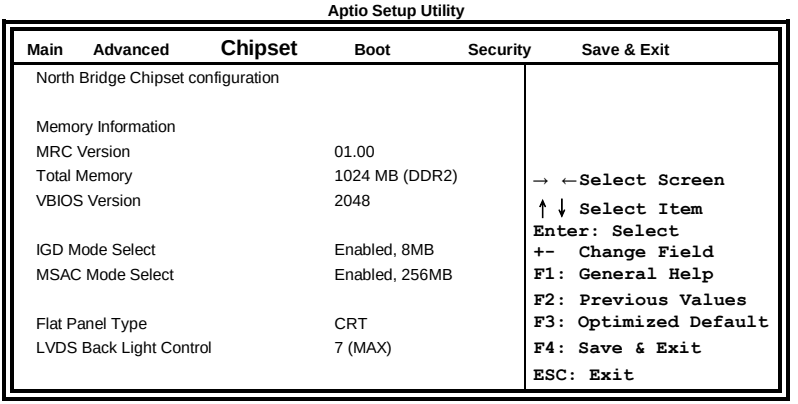
This field enables (70C/75C/80C/85C/90C/95C) or disables the CPU Shutdown Temperature.

Chipset Settings

This section allows you to configure and improve your system and allows you to set up some system features according to your preference.



North Bridge Chipset configuration



Flat Panel Type

This field options are: CRT/ CRT+LVDS 640*480 18Bit / CRT+LVDS 800*600 18Bit / CRT+LVDS 1024*768 24 Bit / CRT+LVDS 1366*768 24 Bit.

LVDS Back Light Control

Select the LFP Panel Inverted voltage: 0(Min)~7(Max)

South Bridge Chipset configuration

Aptio Setup Utility

Main	Advanced	Chipset	Boot	Security	Save & Exit

South Bridge Chipset configuration

Audio Controller Auto

High Precision Event Timer Configuration

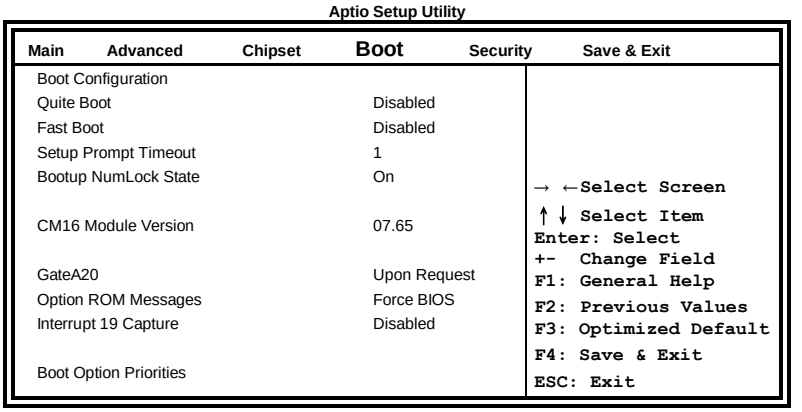
High Precision Timer Enabled

→ ←Select Screen
↑ ↓ Select Item
Enter: Select
+- Change Field
F1: General Help
F2: Previous Values
F3: Optimized Default
F4: Save & Exit
ESC: Exit

Audio Controller

Audio Controller options

Boot Settings



Setup Prompt Timeout

Number of seconds to wait for setup activation key. 65535(0xFFFF) means indefinite waiting.

GateA20 Active

UPON REQUEST – GA20 can be disabled using BIOS services.
ALWAYS – do not allow disabling GA20; this option is useful when any RT code is executed above 1MB.

Option ROM Messages

Set display mode for Option ROM. Options are Force BIOS and Keep Current.

Interrupt 19 Capture

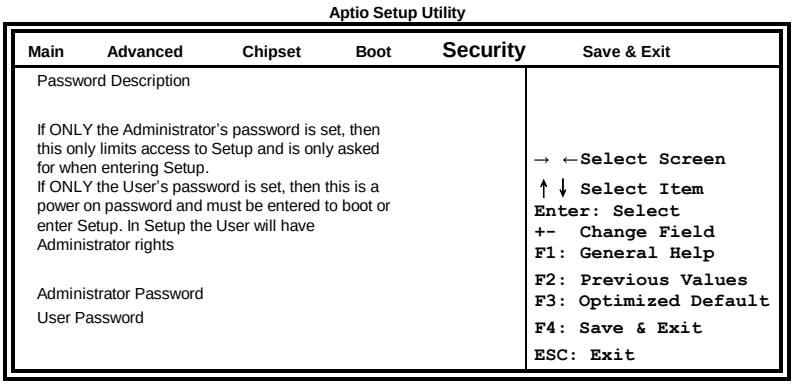
Enable: Allows Option ROMs to trap Int 19.

Boot Option Priorities

Sets the system boot order.

Security Settings

This section allows you to configure and improve your system and allows you to set up some system features according to your preference.



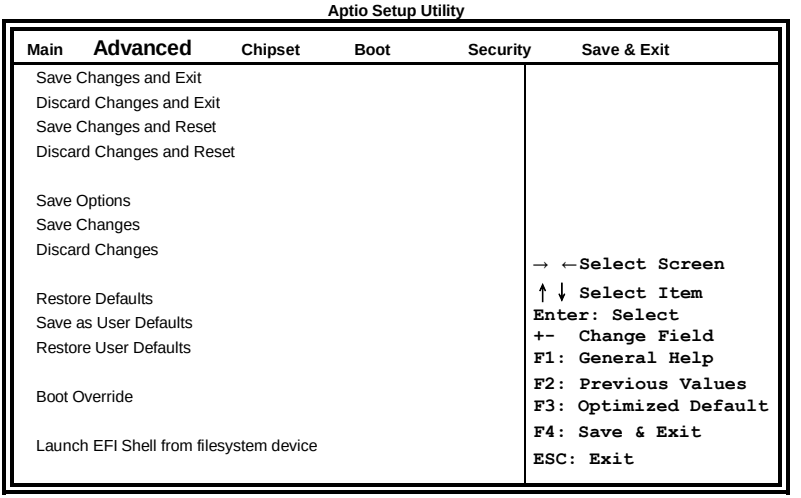
Administrator Password

Set Setup Administrator Password.

User Password

Set User Password.

Save & Exit Settings



Save Changes and Exit
Exit system setup after saving the changes.

Discard Changes and Exit
Exit system setup without saving any changes.

Save Changes and Reset
Reset the system after saving the changes.

Discard Changes and Reset
Reset system setup without saving any changes.

Save Changes
Save Changes done so far to any of the setup options.

Discard Changes
Discard Changes done so far to any of the setup options.

Restore Defaults
Restore/Load Defaults values for all the setup options.

Save as User Defaults
Save the changes done so far as User Defaults.

Restore User Defaults
Restore the User Defaults to all the setup options.

Boot Override
Pressing ENTER causes the system to enter the OS.

Launch EFI Shell from filesystem device
Attempts to Launch EFI Shell application (Shellx64.efi) from one of the available filesystem devices.

This page is intentionally left blank.

Drivers Installation

This section describes the installation procedures for software and drivers under the **Windows XP**. The software and drivers are included with the motherboard. If you find the items missing, please contact the vendor where you made the purchase. The contents of this section include the following:

Intel Chipset Software Installation Utility 34
Intel Pineview Chipset Family Graphics Driver Installation 36
Realtek High Definition Codec Audio Driver Installation 38
Intel 82574L LAN Drivers Installation 39

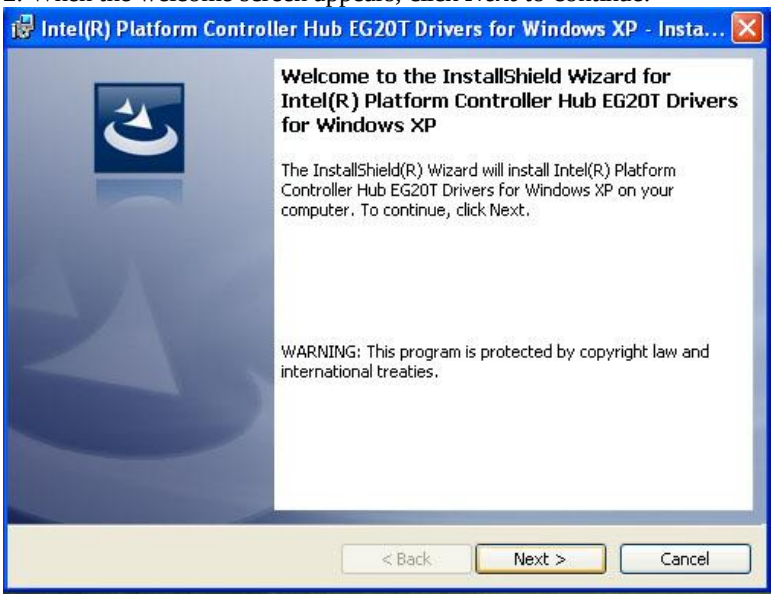
IMPORTANT NOTE:
After installing your Windows operating system (Windows XP/ Vista/ 7), you must install first the Intel Chipset Software Installation Utility before proceeding with the drivers installation.

Intel Chipset Software Installation Utility

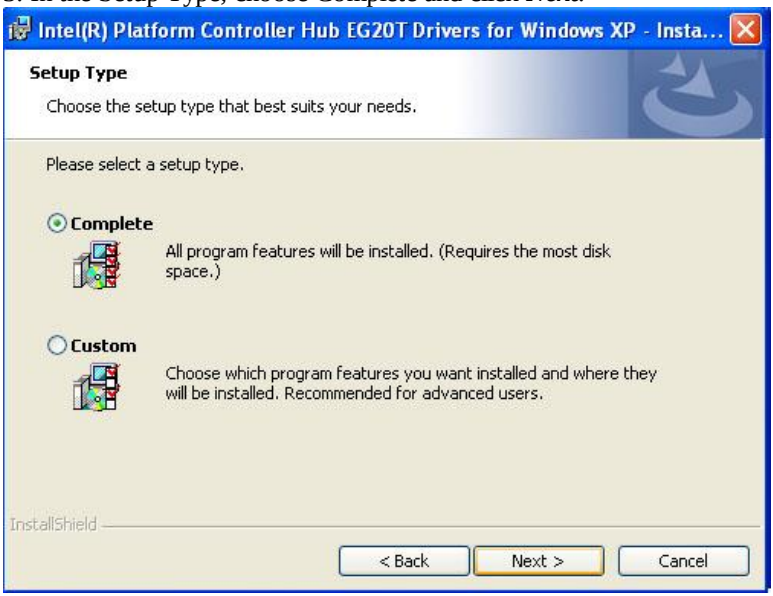
The Intel Chipset Drivers should be installed first before the software drivers to enable Plug & Play INF support for Intel chipset components. Follow the instructions below to complete the installation.

1. Insert the drivers DVD into the DVD drive. Click ***Intel*** and then ***Intel(R)Chipset Software Installation Utility***.

2. When the welcome screen appears, click ***Next*** to continue.



3. In the Setup Type, choose Complete and click ***Next***.



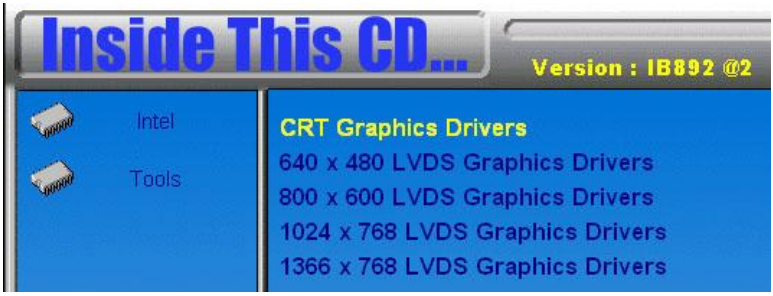
4. Now click **Continue Anyway** to continue the installation.

5. The InstallShield Wizard has finished installing the Intel Platform controller Hub EG20T Drivers. Click ***Finish*** to exit the wizard.

Intel Pineview Chipset Family Graphics Driver Installation

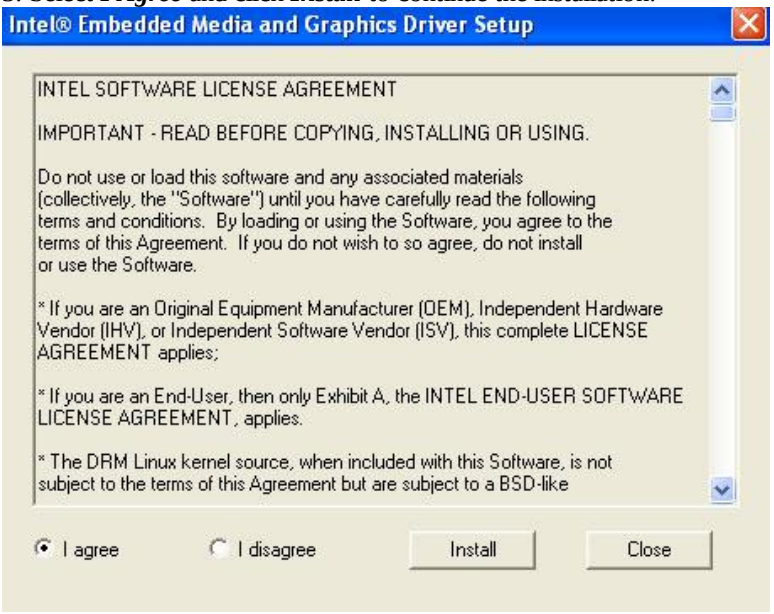
To install the VGA drivers, follow the steps below to proceed with the installation.

1. Insert the drivers DVD into the DVD drive. Click ***Intel*** and then ***Intel(R) EG20T Graphics Driver***. Then click ***CRT Graphics Drivers***.



2. Click ***Next*** to continue.

3. Select ***I Agree*** and click ***Install*** to continue the installation.



4. Click ***Next*** in the Readme File Information window.

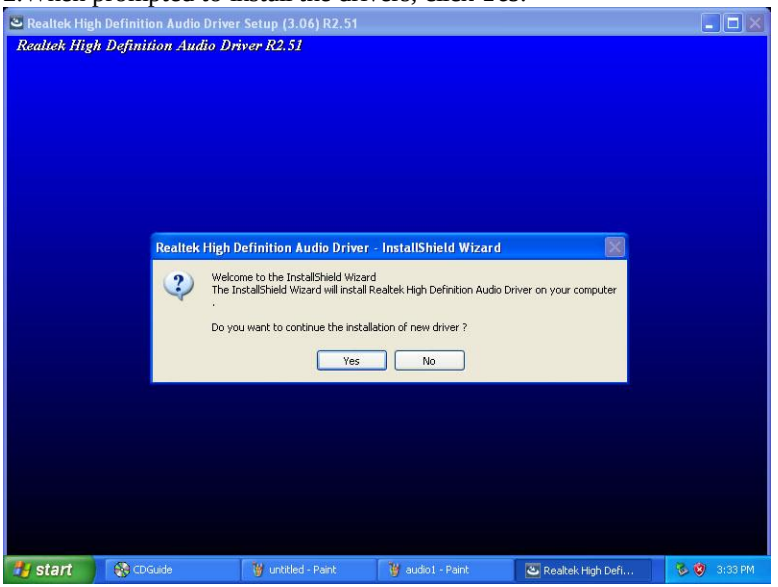
5. Restart the computer when prompted.

Realtek High Definition Codec Audio Driver Installation

Follow the steps below to install the Realtek HD Codec Audio Drivers.

1. Insert the drivers DVD into the DVD drive. Click ***Intel*** and then ***Realtek High Definition Codec Audio Driver.***

2.When prompted to install the drivers, click ***Yes***.



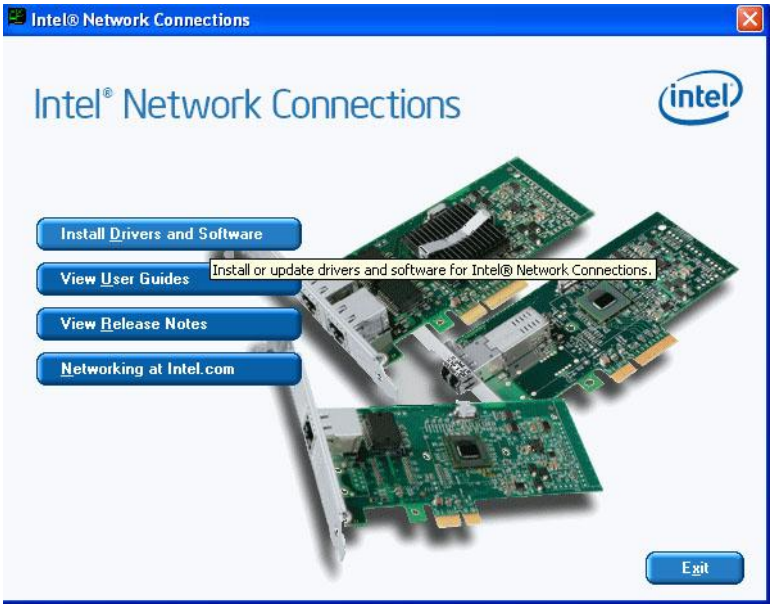
3.When the InstallShieldWizard has finished the installation, restart the computer when prompted.

Intel 82574L LAN Drivers Installation

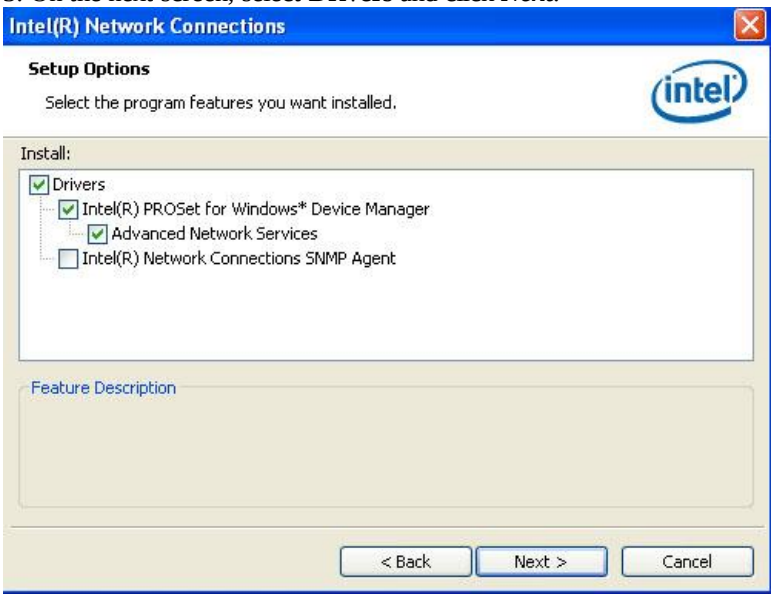
Follow the steps below to install Intel 82574L LAN Drivers.

1. Insert the drivers DVD into the DVD drive. Click ***Intel*** and then ***Intel(R) PRO LAN Network Drivers***.

2. On the next screen, select **Install Drivers and Software**.



3. On the next screen, select **Drivers** and click ***Next***.



4. InstallShield Wizard completed. Click ***Finish*** to exit the Wizard.

Appendix

A. I/O Port Address Map

Each peripheral device in the system is assigned a set of I/O port addresses that also becomes the identity of the device. The following table lists the I/O port addresses used.

Address	Device Description
000h - 01Fh	DMA Controller #1
020h - 03Fh	Interrupt Controller #1
040h - 05Fh	Timer
060h - 06Fh	Keyboard Controller
070h - 07Fh	Real Time Clock, NMI
080h - 09Fh	DMA Page Register
0A0h - 0BFh	Interrupt Controller #2
0C0h - 0DFh	DMA Controller #2
0F0h	Clear Math Coprocessor Busy Signal
0F1h	Reset Math Coprocessor
1F0h - 1F7h	IDE Interface
2B0h - 2DFh	Graphics adapter Controller
2E8h - 2EFh	Serial Port #4(COM4)
2F8h - 2FFh	Serial Port #2(COM2)
360h - 36Fh	Network Ports
3B0h - 3BFh	Monochrome & Printer adapter
3C0h - 3CFh	EGA adapter
3D0h - 3DFh	CGA adapter
3E8h - 3EFh	Serial Port #3(COM3)
3F8h - 3FFh	Serial Port #1(COM1)

B. Interrupt Request Lines (IRQ)

Peripheral devices use interrupt request lines to notify CPU for the service required. The following table shows the IRQ used by the devices on board.

Level	Function
IRQ0	System Timer Output
IRQ1	Keyboard
IRQ2	Interrupt Cascade
IRQ3	Serial Port #2
IRQ4	Serial Port #1
IRQ5	Reserved
IRQ6	Reserved
IRQ7	Reserved
IRQ8	Real Time Clock
IRQ9	Reserved
IRQ10	Serial Port #4
IRQ11	Serial Port #3
IRQ12	PS/2 Mouse
IRQ13	80287
IRQ14	Primary IDE

C. Watchdog Timer Configuration

The WDT is used to generate a variety of output signals after a user programmable count. The WDT is suitable for use in the prevention of system lock-up, such as when software becomes trapped in a deadlock. Under these sorts of circumstances, the timer will count to zero and the selected outputs will be driven. Under normal circumstance, the user will restart the WDT at regular intervals before the timer counts to zero.

SAMPLE CODE:

```
File of the W627UHG.CPP
//-----------------------------------------------------------------------------
//
// THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR
// PURPOSE.
//
//-----------------------------------------------------------------------------
#include "W627UHG.H"
#include <dos.h>
//-----------------------------------------------------------------------------
unsigned int W627UHG_BASE;
void Unlock_W627UHG (void);
void Lock_W627UHG (void);
//-----------------------------------------------------------------------------
unsigned int Init_W627UHG(void)
{
        unsigned int result;
        unsigned char ucDid;

        W627UHG_BASE = 0x4E;
        result = W627UHG_BASE;

        ucDid = Get_W627UHG_Reg(0x20);
        if (ucDid == 0xA2)                              //W83627UHG??
        {       goto Init_Finish;    }

        W627UHG_BASE = 0x2E;
        result = W627UHG_BASE;

        ucDid = Get_W627UHG_Reg(0x20);
        if (ucDid == 0xA2)                              //W83627UHG??
        {       goto Init_Finish;    }

        W627UHG_BASE = 0x00;
        result = W627UHG_BASE;

Init_Finish:
        return (result);
}
//-----------------------------------------------------------------------------
void Unlock_W627UHG (void)
```

```
{
        outportb(W627UHG_INDEX_PORT, W627UHG_UNLOCK);
        outportb(W627UHG_INDEX_PORT, W627UHG_UNLOCK);
}
//-----------------------------------------------------------------------
void Lock_W627UHG (void)
{
        outportb(W627UHG_INDEX_PORT, W627UHG_LOCK);
}
//-----------------------------------------------------------------------
void Set_W627UHG_LD( unsigned char LD)
{
        Unlock_W627UHG();
        outportb(W627UHG_INDEX_PORT, W627UHG_REG_LD);
        outportb(W627UHG_DATA_PORT, LD);
        Lock_W627UHG();
}
//-----------------------------------------------------------------------
void Set_W627UHG_Reg( unsigned char REG, unsigned char DATA)
{
        Unlock_W627UHG();
        outportb(W627UHG_INDEX_PORT, REG);
        outportb(W627UHG_DATA_PORT, DATA);
        Lock_W627UHG();
}
//-----------------------------------------------------------------------
unsigned char Get_W627UHG_Reg(unsigned char REG)
{
        unsigned char Result;
        Unlock_W627UHG();
        outportb(W627UHG_INDEX_PORT, REG);
        Result = inportb(W627UHG_DATA_PORT);
        Lock_W627UHG();
        return Result;
}
//-----------------------------------------------------------------------
```

File of the W627UHG.H

```
//---------------------------------------------------------------------------
//
// THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR
// PURPOSE.
//
//---------------------------------------------------------------------------
#ifndef __W627UHG_H
#define __W627UHG_H                1
//---------------------------------------------------------------------------
#define       W627UHG_INDEX_PORT            (W627UHG_BASE)
#define       W627UHG_DATA_PORT             (W627UHG_BASE+1)
//---------------------------------------------------------------------------
#define       W627UHG_REG_LD                0x07
//---------------------------------------------------------------------------
#define W627UHG_UNLOCK                0x87
#define       W627UHG_LOCK                  0xAA
//---------------------------------------------------------------------------
unsigned int Init_W627UHG(void);
void Set_W627UHG_LD( unsigned char);
void Set_W627UHG_Reg( unsigned char, unsigned char);
unsigned char Get_W627UHG_Reg( unsigned char);
//---------------------------------------------------------------------------
#endif        //__W627UHG_H
```

File of the MAIN.CPP

```
//---------------------------------------------------------------------------
//
// THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR
// PURPOSE.
//
//---------------------------------------------------------------------------
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include "W627UHG.H"
//---------------------------------------------------------------------------
int main (void);

void WDTInitial(void);
void WDTEnable(unsigned char);
void WDTDisable(void);

//---------------------------------------------------------------------------
int main (void)
{
        char SIO;

        SIO = Init_W627UHG();
        if (SIO == 0)
        {
        .....printf("Can not detect Winbond 83627UHG, program abort.\n");
        ..... return(1);
        }

        WDTInitial();

        WDTEnable(10);

        WDTDisable();

        return 0;
}
//---------------------------------------------------------------------------
void WDTInitial(void)
{
        unsigned char bBuf;
        Set_W627UHG_LD(0x08);..... //switch to logic device 8
        bBuf = Get_W627UHG_Reg(0x30);
        bBuf &= (~0x01);
        Set_W627UHG_Reg(0x30, bBuf);..... //Enable WDTO
}
//---------------------------------------------------------------------------
void WDTEnable(unsigned char NewInterval)
{
        unsigned char bBuf;

        Set_W627UHG_LD(0x08);.....
        Set_W627UHG_Reg(0x30, 0x01);..... //enable timer
```

```
        bBuf = Get_W627UHG_Reg(0xF5);
        bBuf &= (~0x08);
        Set_W627UHG_Reg(0xF5, bBuf);.....//count mode is second

        Set_W627UHG_Reg(0xF6, NewInterval); .....//set timer
}
//-------------------------------------------------------------------------
void WDTDisable(void)
{
        Set_W627UHG_LD(0x08);.....
        Set_W627UHG_Reg(0xF6, 0x00);.....//clear watchdog timer
        Set_W627UHG_Reg(0x30, 0x00);.....
}
//-------------------------------------------------------------------------
```

D. Digital I/O Sample Code

File of the W627UHG.H

```
//---------------------------------------------------------------------------
//
// THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR
// PURPOSE.
//
//---------------------------------------------------------------------------
#ifndef __W627UHG_H
#define __W627UHG_H                1
//---------------------------------------------------------------------------
#define       W627UHG_INDEX_PORT            (W627UHG_BASE)
#define       W627UHG_DATA_PORT             (W627UHG_BASE+1)
//---------------------------------------------------------------------------
#define       W627UHG_REG_LD                0x07
//---------------------------------------------------------------------------
#define W627UHG_UNLOCK                0x87
#define       W627UHG_LOCK                  0xAA
//---------------------------------------------------------------------------
unsigned int Init_W627UHG(void);
void Set_W627UHG_LD( unsigned char);
void Set_W627UHG_Reg( unsigned char, unsigned char);
unsigned char Get_W627UHG_Reg( unsigned char);
//---------------------------------------------------------------------------
#endif        //__W627UHG_H
```

File of the W627UHG.CPP

```
//---------------------------------------------------------------------------
//
// THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR
// PURPOSE.
//
//---------------------------------------------------------------------------
#include "W627UHG.H"
#include <dos.h>
//---------------------------------------------------------------------------
unsigned int W627UHG_BASE;
void Unlock_W627UHG (void);
void Lock_W627UHG (void);
//---------------------------------------------------------------------------
unsigned int Init_W627UHG(void)
{
        unsigned int result;
        unsigned char ucDid;

        W627UHG_BASE = 0x4E;
        result = W627UHG_BASE;

        ucDid = Get_W627UHG_Reg(0x20);
        if (ucDid == 0xA2)                                      //W83627UHG??
        {       goto Init_Finish;    }

        W627UHG_BASE = 0x2E;
        result = W627UHG_BASE;

        ucDid = Get_W627UHG_Reg(0x20);
        if (ucDid == 0xA2)                                      //W83627UHG??
        {       goto Init_Finish;    }

        W627UHG_BASE = 0x00;
        result = W627UHG_BASE;

Init_Finish:
        return (result);
}
//---------------------------------------------------------------------------
void Unlock_W627UHG (void)
{
        outportb(W627UHG_INDEX_PORT, W627UHG_UNLOCK);
        outportb(W627UHG_INDEX_PORT, W627UHG_UNLOCK);
}
//---------------------------------------------------------------------------
void Lock_W627UHG (void)
{
        outportb(W627UHG_INDEX_PORT, W627UHG_LOCK);
}
//---------------------------------------------------------------------------
void Set_W627UHG_LD( unsigned char LD)
```

```
{
        Unlock_W627UHG();
        outportb(W627UHG_INDEX_PORT, W627UHG_REG_LD);
        outportb(W627UHG_DATA_PORT, LD);
        Lock_W627UHG();
}
//----------------------------------------------------------------------------
void Set_W627UHG_Reg( unsigned char REG, unsigned char DATA)
{
        Unlock_W627UHG();
        outportb(W627UHG_INDEX_PORT, REG);
        outportb(W627UHG_DATA_PORT, DATA);
        Lock_W627UHG();
}
//----------------------------------------------------------------------------
unsigned char Get_W627UHG_Reg(unsigned char REG)
{
        unsigned char Result;
        Unlock_W627UHG();
        outportb(W627UHG_INDEX_PORT, REG);
        Result = inportb(W627UHG_DATA_PORT);
        Lock_W627UHG();
        return Result;
}
//----------------------------------------------------------------------------
```

File of the MAIN.CPP

```
//---------------------------------------------------------------------------
//
// THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR
// PURPOSE.
//
//---------------------------------------------------------------------------
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include "W627UHG.H"
//---------------------------------------------------------------------------
int main (void);

void Dio5Initial(void);
void Dio5SetOutput(unsigned char);
unsigned char Dio5GetInput(void);
void Dio5SetDirection(unsigned char);
unsigned char Dio5GetDirection(void);
//---------------------------------------------------------------------------
int main (void)
{
	char SIO;

	SIO = Init_W627UHG();
	if (SIO == 0)
	{
		printf("Can not detect Winbond 83627UHG, program abort.\n");
		return(1);
	}

	Dio5Initial();

	//for GPIO50..57
	Dio5SetDirection(0x0F); //GP50..53 = input, GP54..57=output
	printf("Current DIO direction = 0x%X\n", Dio5GetDirection());

	printf("Current DIO status = 0x%X\n", Dio5GetInput());

	printf("Set DIO output to high\n");
	Dio5SetOutput(0x0F);

	printf("Set DIO output to low\n");
	Dio5SetOutput(0x00);

	return 0;
```

```
}
//---------------------------------------------------------------------------
void Dio5Initial(void)
{
	unsigned char ucBuf;

	Set_W627UHG_LD(0x08);                                   //switch to logic device 8
	//enable the GP5 group
	ucBuf = Get_W627UHG_Reg(0x30);
	ucBuf |= 0x02;
	Set_W627UHG_Reg(0x30, ucBuf);
}
//---------------------------------------------------------------------------
void Dio5SetOutput(unsigned char NewData)
{
	Set_W627UHG_LD(0x08);                                   //switch to logic device 8
	Set_W627UHG_Reg(0xE1, NewData);
}
//---------------------------------------------------------------------------
unsigned char Dio5GetInput(void)
{
	unsigned char result;

	Set_W627UHG_LD(0x08);                                   //switch to logic device 8
	result = Get_W627UHG_Reg(0xE1);
	return (result);
}
//---------------------------------------------------------------------------
void Dio5SetDirection(unsigned char NewData)
{
	//NewData : 1 for input, 0 for output
	Set_W627UHG_LD(0x08);                                   //switch to logic device 8
	Set_W627UHG_Reg(0xE0, NewData);
}
//---------------------------------------------------------------------------
unsigned char Dio5GetDirection(void)
{
	unsigned char result;

	Set_W627UHG_LD(0x08);                                   //switch to logic device 8
	result = Get_W627UHG_Reg(0xE0);
	return (result);
}
//---------------------------------------------------------------------------
```