FWA6404 Series

Networking Appliance

User's Manual

Version: 1.1

Table of Contents

Foreword

To prevent damage to the system board, please handle it with care and follow the measures below, which are generally sufficient to protect your equipment from static electricity discharge:
When handling the board, use a grounded wrist strap designed for static discharge elimination grounded to a metal object before removing the board from the antistatic bag. Handle the board by its edges only; do not touch its components, peripheral chips, memory modules or gold contacts.
When handling processor chips or memory modules, avoid touching their pins or gold edge fingers.
Return the Network Appliance system board and peripherals back into the antistatic bag when not in use or not installed in the chassis.
Some circuitry on the system board can continue to operate even though the power is switched off.
Under no circumstances should the Lithium battery cell used to power the real-time clock be allowed to be shorted. The battery cell may heat up under these conditions and present a burn hazard.

WARNING!

"CAUTION: DANGER OF EXPLOSION IF BATTERY IS INCORRECTLY REPLACED. REPLACE ONLY WITH SAME OR EQUIVALENT TYPE RECOMMENDED BY THE MANUFACTURER. DISCARD USED BATTERIES ACCORDING TO THE MANUFACTURER'S INSTRUCTIONS"
This guide is for technically qualified personnel who have experience installing and configuring system boards. Disconnect the system board power supply from its power source before you connect/disconnect cables or install/remove any system board components. Failure to do this can result in personnel injury or equipment damage.
Avoid short-circuiting the lithium battery; this can cause it to superheat and cause burns if touched.
Do not operate the processor without a thermal solution. Damage to the processor can occur in seconds.
Do not block air vents at least minimum 1/2-inch clearance required.

Chapter 1 Introduction

FWA6404 series was specifically designed for the network security & management market.

Network Security Applications:
• Firewall
• Unified Threat Management (UTM)
• Virtual Private Network (VPN)
• Proxy Server
• Caching Server

Network Management Applications:
• Load balancing
• Quality of Service
• Remote Access Service

The FWA networking appliance product line covers the spectrum from offering platforms designed for:
• SOHO
• SMB
• Enterprise

Each product is designed to address the distinctive requirements of its respective market segment from cost effective entry-level solutions to high throughput and performance-bound systems for the Enterprise level.

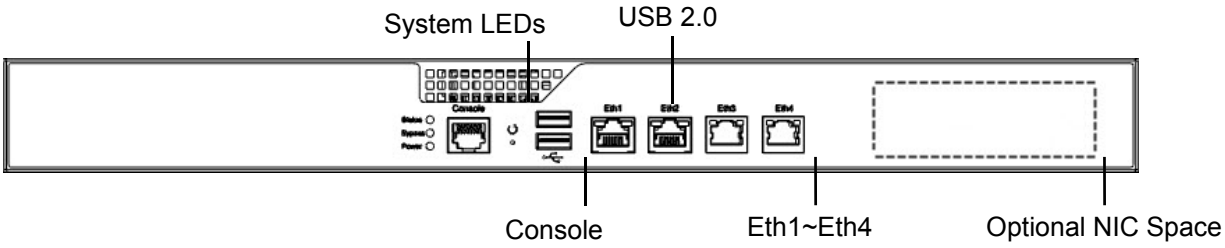
Chapter 2 System Specification

<table>
<tr><td>Product Name</td><td>FWA6404</td></tr>
<tr><td>Form Factor</td><td>19” 1U Mainstream Networking Product</td></tr>
<tr><td>Motherboard</td><td>MB838</td></tr>
<tr><td>Processor</td><td>Support Intel® Rangeley processors:
<table>
<tr><th></th><th>Core#</th><th>SoC</th><th>TDP</th><th>QuickAssist</th></tr>
<tr><td>MB838-2C</td><td>2</td><td>Intel® Atom™ Processor C2358 (1M Cache, 1.70 GHz)</td><td>7W</td><td>Yes</td></tr>
<tr><td>MB838-4C</td><td>4</td><td>Intel® Atom™ Processor C2558 (2M Cache, 2.40 GHz)</td><td>15W</td><td>Yes</td></tr>
<tr><td>MB838-8C</td><td>8</td><td>Intel® Atom™ Processor C2758 (4M Cache, 2.40 GHz)</td><td>20W</td><td>Yes</td></tr>
</table></td></tr>
<tr><td>BIOS</td><td>AMI BIOS</td></tr>
<tr><td>Memory</td><td>Two DDR3 DDR3/L UDIMM total for 32GB max memory
Dual channel DDR3 up to 1600 MHz
Unbuffered
ECC or non-ECC
Support 1.35V and 1.5V</td></tr>
<tr><td>Display</td><td>NA</td></tr>
<tr><td>TPM</td><td>TPM 1.2 (INFINEON SLB9655TT1.2)</td></tr>
<tr><td>Network</td><td>Eth1~2: Intel® I347-AT4 GbE PHY. One segment Bypass
Eth3~4: Intel® I347-AT4 GbE PHY. No Bypass</td></tr>
<tr><td>Bypass</td><td>One segment hardware Bypass (Eth1 & 2)
Bypass mode selection (BIOS)</td></tr>
<tr><td>NIC Slot</td><td>Support one NIC Slot for IBP161~IBP167</td></tr>
<tr><td>Expansion Slot</td><td>One PCIe x4 Slot at Rear Panel
Mini PCI-e Socket (m-SATA compatible)</td></tr>
<tr><td>Storage</td><td>One internal 2.5” HDD</td></tr>
<tr><td>Front Panel</td><td>Three LEDs for Power, Bypass & Status
Console (RJ45, COM1)
Factory Mode Restore Reset Switch
USB 2.0 x2
Four RJ-45 connectors for Eth1~4</td></tr>
<tr><td>Rear Panel</td><td>One Expansion Slot
Two System Fans
PSU inlet</td></tr>
<tr><td>Watchdog Timer</td><td>256 segments, 0, 1, 2...255 sec/min</td></tr>
<tr><td>Power Supply</td><td>Full Range 150W Power Supply</td></tr>
</table>

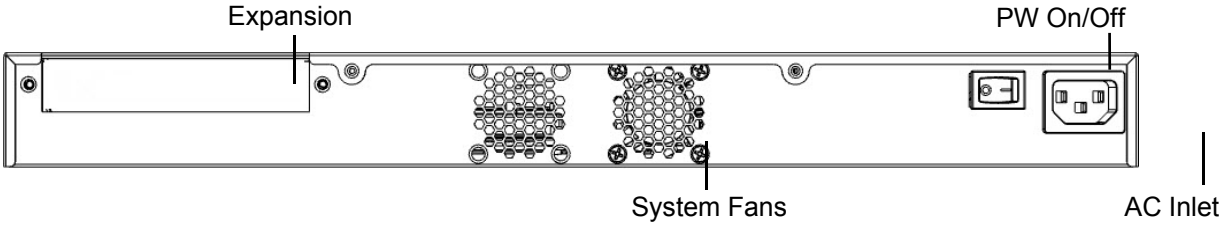
Dimensions	440 (W) x 219 (D) x 44 (H) mm
Operation Temperature	0 ~ 45 ˚C
Storage Temperature	-20 ~ 70 ˚C
Operation Humidity	5% ~ 90%
Certifications	CE, FCC
Compatible Front NIC Cards	IBP161: 4-port RJ-45 10/100/1000 Copper LAN Module Card IBP162: 2-port 10 GbE SFP+ LAN Module Card IBP163: 2+2 ports GbE Copper or SFP LAN Module Card IBP164: Crypto Acceleration Card IBP165: 4-port RJ-45 10/100/1000 Copper LAN Module Card IBP167: 8-port RJ-45 10/100/1000 Copper LAN Module Card

Chapter 3 Hardware Configuration

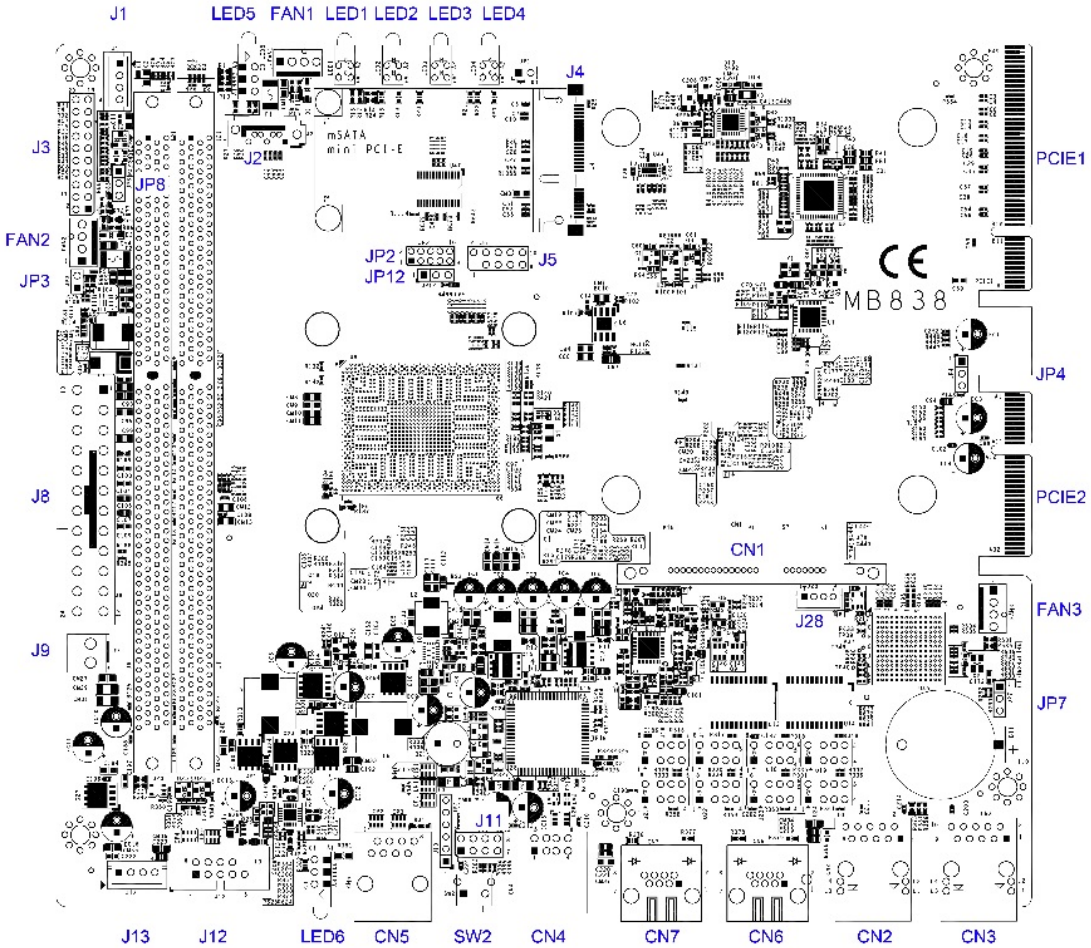
Front Panel Features



Rear Panel Features



Jumper and Connector Locations on MB838



Jumper Settings on MB838

JP2: Function reserved

JP3: DDR power selection

JP3	DDR3 power
Short (Default)	1.5V
Open	1.35V

JP4: AT / ATX mode selection

JP4	AT/ATX
1-2 Short	ATX
2-3 Short (Default)	AT

JP7: Clear CMOS

JP7	CMOS
1-2 Short (Default)	Normal
2-3 Short	Clear CMOS

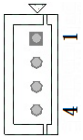
JP8: LED5, LED6 Bypass or HDD LED selection

JP8	CMOS
1-2 Short	HDD
2-3 Short (Default)	LAN Bypass

JP12: Function reserved

Connectors on MB838

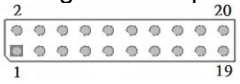
J1, J13: Power Connector, Pitch 2.54mm



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

J2: 7-pin SATA connector

J3: System Function Connector
J3 provides connectors for system indicators that provide light indication of the computer activities and switches to change the computer status. J13 is a 20-pin header that provides interfaces for the following functions



Pin 1, 2: Speaker
This connector provides an interface to a speaker for audio tone generation. An 8-ohm speaker is recommended.

Pin #	Signal Name
1	+5V
2	GND

Pin 13, 14: ATX Power ON Switch
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.

Pin #	Signal Name
13	GND
14	Power_ON

Pin 15, 16: Power LED
The power LED indicates the status of the main power switch.

Pin #	Signal Name
15	+5V
16	GND

Pin 17, 18: Reset Switch
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.

Pin #	Signal Name
17	GND
18	PM_SYSRST#

Pin 19, 20: HDD LED:

Pin #	Signal Name
19	+5V
20	GND

J4: Mini PCI-e (supports mSATA)

J5: Function reserved

J8: Power connector for ATX power supply

J9: 2-pin Power connector for 12V DC power

Pin #	Signal Name
1	+12V
2	GND

J11: USB pin header

Signal Name	Pin #	Pin #	Signal Name
5V	1	2	NC
D-	3	4	NC
D+	5	6	NC
GND	7	8	NC

J12: COM2 pin header

Signal Name	Pin #	Pin #	Signal Name
DCD	1	6	DSR
RXD	2	7	RTS
TXD	3	8	CTS
DTR	4	9	RI
GND	5		

J28: Power Connector, Pitch 2.0mm

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

CN1: 22-pin right angle SATA connector

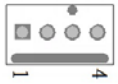
CN2, CN3: RJ-45 LAN port 3 and 4

CN4: USB 2.0 connectors

CN5: COM1 (Console Port)

CN6, CN7: RJ-45 LAN port 1 and 2

FAN1, FAN2, FAN3: System Fan Connectors
FAN1, FAN2, FAN3 is a 4-pin header for system fans.
The fan must be 12V (Max. 1A).



Pin #	Signal Name
1	Ground
2	+12V
3	Rotation detection
4	Rotation control

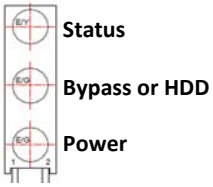
LED1: LAN port 1 LED

LED2: LAN port 2 LED

LED3: LAN port 3 LED

LED4: LAN port 4 LED

LED5, 6: Status LED
A1 & C1 : Status LED
A2 & C2 : Bypass or HDD status LED
A3 & C3 : Power LED



Signal Name	Pin #	Pin #	Signal Name
SIO_GPIO 33	A1	C1	SIO_GPIO 32
+5 V	A2	C2	JP15 Selection
+3.3 V	A3	C3	GND

SW2: Push button (GPI, porting by software)

SIGNAL NAME	Pin #	Pin #	Signal Name
SIO_GPIO33	A1	C1	SIO_GPIO32
+5 V	A2	C2	JP15 Selection
+3.3 V	A3	C3	GND

Chapter 4 Console Mode Information

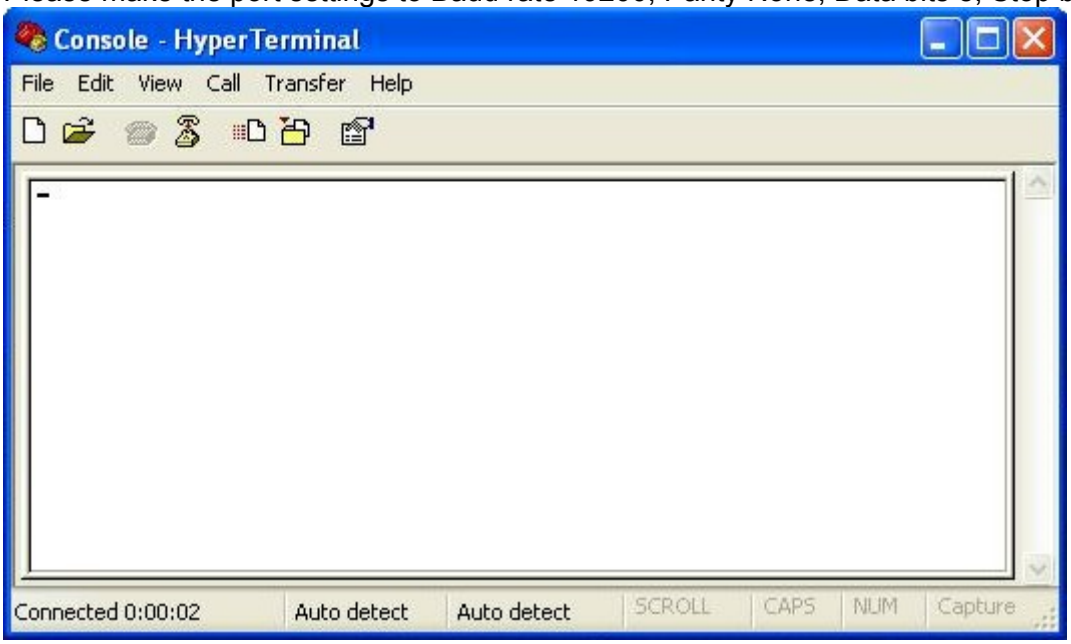
FWA6404 supports output information via Console in BIOS level.

Prepare a computer as client loaded with an existing OS such as Windows 7.
Connect client computer and FWA6404 with NULL Modem cable.
Follow the steps below to configure the Windows Hyper Terminal application setting:

Execute Hyper Terminal. Issue command “hypertrm”.
Customize your name for the new connection.

Choose COM port on the client computer for the connection.

Please make the port settings to Baud rate 19200, Parity None, Data bits 8, Stop bits 1



Power on FWA6404.
Press <**Tab**> key to enter BIOS setup screen in **Console mode**.
Press <**Del**> key to enter BIOS setup screen in **VGA mode**.

Open the Chassis

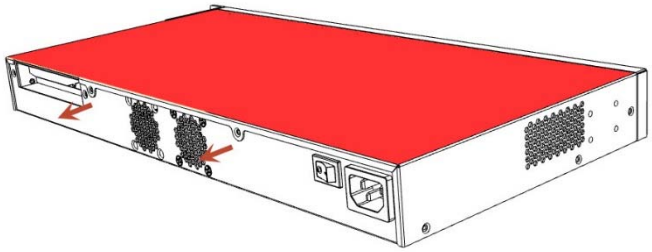


Fig. 5-1 Loosen screws and remove the cover

Installing DDR3 Memory

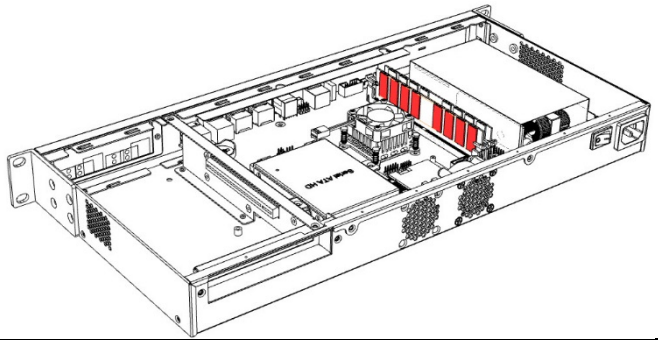


Fig. 5-2 Align module to the memory slot and push module in an upright position until clips of the slot close to hold the module.

Installing 2.5" HDD/SSD

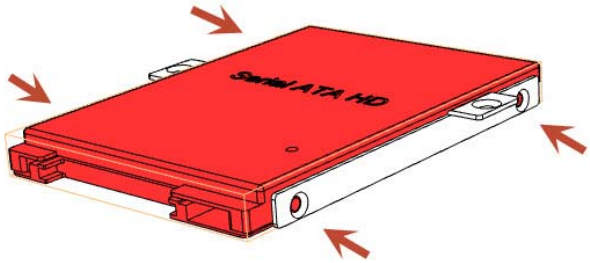


Fig. 5-3 Fasten the screws with bracket

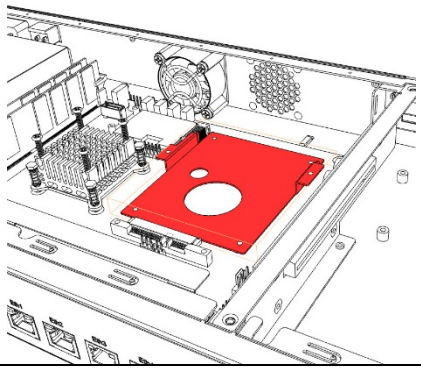


Fig. 5-4 Fasten the HDD/SSD with base bracket

Chapter 6 BIOS Information

BIOS Introduction

The BIOS (Basic Input/Output System) installed in your computer system's ROM provides critical low-level support for a standard device such as disk drives, serial ports and parallel ports. It also adds virus and 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:

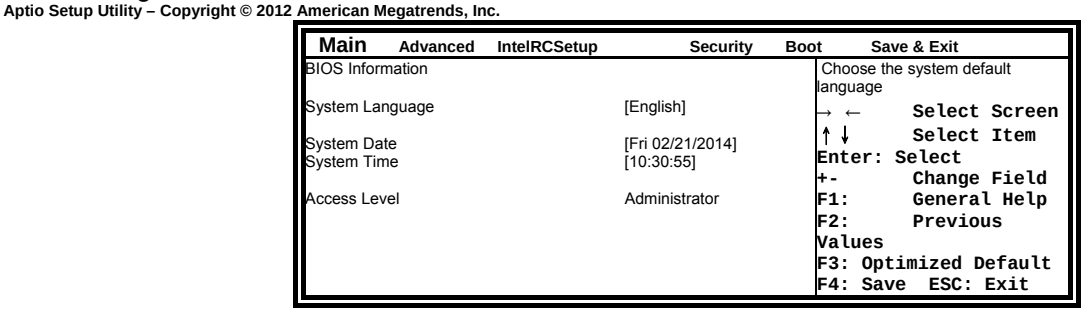
Press <DEL> or <F2> to Enter Setup

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.

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.*

Main Settings



System Language

Choose the system default language.

System Date

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

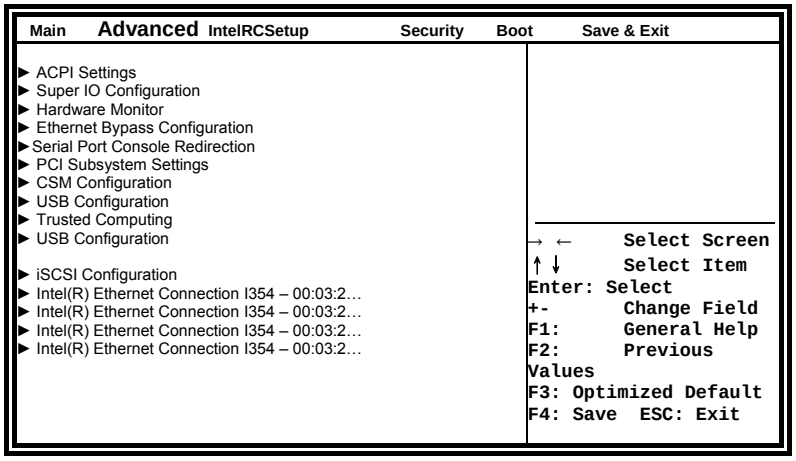
System Time

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

Advanced Settings

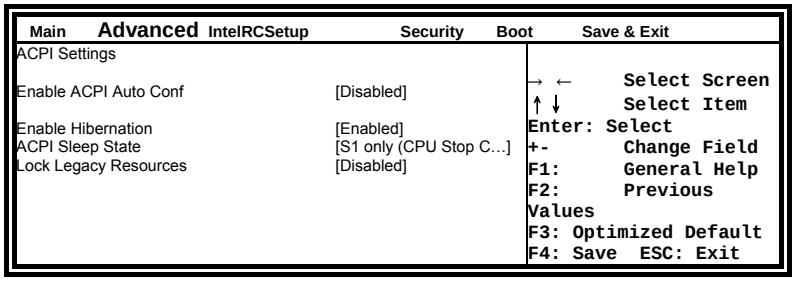
This section allows you to configure and improve your system and allows you to set up some system features according to your preference.

Aptio Setup Utility



ACPI Settings

Aptio Setup Utility



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

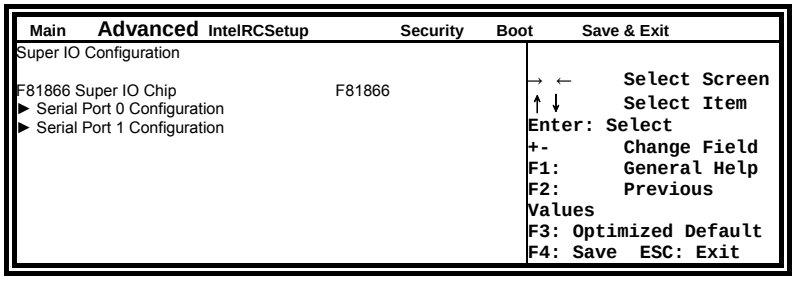
Select ACPI sleep state the system will enter, when the SUSPEND button is pressed.

Lock Legacy Resources

Enabled or Disabled Lock of Legacy Resources.

Super IO Configuration

Aptio Setup Utility

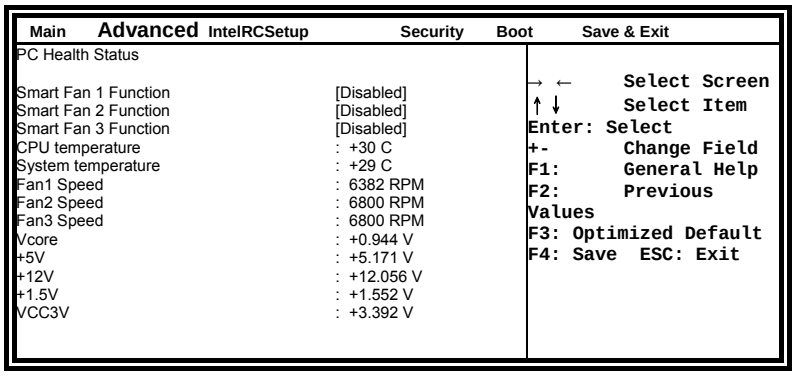


Serial Port Configuration

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

Hardware Monitor

Aptio Setup Utility

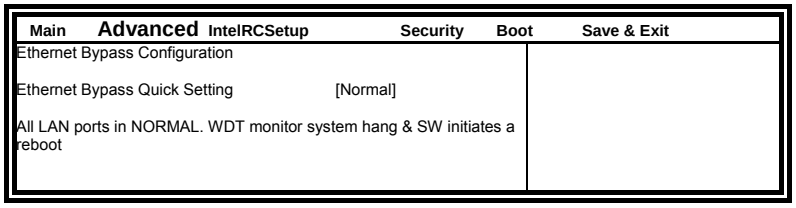


Main	Advanced	IntelRCSetup	Security	Boot	Save & Exit
PC Health Status					
Smart Fan 1 Function		[Disabled]			
Smart Fan 2 Function		[Disabled]			
Smart Fan 3 Function		[Disabled]			
CPU temperature		: +30 C			
System temperature		: +29 C			
Fan1 Speed		: 6382 RPM			
Fan2 Speed		: 6800 RPM			
Fan3 Speed		: 6800 RPM			
Vcore		: +0.944 V			
+5V		: +5.171 V			
+12V		: +12.056 V			
+1.5V		: +1.552 V			
VCC3V		: +3.392 V			

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

Ethernet Bypass Configuration

Aptio Setup Utility



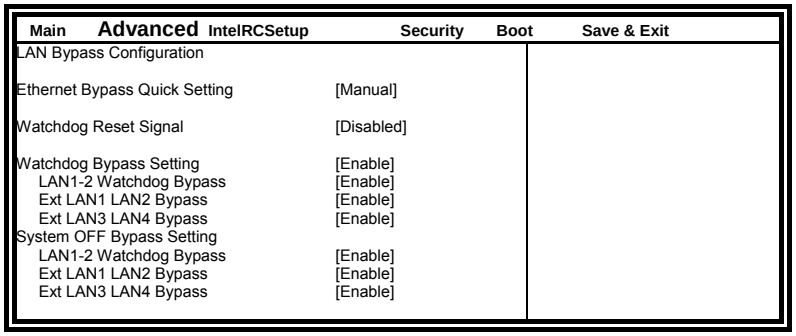
Main	Advanced	IntelRCSetup	Security	Boot	Save & Exit
Ethernet Bypass Configuration					
Ethernet Bypass Quick Setting		[Normal]			
All LAN ports in NORMAL. WDT monitor system hang & SW initiates a reboot					

Bypass Quick Setting

Set LAN Bypass to Normal, Bypass, Firewall or Custom Define Mode

Normal mode: All LAN ports in NORMAL. When Watchdog monitor system hangs, software will initiates a system reboot.
Bypass mode: All LAN ports in BYPASS during power-off or watchdog initiates Bypass. System will not reboot.
Firewall mode: All LAN ports in BYPASS until software change it to NORMAL under OS. When watchdog monitors system hang, software will initiates a system reboot.
Manual mode: Customer defines watchdog reset, watchdog Bypass and power-off Bypass settings.

Aptio Setup Utility

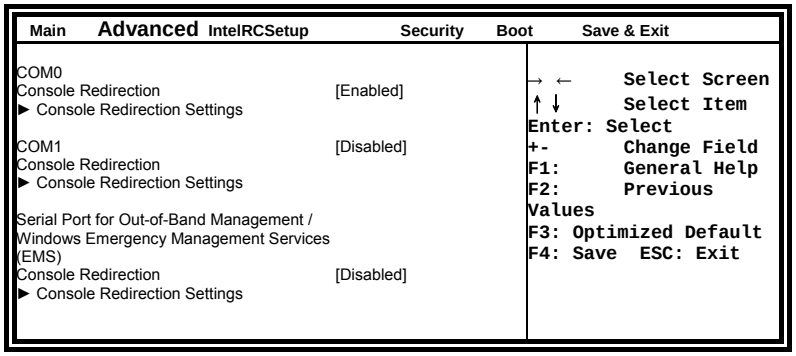


Main	Advanced	IntelRCSetup	Security	Boot	Save & Exit
LAN Bypass Configuration					
Ethernet Bypass Quick Setting		[Manual]			
Watchdog Reset Signal		[Disabled]			
Watchdog Bypass Setting		[Enable]			
LAN1-2 Watchdog Bypass		[Enable]			
Ext LAN1 LAN2 Bypass		[Enable]			
Ext LAN3 LAN4 Bypass		[Enable]			
System OFF Bypass Setting					
LAN1-2 Watchdog Bypass		[Enable]			
Ext LAN1 LAN2 Bypass		[Enable]			
Ext LAN3 LAN4 Bypass		[Enable]			

Note: "Ext LAN Bypass" items only appear when extended IBASE LAN module card installed.

Serial Port Console Redirection

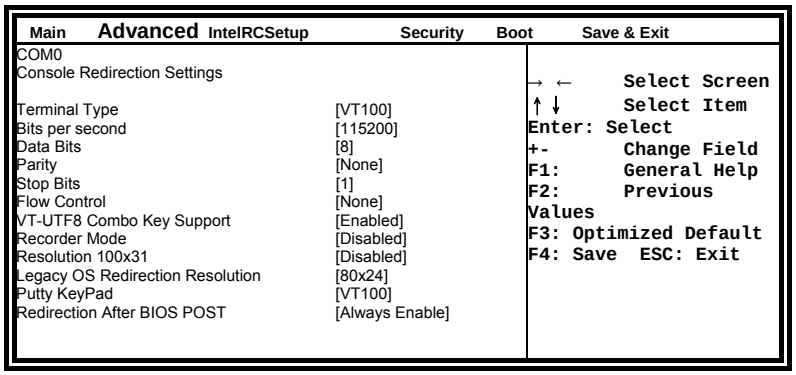
Aptio Setup Utility



Main	Advanced	IntelRCSetup	Security	Boot	Save & Exit
COM0					
Console Redirection		[Enabled]			
► Console Redirection Settings					
COM1		[Disabled]			
Console Redirection					
► Console Redirection Settings					
Serial Port for Out-of-Band Management / Windows Emergency Management Services (EMS)					
Console Redirection		[Disabled]			
► Console Redirection Settings					

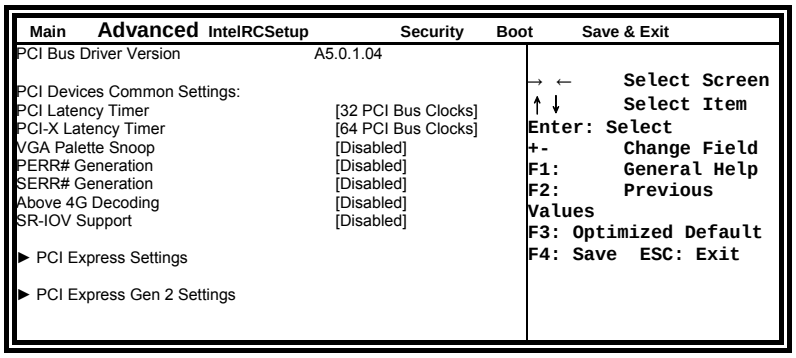
→ ← Select Screen
↑ ↓ Select Item
Enter: Select
+- Change Field
F1: General Help
F2: Previous Values
F3: Optimized Default
F4: Save ESC: Exit

Aptio Setup Utility



PCI Subsystem Settings

Aptio Setup Utility



PCI Latency Timer

Value to be programmed into PCI Latency Timer Register.

VGA Palette Snoop

Enables or disables VGA Palette Registers Snooping.

PERR# Generation

Enables or disables PCI device to generate PERR#.

SERR# Generation

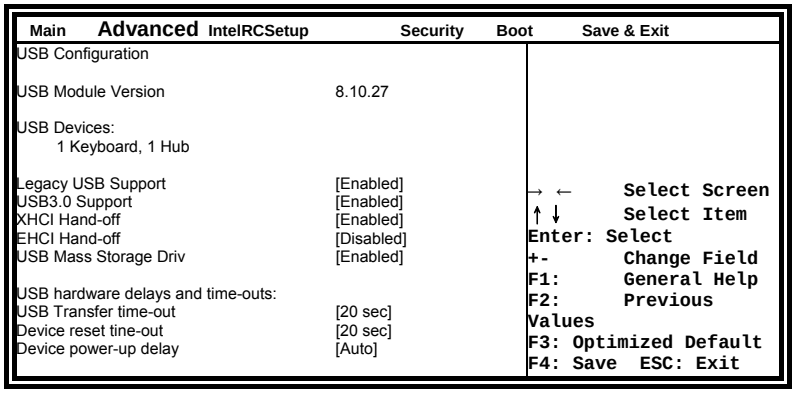
Enables or disables PCI device to generate SERR#.

PCI Express Settings

Change PCI Express devices settings.

USB Configuration

Aptio Setup Utility



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.

USB3.0 Support

Enable/Disable USB3.0 (XHCI) Controller support.

XHCI Hand-off

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

EHCI Hand-off

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

Port 64/60 Emulation

Enables I/O port 60h/64h emulation support. This should be enabled for the complete USB keyboard legacy support for non-USB aware OSes.

USB Transfer time-out

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

Device reset tine-out

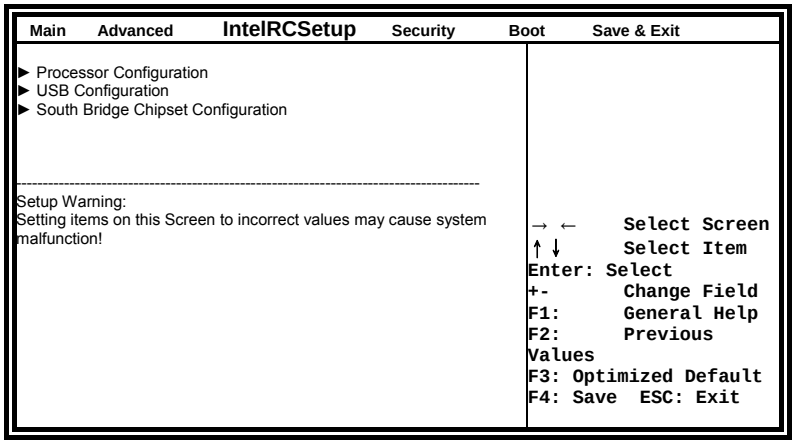
USB mass Storage device start Unit command time-out.

Device power-up delay

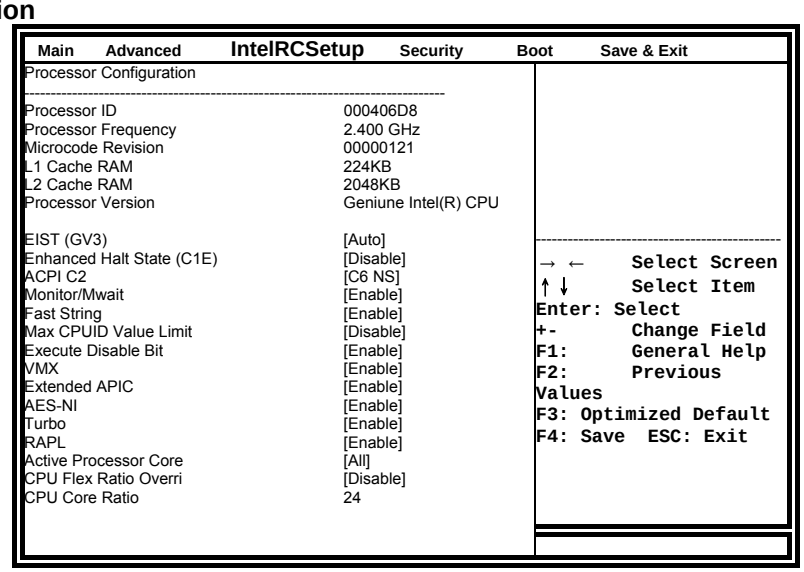
Maximum time the device will take before it properly reports itself to the Host Controller. 'Auto' uses default value: for a Root port it is 100ms, for a Hub port the delay is taken from Hub descriptor.

IntelRCSetup

Aptio Setup Utility



Processor Configuration



EIST (GV3)

Enable/Disable EIST. GV3 and TM1 must be enabled for TM2 to be available. GV3 must be enabled for Turbo. Auto – Enable for B0 CPU stepping, all others disabled, change setting to override.

Fast String

When enabled, enable fast string for REP MOVS/STOS.

MAX CPUID Value Limit
This should be enabled in order to boot legacy OSes that cannot support CPUs with extended CPUID functions.

Execute Disable Bit
When disabled, force the XD feature flag to always return 0.

VMX
Enables the Vanderpool Technology, takes effect after reboot.

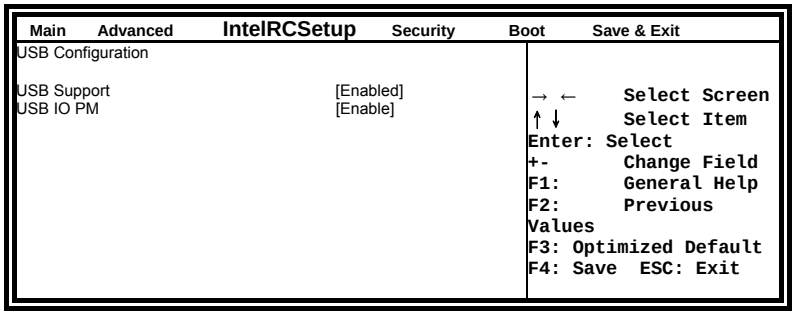
Extended APIC
Enable/Disable extended APIC support.
AES-NI
Enable/Disable AES-NI support

Turbo
Enable or Disable CPU Turbo capability. This option only applies to ES2 and above.

Active Processor Core
Number of cores to enable in SoC package.

CPU Flex Ratio Override
Enable/Disable CPU Flex Ration Programming.

USB Configuration



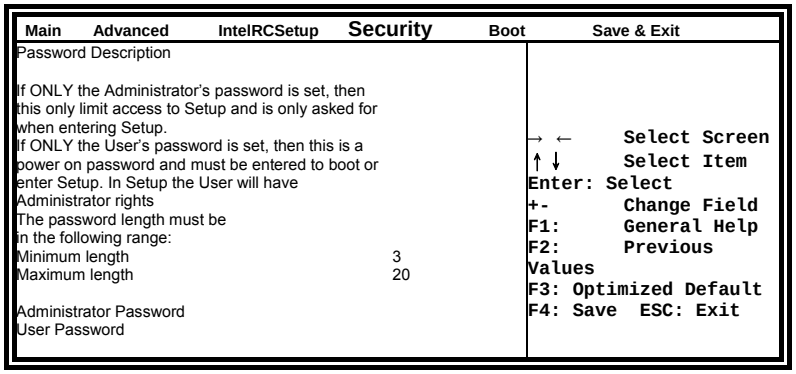
USB Support
USB Support Parameters.

USB IO PM
Enable/Disable IO PM.

Security

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

Aptio Setup Utility

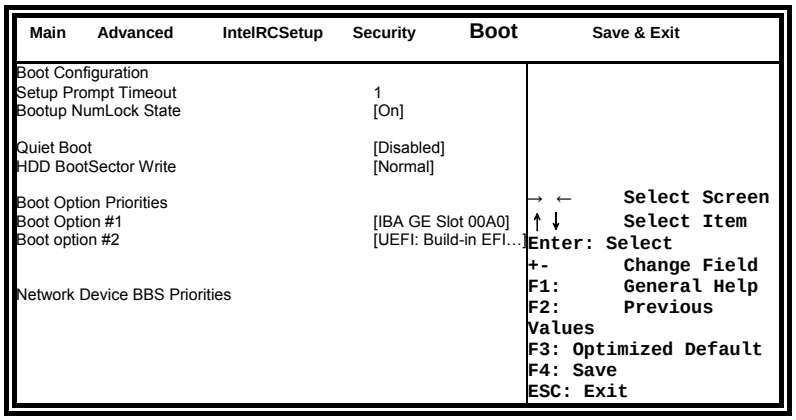


Administrator Password
Set Setup Administrator Password.

User Password
Set User Password.

Boot Settings

Aptio Setup Utility



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

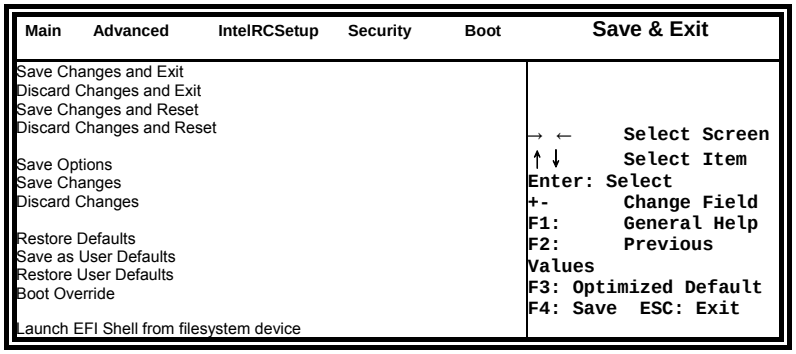
Bootup NumLock State
Select the keyboard NumLock state.

Quiet Boot
Enables/Disables Quiet Boot option.

Boot Option Priorities
Sets the system boot order.

Save & Exit Settings

Aptio Setup Utility



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.

Chapter 7 Drivers Installation

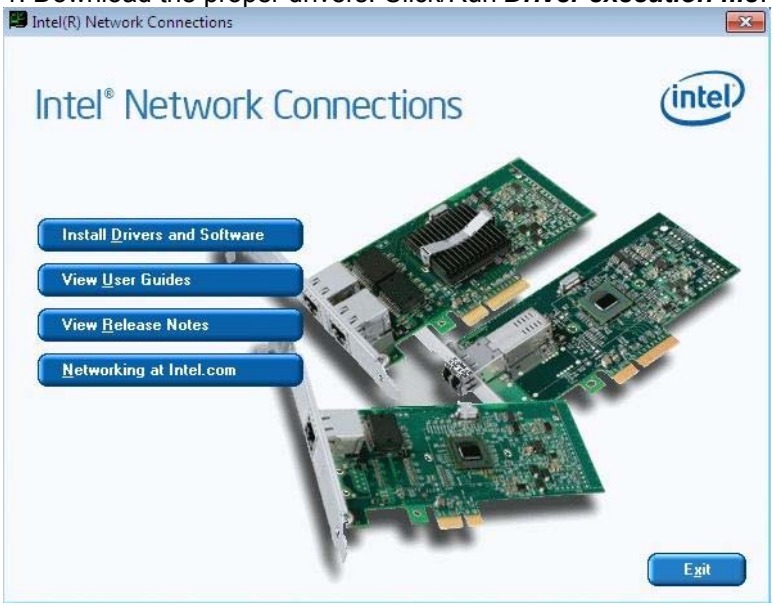
This section describes the installation procedures for software and drivers under the Windows. The software and drivers are included with the board. If you find the items missing, please contact the vendor where you made the purchase. The contents of this section include the following:

LAN Drivers Installation

IMPORTANT NOTE:
After installing your Windows operating system, you must install first the Intel® Chipset Software Installation Utility before proceeding with the drivers installation.

LAN Drivers Installation

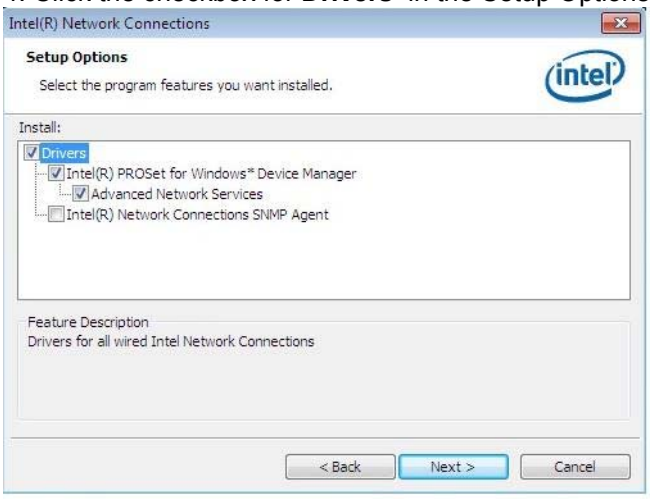
1. Download the proper drivers. Click/Run ***Driver execution file***.



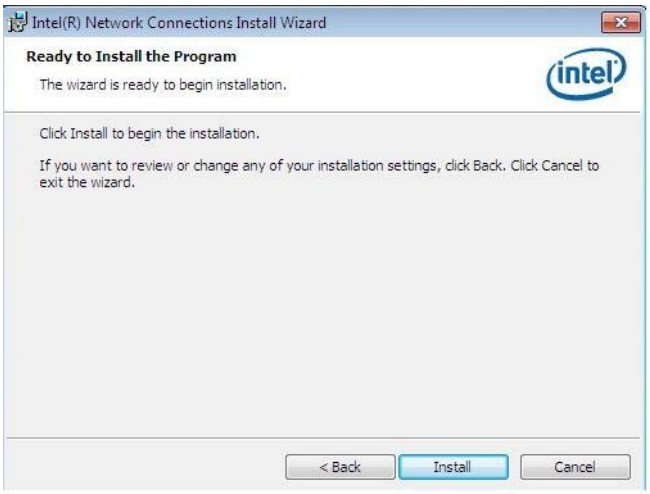
2. When the Welcome screen appears, click ***Next***.

3. Click ***Next*** to to agree with the license agreement.

4. Click the checkbox for **Drivers** in the Setup Options screen to select it and click **Next** to continue.



5. The wizard is ready to begin installation. Click ***Install*** to begin the installation.



Filename：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 "F81865.H"

#define BIT0 0x01
#define BIT1 0x02
#define BIT2 0x04
#define BIT3 0x08
#define BIT4 0x10
#define BIT5 0x20
#define BIT6 0x40
#define BIT7 0x80

//---------------------------------------------------------------------------
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;
        unsigned char DIO;

        printf("Fintek 81865/81866 digital I/O test program\n");

        SIO = Init_F81865();
        if (SIO == 0)
        {
                printf("Can not detect Fintek 81865/81866, program abort.\n");
                return(1);
        }//if (SIO == 0)

        Dio5Initial();
/*
        //for GPIO50..57
        Dio5SetDirection(0xF0);     //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);
*/

        //for GPIO50..57
        Dio5SetDirection(0xF0);     //GP50..53 = input, GP54..57=output

        Dio5SetOutput(0x00);                                    //clear
//      DIO = Dio5GetInput() & 0x0F;
```

```
		Dio5SetOutput(0x00);					//clear
		DIO = Dio5GetInput() & 0x0F;
		if (DIO != 0x0A)
		{
			printf("The Fintek 81865 digital IO abnormal, abort.\n");
			return(1);
		}//if (DIO != 0x0A)

		Dio5SetOutput(0xA0);					//clr# is high
		Dio5SetOutput(0xF0);					//clk and clr# is high
		Dio5SetOutput(0xA0);					//clr# is high

		DIO = Dio5GetInput() & 0x0F;
		if (DIO != 0x05)
		{
			printf("The Fintek 81865 digital IO abnormal, abort.\n");
			return(1);
		}
		printf("!!! Pass !!!\n");
		return 0;
}
//-----------------------------------------------------------------------------
void Dio5Initial(void)
{
		unsigned char ucBuf;

		//switch GPIO multi-function pin for	gpio 50~57

//gpio53~57 UR5_FULL_EN(bit1), clear UR6_FULL_EN(bit3)
		//set UR5_FULL_EN,should set UR_GP_PROG_EN = 1 (reg26,bit0) first

		ucBuf = Get_F81865_Reg(0x26);
		ucBuf |= BIT0;
		Set_F81865_Reg(0x26, ucBuf);
		//set UR5_FULL_EN(bit1), clear UR6_FULL_EN(bit3)
		ucBuf = Get_F81865_Reg(0x2A);
		ucBuf &= ~BIT3;//clear bit 3,
		ucBuf |= BIT1;//set bit 1,
		Set_F81865_Reg(0x2a, ucBuf);

//GPIO51 ~ GPIO52
		//clear UR6_ALT_EN(bit5), IR_ALT_EN(bit4),set FDC_GP_EN(bit3)

//GPIO50
//set FDC_GP_EN(bit3), clear  RTS6_ALT_EN(RTS6_2_ALT_EN)(bit6)

		ucBuf = Get_F81865_Reg(0x2A);
		ucBuf &= ~(BIT4+BIT5+BIT6);		//clear UR6_ALT_EN(bit5), IR_ALT_EN(bit4),  RTS6_ALT_EN(RTS6_2_ALT_EN)(bit6)
		Set_F81865_Reg(0x2a, ucBuf);
		//set FDC_GP_EN(bit3), should clear UR_GP_PROG_EN (reg26,bit0) first
		ucBuf = Get_F81865_Reg(0x26);
		ucBuf &= ~BIT0;
		Set_F81865_Reg(0x26, ucBuf);//clear UR_GP_PROG_EN = 0 (reg26,bit0)

		ucBuf = Get_F81865_Reg(0x2A);
		ucBuf |= BIT3;			//set FDC_GP_EN(bit3),
		Set_F81865_Reg(0x2a, ucBuf);


		Set_F81865_LD(0x06);					//switch to logic device 6

		//enable the GP5 group
		ucBuf = Get_F81865_Reg(0x30);
		ucBuf |= 0x01;
		Set_F81865_Reg(0x30, ucBuf);

		Set_F81865_Reg(0xA0, 0x00);				//define as input mode
		Set_F81865_Reg(0xA3, 0xFF);				//push pull mode
}
//-----------------------------------------------------------------------------
```

```
void Dio5SetOutput(unsigned char NewData)
{
        Set_F81865_LD(0x06);                                    //switch to logic device 6
        Set_F81865_Reg(0xA1, NewData);
}
//----------------------------------------------------------------------------
unsigned char Dio5GetInput(void)
{
        unsigned char result;

        Set_F81865_LD(0x06);                                    //switch to logic device 6
        result = Get_F81865_Reg(0xA2);
        return (result);
}
//----------------------------------------------------------------------------
void Dio5SetDirection(unsigned char NewData)
{
        //NewData : 1 for input, 0 for output
        Set_F81865_LD(0x06);                                    //switch to logic device 6
        Set_F81865_Reg(0xA0, NewData);
}
//----------------------------------------------------------------------------
unsigned char Dio5GetDirection(void)
{
        unsigned char result;

        Set_F81865_LD(0x06);                                    //switch to logic device 6
        result = Get_F81865_Reg(0xA0);
        return (result);
}
//----------------------------------------------------------------------------
```

Filename：81865.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 "F81865.H"
#include <dos.h>
//----------------------------------------------------------------------------
unsigned int F81865_BASE;
void Unlock_F81865 (void);
void Lock_F81865 (void);
//----------------------------------------------------------------------------
unsigned int Init_F81865(void)
{
        unsigned int result;
        unsigned char ucDid;

        F81865_BASE = 0x4E;
        result = F81865_BASE;

        ucDid = Get_F81865_Reg(0x20);
        if (ucDid == 0x07||ucDid == 0x10)                       //Fintek 81865/66
        {       goto Init_Finish;       }

        F81865_BASE = 0x2E;
        result = F81865_BASE;

        ucDid = Get_F81865_Reg(0x20);
        if (ucDid == 0x07||ucDid == 0x10)                       //Fintek 81865/66
        {       goto Init_Finish;       }

        F81865_BASE = 0x00;
        result = F81865_BASE;

Init_Finish:
        return (result);
}
//----------------------------------------------------------------------------
void Unlock_F81865 (void)
{
        outportb(F81865_INDEX_PORT, F81865_UNLOCK);
        outportb(F81865_INDEX_PORT, F81865_UNLOCK);
}
//----------------------------------------------------------------------------
void Lock_F81865 (void)
{
        outportb(F81865_INDEX_PORT, F81865_LOCK);
}
//----------------------------------------------------------------------------
void Set_F81865_LD( unsigned char LD)
{
        Unlock_F81865();
        outportb(F81865_INDEX_PORT, F81865_REG_LD);
        outportb(F81865_DATA_PORT, LD);
        Lock_F81865();
}
//----------------------------------------------------------------------------
void Set_F81865_Reg( unsigned char REG, unsigned char DATA)
{
        Unlock_F81865();
        outportb(F81865_INDEX_PORT, REG);
        outportb(F81865_DATA_PORT, DATA);
        Lock_F81865();
}
//----------------------------------------------------------------------------
unsigned char Get_F81865_Reg(unsigned char REG)
{
        unsigned char Result;
        Unlock_F81865();
```

```
	outportb(F81865_INDEX_PORT, REG);
	Result = inportb(F81865_DATA_PORT);
	Lock_F81865();
	return Result;
}
//-----------------------------------------------------------------------------
```

Filename：81865.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 __F81865_H
#define __F81865_H                                  1
//-----------------------------------------------------------------------------
#define	F81865_INDEX_PORT               (F81865_BASE)
#define	F81865_DATA_PORT                (F81865_BASE+1)
//-----------------------------------------------------------------------------
#define	F81865_REG_LD                               0x07
//-----------------------------------------------------------------------------
#define F81865_UNLOCK                   0x87
#define	F81865_LOCK                                 0xAA
//-----------------------------------------------------------------------------
unsigned int Init_F81865(void);
void Set_F81865_LD( unsigned char);
void Set_F81865_Reg( unsigned char, unsigned char);
unsigned char Get_F81865_Reg( unsigned char);
//-----------------------------------------------------------------------------
#endif	//__F81865_H
```

Chapter 9 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:
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.

```
//---------------------------------------------------------------------------
//
// 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 "F81866.H"
//---------------------------------------------------------------------------
int main (int argc, char *argv[]);
void EnableWDT(int);
void DisableWDT(void);
//---------------------------------------------------------------------------
int main (int argc, char *argv[])
{
        unsigned char bBuf;
        unsigned char bTime;
        char **endptr;

        char SIO;

        printf("Fintek 81866 watch dog program\n");

        SIO = Init_F81866();
        if (SIO == 0)
        {
                printf("Can not detect Fintek 81866, program abort.\n");
                return(1);
        }//if (SIO == 0)

        if (argc != 2)
        {
                printf(" Parameter incorrect!!\n");
                return (1);
        }

        bTime = strtol (argv[1], endptr, 10);
        printf("System will reset after %d seconds\n", bTime);

        if (bTime)
        {       EnableWDT(bTime);       }
        else
        {       DisableWDT();   }
```

```
	return 0;
}

//-----------------------------------------------------------------------
void EnableWDT(int interval)
{
	unsigned char bBuf;

	bBuf = Get_F81866_Reg(0x2B);
	bBuf &= (~0x20);
	Set_F81866_Reg(0x2B, bBuf);			//Enable WDTO

	Set_F81866_LD(0x07);				//switch to logic device 7
	Set_F81866_Reg(0x30, 0x01);			//enable timer

	bBuf = Get_F81866_Reg(0xF5);
	bBuf &= (~0x0F);
	bBuf |= 0x52;
	Set_F81866_Reg(0xF5, bBuf);			//count mode is second

	Set_F81866_Reg(0xF6, interval);		//set timer

	bBuf = Get_F81866_Reg(0xFA);
	bBuf |= 0x01;
	Set_F81866_Reg(0xFA, bBuf);			//enable WDTO output

	bBuf = Get_F81866_Reg(0xF5);
	bBuf |= 0x20;
	Set_F81866_Reg(0xF5, bBuf);			//start counting
}
//-----------------------------------------------------------------------
void DisableWDT(void)
{
	unsigned char bBuf;

	Set_F81866_LD(0x07);				//switch to logic device 7

	bBuf = Get_F81866_Reg(0xFA);
	bBuf &= ~0x01;
	Set_F81866_Reg(0xFA, bBuf);			//disable WDTO output

	bBuf = Get_F81866_Reg(0xF5);
	bBuf &= ~0x20;
	bBuf |= 0x40;
	Set_F81866_Reg(0xF5, bBuf);			//disable WDT
}
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
//
// 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 "F81866.H"
#include <dos.h>
//-----------------------------------------------------------------------
unsigned int F81866_BASE;
void Unlock_F81866 (void);
void Lock_F81866 (void);
//-----------------------------------------------------------------------
unsigned int Init_F81866(void)
{
	unsigned int result;
	unsigned char ucDid;

	F81866_BASE = 0x4E;
	result = F81866_BASE;
```

```
        ucDid = Get_F81866_Reg(0x20);
        if (ucDid == 0x07)                                      //Fintek 81866
        {       goto Init_Finish;   }

        F81866_BASE = 0x2E;
        result = F81866_BASE;
        ucDid = Get_F81866_Reg(0x20);
        if (ucDid == 0x07)                                      //Fintek 81866
        {       goto Init_Finish;   }

        F81866_BASE = 0x00;
        result = F81866_BASE;

Init_Finish:
        return (result);
}
//-----------------------------------------------------------------------------
void Unlock_F81866 (void)
{
        outportb(F81866_INDEX_PORT, F81866_UNLOCK);
        outportb(F81866_INDEX_PORT, F81866_UNLOCK);
}
//-----------------------------------------------------------------------------
void Lock_F81866 (void)
{
        outportb(F81866_INDEX_PORT, F81866_LOCK);
}
//-----------------------------------------------------------------------------
void Set_F81866_LD( unsigned char LD)
{
        Unlock_F81866();
        outportb(F81866_INDEX_PORT, F81866_REG_LD);
        outportb(F81866_DATA_PORT, LD);
        Lock_F81866();
}
//-----------------------------------------------------------------------------
void Set_F81866_Reg( unsigned char REG, unsigned char DATA)
{
        Unlock_F81866();
        outportb(F81866_INDEX_PORT, REG);
        outportb(F81866_DATA_PORT, DATA);
        Lock_F81866();
}
//-----------------------------------------------------------------------------
unsigned char Get_F81866_Reg(unsigned char REG)
{
        unsigned char Result;
        Unlock_F81866();
        outportb(F81866_INDEX_PORT, REG);
        Result = inportb(F81866_DATA_PORT);
        Lock_F81866();
        return Result;
}
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//
// 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 __F81866_H
#define __F81866_H                                     1
//-----------------------------------------------------------------------------
#define  F81866_INDEX_PORT              (F81866_BASE)
#define  F81866_DATA_PORT               (F81866_BASE+1)
//-----------------------------------------------------------------------------
#define  F81866_REG_LD                  0x07
//-----------------------------------------------------------------------------
```

```
#define F81866_UNLOCK                   0x87
#define   F81866_LOCK                         0xAA
//----------------------------------------------------------------------------
unsigned int Init_F81866(void);
void Set_F81866_LD( unsigned char);
void Set_F81866_Reg( unsigned char, unsigned char);
unsigned char Get_F81866_Reg( unsigned char);
//----------------------------------------------------------------------------
#endif    //__F81866_H
```

A. I/O Port Address Map

Each peripheral device in the system is assigned a set of I/O port addresses which 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
2F8h - 2FFh	Serial Port #2(COM2)
2B0h- 2DFh	Graphics adapter Controller
360h - 36Fh	Network Ports
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
IRQ4	Serial Port #1
IRQ3	Serial Port #2
IRQ5	Serial Port #3
IRQ11	Serial Port #4
IRQ8	Real Time Clock
IRQ14	Primary IDE
IRQ15	Secondary IDE

C. Register of the LAN Bypass Controller

To fulfill the varied requests on LAN Bypass controller, IBASE provide the smart LAN Bypass controller. User can define the Bypass function behavior when the system is power-on, power-off and WDT signal is asserted.
The controller is behind the SMBus controller. The I^2C address is listed as below:

	I^2C Address (8bit)	Remark
1st Controller	0x68	
2nd Controller	0x6A	Optional

CR 0x22 : System-On Bypass Register
Attribute : Read / Write
Reset default : 0x0000

Bit	LAN Port#	Read / Write	Description
0	Eth1, 2	Read / Write	Enable / Disable LAN Bypass function when the system is power On. 1 = Enable LAN Bypass function 0 = Disable LAN Bypass function

CR 0x24 : System-Off Bypass Register
Attribute : Read / Write
Reset default : 0x0000

Bit	LAN Port#	Read / Write	Description
0	Eth1, 2	Read / Write	Enable / Disable LAN Bypass function when the system is power Off. 1 = Enable LAN Bypass function 0 = Disable LAN Bypass function

CR 0x26 : Watchdog (WDT) WDT_IN# Signal Control Register
Attribute : Read / Write
Reset default : 0x00

Bit	Read / Write	Description
0	Read / Write	**WDT_OUT# Generator** The capacity use the WDT to reset the system 1 = Generate 100ms pulse to reset signal when WDT signal is asserted. 0 = Ignore the WDT signal.

1	Read / Write	**WDT LAN Bypass Enable** The capacity use the WDT to set LAN Bypass function 1 = Enable LAN Bypass function when the WDT signal is asserted. **CR 0x28** and **CR 0x2A** will be available if this bit is set to "1". 0 = Disable WDT LAN Bypass function.
2 ~ 7		Reserved

CR 0x28 : Watchdog (WDT) Bypass Control Register
Attribute : Read / Write
Reset default : 0x0000

Bit	LAN Port#	Read / Write	Description
0	Eth1, 2	Read / Write	Enable / Disable WDT Bypass function for each LAN port. 1 = Follow the setting in "WDT Bypass Register **CR 0x2A**" when the WDT signal is asserted. 0 = Ignore to control the bypass when the WDT is asserted.

CR 0x2A : Watchdog (WDT) Bypass Register
Attribute : Read / Write
Reset default : 0x0000

Bit	LAN Port#	Read / Write	Description
0	Eth1, 2	Read / Write	The function works when the bit in **CR 0x28** is “1”. It controls LAN Bypass function should be Enabled / Disabled when the WDT signal is asserted. If the bit is set to "1" in "WDT Bypass Control Register **CR 0x28**", it will follow below setting: 1 = Enable LAN Bypass function 0 = Disable LAN Bypass function