MB967-FT

Intel® Ivy Bridge / C216 PCH

USER'S MANUAL

Version 1.3

Acknowledgments

Table of Contents

This page is intentionally left blank.

Introduction

Product Description

The MB967-FT networking motherboard is based on the latest Intel® C216 chipset. The platform supports 3rd generation Intel® Core processor family with LGA1155 packing and features an integrated dual-channel ECC or Non-ECC DDR3 memory controller as well as a graphics core.

The latest Intel® processors provide advanced performance in both computing and graphics quality. This meets the requirement of customers in the network security (Firewall, VPN, UTM…) and server market segment.

The C216 platform is made with 22-nanometer technology that supports Intel's first processor architecture to unite the CPU and the graphics core on the transistor level. MB967-FT utilizes the dramatic increase in performance provided this Intel's latest cutting-edge technology. Measuring 355mm x 185mm, MB967-FT features Intel® Active Management Technology 8.0, offers eight GbE LAN ports, two fast 6Gbps SATA ports and two USB3.0 ports.

MB967-FT Features

- Supports 8 Intel® 10/100/1000 LAN ports
- Supports Intel® 3rd Generation Core i7/i5/i3 / Xeon® processors
- DDR3 x4, up to 32 GB
- Mini PCI-e (m-SATA compatible) socket x1, CF socket x1
- iAMT 8.0 support

Specifications

Product Name	MB967-FT
Processor	Support for Intel® Sandy-Bridge and Ivy-Bridge processors which fall within mainstream TDP envelope, up to 95W TDP. Support for the following Intel® technologies: • Intel SSE/MMX, AES and AVX new instructions • VT-x and VT-d Virtualization Technologies • ACPI 3.0 • Intel® 64 (64-bit extensions) • PCI-e 3.0 (1x16 or 2x8) • DMI 2.0
Chipset	Intel® Panther Point C216 PCH • SATA 3.0 • iAMT 8.0
BIOS	AMI BIOS
Memory	• Four DDR3 UDIMM total for 32GB max memory • Dual channel DDR3 up to 1600 MHz • Unbuffered • ECC or non-ECC
Video	• Intel® CPU integrated graphics • IBASE VGA4 pin header on board
Network Controller	• Eth1: Intel® 82579LM GbE PHY w/ iAMT 8.0 supporting. No Bypass • Eth2~4: Intel® 82583V GbE. No Bypass. • Eth5~6: Intel® 82583V GbE. No Bypass. • Eth7~8: Intel® 82583V GbE. No Bypass.
SATA Connector	• SATA III (6.0Gb/s): 2 ports
USB Port	• Two USB 3.0 ports at front panel • Four USB 2.0 pin header on board
Compact Flash	ACARD ARC772 SATA to PATA for CompactFlash type II
LPC I/O	Fintek F81866A: • RS-232 [2x5] Pin Header on board x1 • RJ-45 Console x1 • KB/Mouse [1x6] Pin header • Hardware monitors • Fan connector x4
Fan Connector	4-pin fan connectors: • One for CPU fan (Smart Fan) • Three for System fans (Smart Fan)

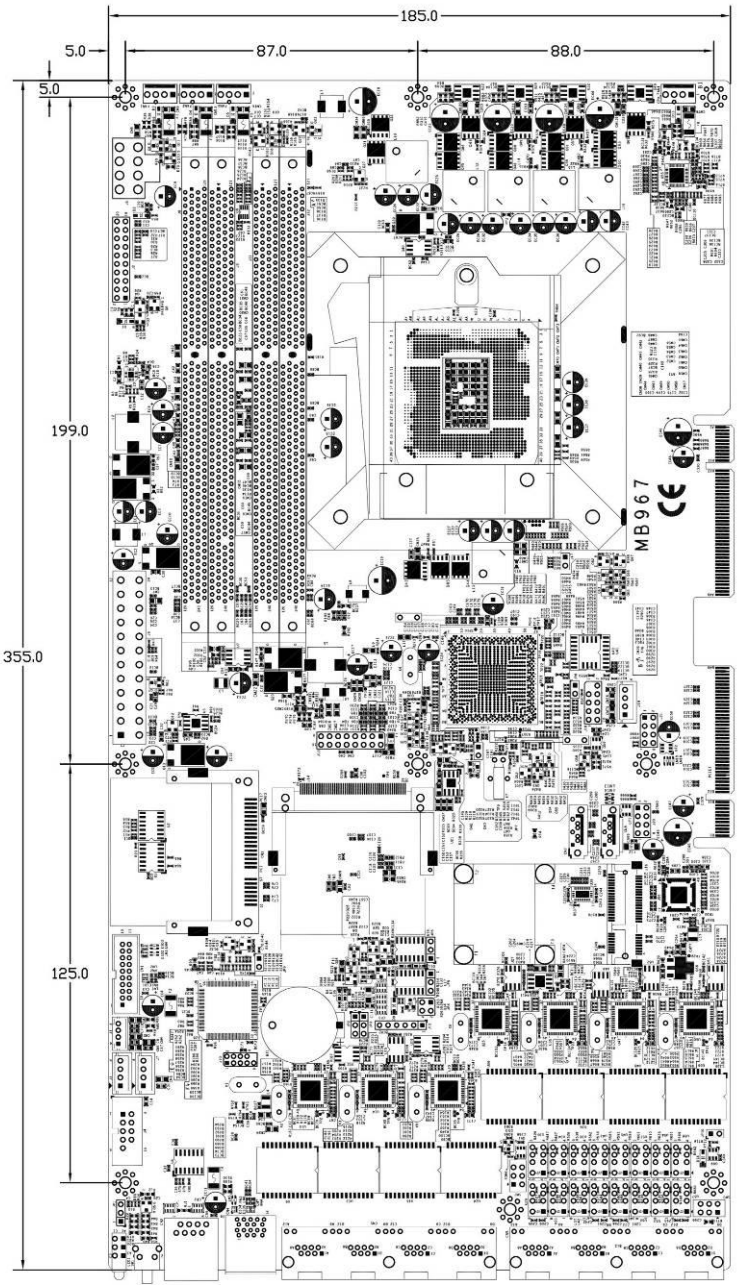
Smart Fan Control	The active temperature may be adjusted based on system thermal test result
RTC	Intel C216 built-in RTC with on-board lithium battery & holder
Expansion Slot	• Golden finger PCI-e x8, Front • Golden finger PCI-e x8, Rear
Expansion Interface	• Mini PCI-e socket x1 (mSATA compatible) • CF card socket x1
Front Panel LED	 #3 LED: Status (GPIO control, Yellow / Red) #2 LED: Reserved #1 LED: Power (Green = Power On, Off= No Power)
LCM	2x16 characters LCM (COM2)
Front Button & Connector	• Two RJ-45 1x4 connectors for Eth1~4 & 5~6 • USB 3.0 x2 • RJ-45 (for console, COM1) • Three LEDs for Power & Status • Factory Mode Restore Reset Switch
Watchdog Timer	Yes (256 segments, 0, 1, 2…255 sec/min) Present in BMC and configurable by O/S agent to automatically reset system if O/S hangs
TPM	nuvoTon WPCT210 TPM1.2
Operating Temperature	0°C ~ 60°C
Storage Temperature	-20°C ~ 80°C
Operational Humidity	10% ~ 90% Relative Humidity (non-condensing)
RoHS Compliant	Yes
Board Size	355 x 185mm, 1.6mm thickness

Checklist

Your MB967-FT package should include the items listed below.

- MB967-FT motherboard
- Driver DVD

Board Dimensions



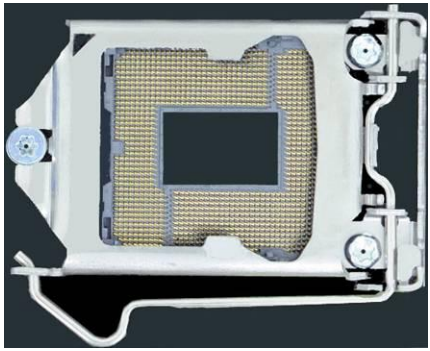
Installations

This section provides information on how to use the jumpers and connectors on the MB967-FT in order to set up a workable system. The topics covered are:

Installing the CPU

The MB967-FT board supports an LGA1155 Socket (shown below) for Intel Clarkdale processors.

To install the CPU, unlock first the socket by pressing the lever sideways, then lift it up to a 90-degree. Then, position the CPU above the socket such that the CPU corner aligns with the gold triangle matching the socket corner with a small triangle. Carefully insert the CPU into the socket and push down the lever to secure the CPU. Then, install the heat sink and fan.



***NOTE**: Ensure that the CPU heat sink and the CPU top surface are in total contact to avoid CPU overheating problem that would cause your system to hang or be unstable.*

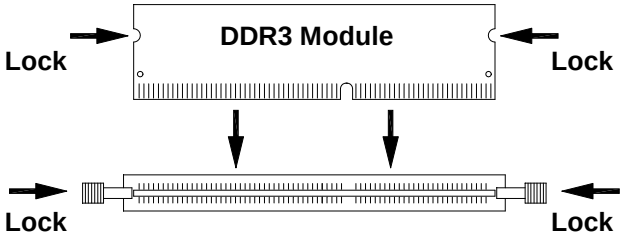
Installing the Memory

The MB967-FT board supports four DDR3 memory socket for a maximum total memory of 32GB in DDR3 DIMM memory type.

Installing and Removing Memory Modules

To install the DDR3 modules, locate the memory slot on the board and perform the following steps:

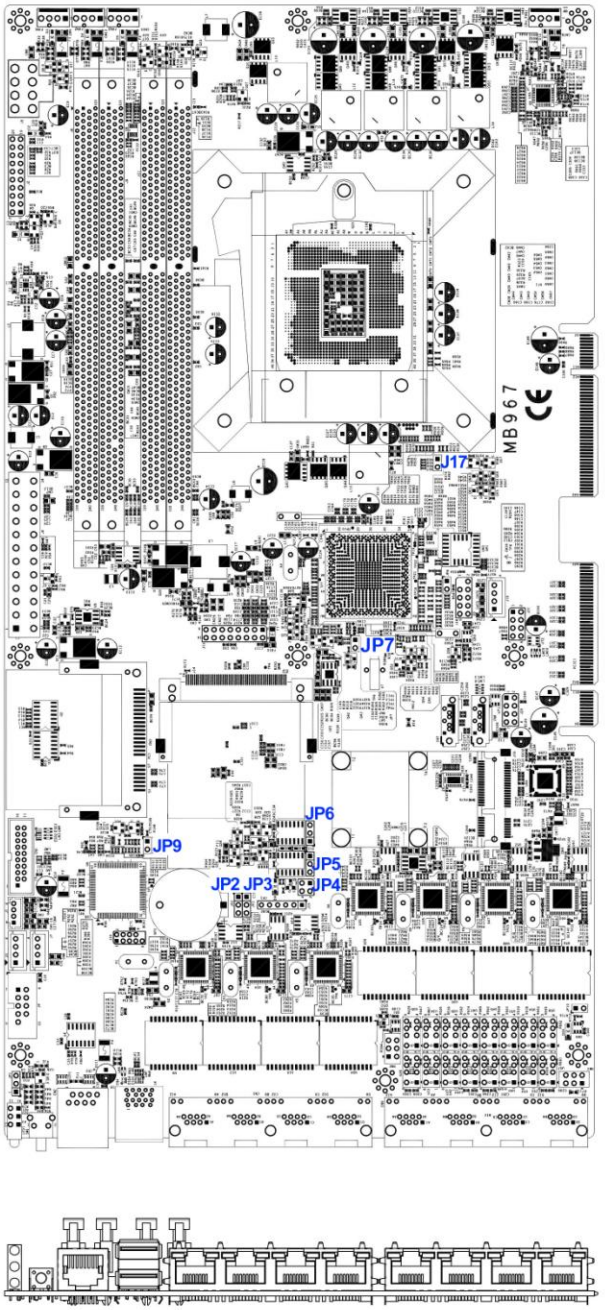
1. Hold the DDR3 module so that the key of the DDR3 module aligned with that on the memory slot.
2. Gently push the DDR3 module in an upright position until the clips of the slot close to hold the DDR3 module in place when the DDR3 module touches the bottom of the slot.
3. To remove the DDR3 module, press the clips with both hands.



Setting the Jumpers

Jumpers are used on MB967-FT 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 MB967-FT and their respective functions.

Jumper Locations on MB967-FT



Jumper Settings on MB967-FT

JP2: Clear CMOS Contents

Use JP2 to clear the CMOS contents. *Note that the ATX-power connector should be disconnected from the board before clearing CMOS.*

JP3	Setting	Function
	Pin 1-2 Short/Closed	Normal
	Pin 2-3 Short/Closed	Clear CMOS

JP3: Clear ME RTC Contents

JP3	Setting	Function
	Pin 1-2 Short/Closed	Normal
	Pin 2-3 Short/Closed	Clear ME RTC

JP4: Watchdog Reboot (WDT) Select

JP4	Setting	Function
	Pin 3-4 Closed	System will reboot upon the time out of watchdog timer.
	Pin 3-4 Open	Watchdog function Disabled

JP7: Flash Descriptor Security Override (ME BIOS Update Jumper)

JP7	Setting	Function
	Open	Disable (Default)
	Short/Closed	Enable to update BIOS

JP9: ATX & AT Mode Select

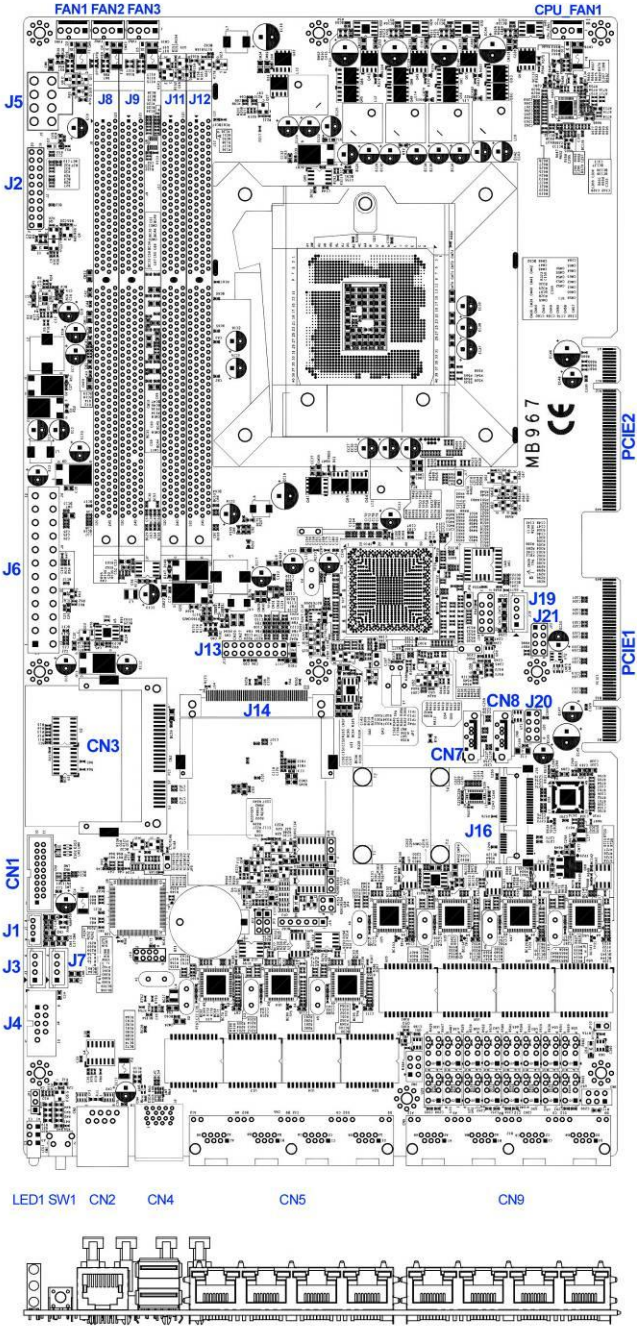
JP9	Setting	Function
	Short/Closed	AT Mode (Default)
	Open	ATX Mode

J17: PCIE1 & PCIE2 Golden Finger PCI-e Configuration

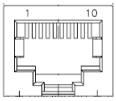
J17	Setting	Function	Remarks
	Open	Combine to 1x16	For CPU with 1x16 support
	Short / Closed	Separate to2x8	Default for CPU with 2x8 support

Connectors on MB967-FT

Connector Locations on MB967-FT



CN2: COM1 RJ45 Connector

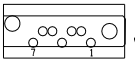


Pin #	Signal Name (RS-232)
1	RTS, Request to send
2	DTR, Data terminal ready
3	TXD, Transmit data
4	Ground
5	Ground
6	RXD, Receive data
7	DSR, Data set ready
8	CTS, Clear to send

CN4: USB Connector

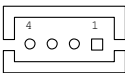
CN5, CN9: LAN Connectors

CN7, CN8: SATA HDD Connector



Pin #	Signal Name
1	Ground
2	TX+
3	TX-
4	Ground
5	RX-
6	RX+
7	Ground

J1, J3, J7, J19: Power Output Connector

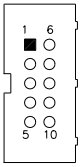


Pin #	Signal Name
1	+5V
2	GND
3	GND
4	+12V

J2: Front Panel Function Connector

Signal Name	Pin #	Pin #	Signal Name
PWR LED +	1	2	SPK +
NC	3	4	NC
PWR LED- (GND)	5	6	SPK – (GND)
NC	7	8	+5V
GND	9	10	NC
GND	11	12	NC
PWR_SW	13	14	PWR_SW
SLED+	15	16	SLED-
GND	17	18	RST
HDD LED +	19	20	HDD LED -

J4: COM2 Serial Port

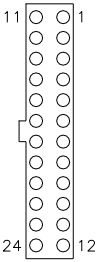


Pin #	Signal Name (RS-232)
1	DCD, Data carrier detect
2	RXD, Receive data
3	TXD, Transmit data
4	DTR, Data terminal ready
5	Ground
6	DSR, Data set ready
7	RTS, Request to send
8	CTS, Clear to send
9	RI, Ring indicator
10	No Connect.

J5: ATX 12V Power Connector

Signal Name	Pin #	Pin #	Signal Name
+12V	5	1	Ground
+12V	6	2	Ground
+12V	7	3	Ground
+12V	8	4	Ground

J6: 24-pin ATX Power Connector



Signal Name	Pin #	Pin #	Signal Name
3.3V	13	1	3.3V
-12V	14	2	3.3V
Ground	15	3	Ground
PS-ON	16	4	+5V
Ground	17	5	Ground
Ground	18	6	+5V
Ground	19	7	Ground
-5V	20	8	Power good
+5V	21	9	5VSB
+5V	22	10	+12V
+5V	23	11	+12V
Ground	24	12	+3.3V

J8, J9: Channel B DDR3 Socket

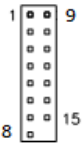
J8, J9 are the second-channel DDR3 sockets.

J10: LPC Debug Port (Reserved for factory use only)

J11, J12: Channel A DDR3 Socket

J11, J12 are the first-channel DDR3 sockets.

J13: VGA Box Header



Signal Name	Pin #	Pin #	Signal Name
R	1	9	+5V
G	2	10	GND
B	3	11	NC
NC	4	12	SPD1
GND	5	13	Hsync
GND	6	14	Vsync
GND	7	15	SPCLK
GND	8		

J14: Slim Type II Compact Flash Connector

J16: Mini PCI-e Card & m-SATA Connector

J18: SPI Debug Port (Reserved for factory use only)

J20, J21: USB 2.0 Pin Header

Signal Name	Pin #	Pin #	Signal Name
VCC	1	2	Ground
USB1-	3	4	USB2+
USB1+	5	6	USB2-
Ground	7	8	VCC

FAN1, 2, 3: System Fan Power Connector

FAN1/2/3 are 4-pin headers for System fan power.

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

CPU_FAN1: CPU Fan Power Connector

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

SW1: Software reset button

Signal Name	Pin #	Pin #	Signal Name
GND	1	2	PCH GPIO7

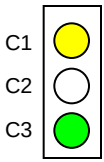
IO Base:
Read IO 0x500 and set bit 7 to "1" (Enabled GPIO function)
Read IO 0x504 and set bit 7 to "1" (GPIO act as GPI)
Read IO 0x50C and check the bit 7 (Control Pin)

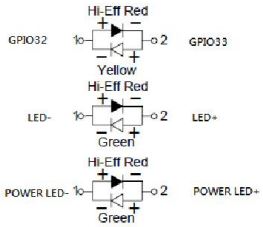
Note: SW3 is controlled by GPIO only.

PCIE1: PCI-e x8 Golden Finger 1

PCIE2: PCI-e x8 Golden Finger 2

LED1: Power & Status LED





Signal Name	Pin #	Pin #	Signal Name
SIO GPIO32	C1	A1	SIO GPIO33
Reserved	C2	A2	Reserved
POWER LED-	C3	A3	POWER LED+

STATUS LED	GPIO33	GPIO32
YELLOW	H	L
RED	L	H

Index port: 4E
Data port: 4F
Device: 06
30h ➔ bit1 = 1
C0h ➔ set bit2, bit3 = 1 for GPO
C3h ➔ set bit2, bit3 = 1 for Push Pull
C1h ➔ bit2, bit3 (Control pin) for GPO32, 33

This page is intentionally left blank.

Digital I/O Sample Configuration

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
```

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
```

BIOS Setup

This chapter describes the different settings available in the AMI BIOS that comes with the board. The topics covered in this chapter are as follows:

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:

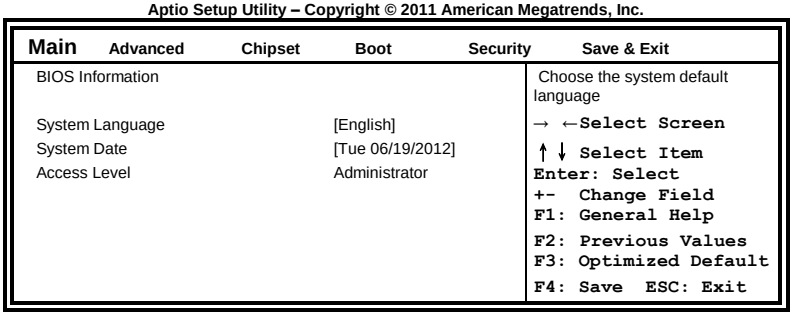
```
Press <DEL> / <F2> to Enter Setup
```

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

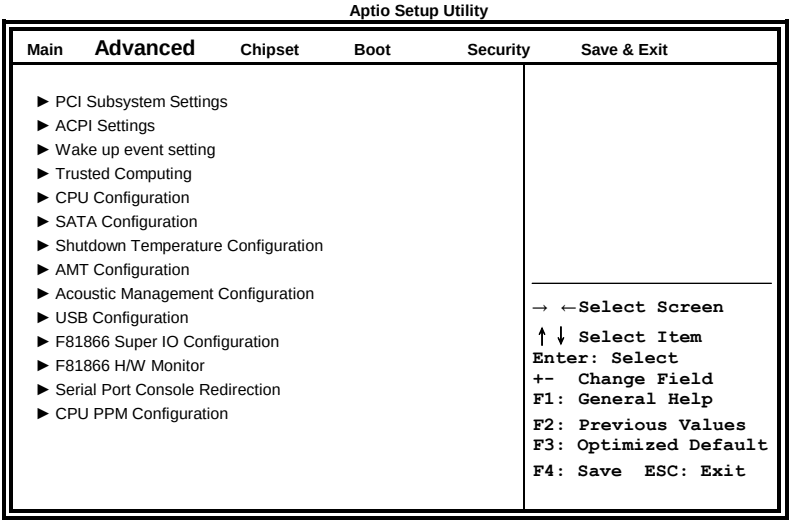
Set the Date. Use Tab to switch between Data elements.

System Time

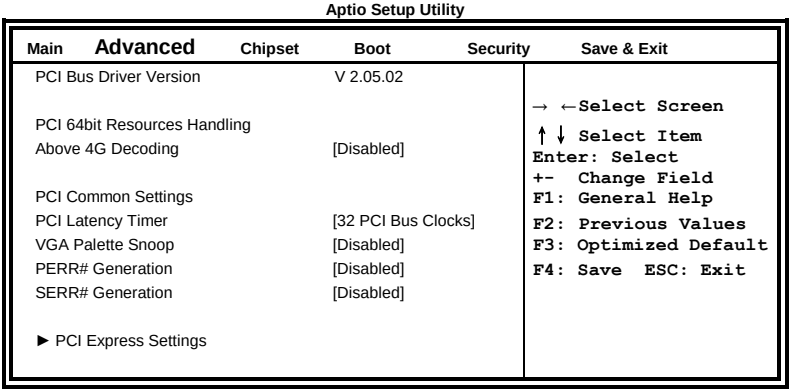
Set the Time. Use Tab to switch between Data elements.

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.



PCI Subsystem Settings



Above 4G Decoding

Enables or Disables 64bit capable devices to be decoded in above 4G address space (only if system supports 64 bit PCI decoding).

PCI Latency Timer
Value to be programmed into PCI Latency Timer Register.

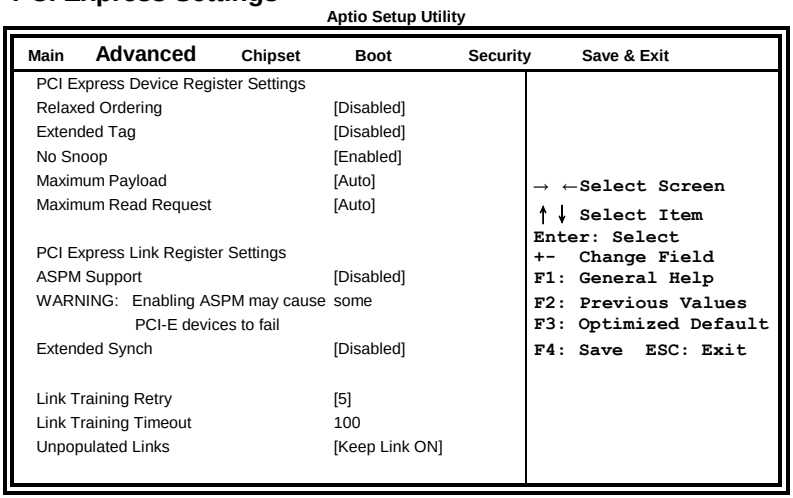
VGA Palette Snoop
Enables or disables VGA Palette Registers Snooping.

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.

PCI Express Settings



Relaxed Ordering
Enables or disables PCI Express Device Relaxed Ordering.

Extended Tag
If ENABLED allows device to use 8-bit Tag field as a requester.

No Snoop
Enables or disables PCI Express Device No Snoop option.

Maximum Payload
Set Maximum Payload of PCI Express Device or allow System BIOS to select the value.

Maximum Read Request
Set Maximum Read Request Size of PCI Express Device or allow System BIOS to select the value.

ASPM Support
Set the ASPM Level: Force L0s – Force all links to L0s State:
AUTO – BIOS auto configure ： DISABLE – Disables ASPM.

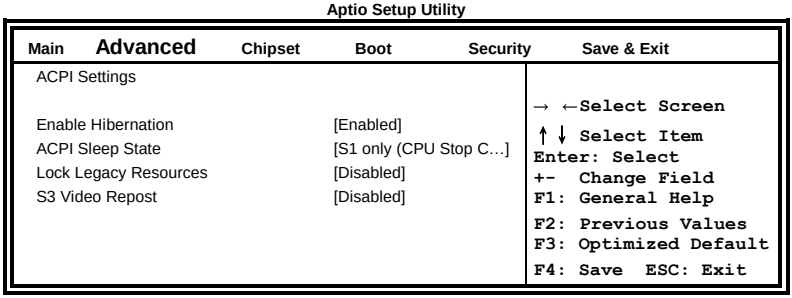
Extended Synch
If ENABLED allows generation of Extended Synchronization patterns.

Link Training Retry
Defines number of Retry Attempts software will take to retrain the link if previous training attempt was unsuccessful.

Link Training Timeout
Defines number of Microseconds software will wait before polling 'Link Training' bit in Link Status register. Value range from 10 to 1000 uS.

Unpopulated Links
In order to save power, software will disable unpopulated PCI Express links, if this option set to 'Disable Link'.

ACPI Settings



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.

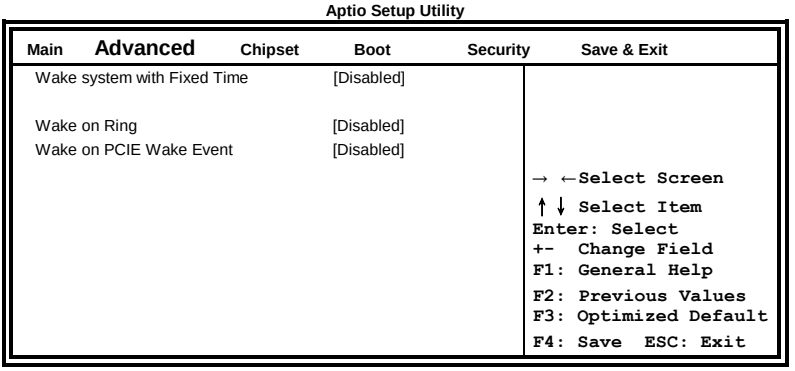
Lock Legacy Resources

Enabled or Disabled Lock of Legacy Resources.

S3 Video Repost

Enable or disable S3 Video Repost.

Wake up event settings



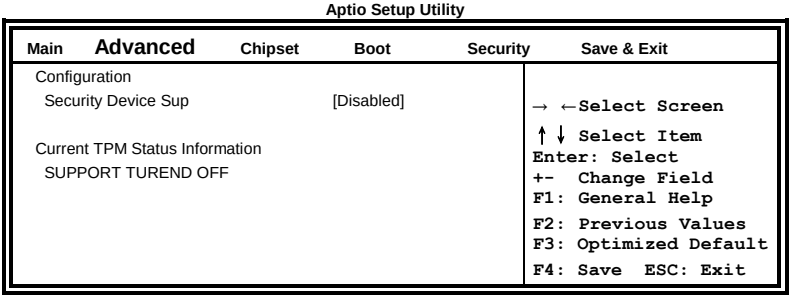
Wake system with Fixed Time

Enables or Disables System wake on alarm event. When enabled, System will wake on the hr::min:: sec specified.

Wake on PCIE PME Wake Event

The options are Disabled and Enabled.

Trusted Computing

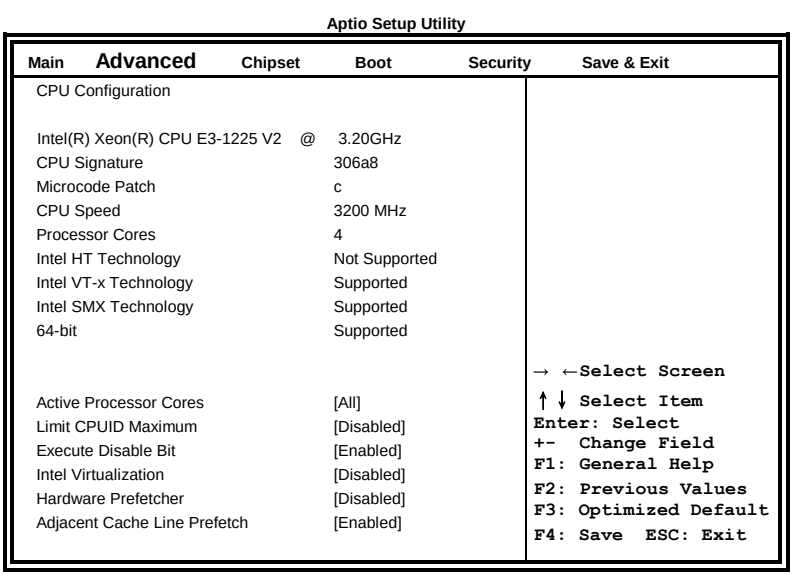


Security Device Support

Enables or disables BIOS support for security device. O.S. will not show Security Device. TCG EFI protocol and INT1A interface will not be available.

CPU Configuration

This section shows the CPU configuration parameters.



Active Processor Cores

Number of cores to enable in each processor package.

Limit CPUID Maximum

Disabled for Windows XP.

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, Re33dHat Enterprise 3 Update 3.)

Intel Virtualization Technology

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

Hardware Prefetcher

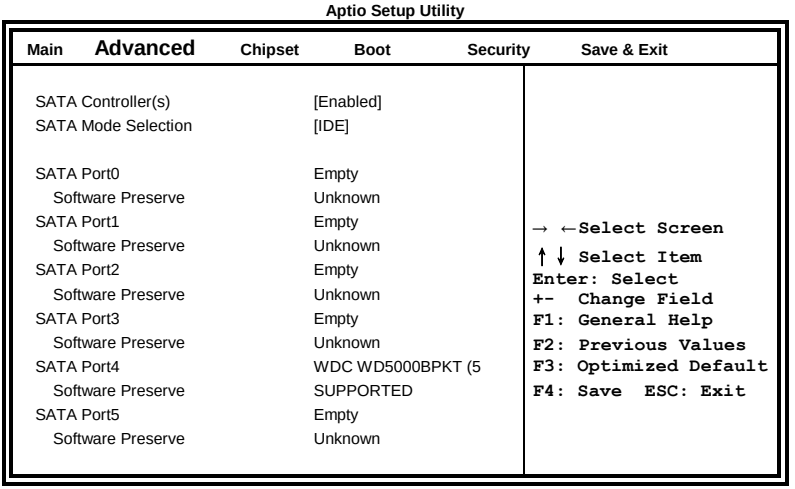
To turn on/off the Mid level Cache (L2) streamer Prefetcher.

Adjacent Cache Line Prefetch

To turn on/off prefetching of adjacent cache lines.

SATA Configuration

SATA Devices Configuration.



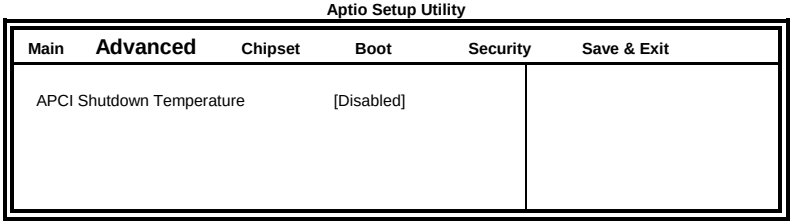
SATA Controller(s)

Enable / Disable Serial ATA Controller.

SATA Mode Selection

(1) IDE Mode.
(2) AHCI Mode.
(3) RAID Mode.

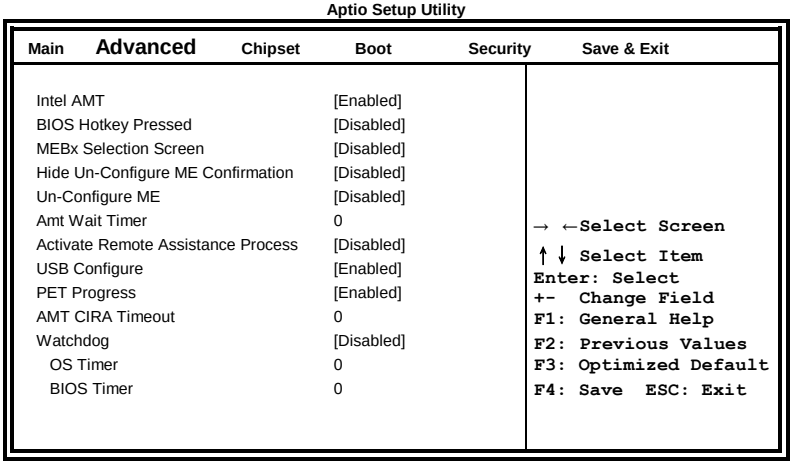
Shutdown Temperature Configuration



ACPI Shutdown Temperature

Set function Disabled or 70/75/80/85/90/95 ℃

AMT Configuration



AMT Configuration

Options are Enabled and Disabled.
Note: iAMT H/W is always enabled. This option just controls the BIOS extension execution. If enabled, this requires additional firmware in the SPI device.

Unconfigure ME

Perform AMT/ME unconfigure without password operation.

Amt Wait Timer

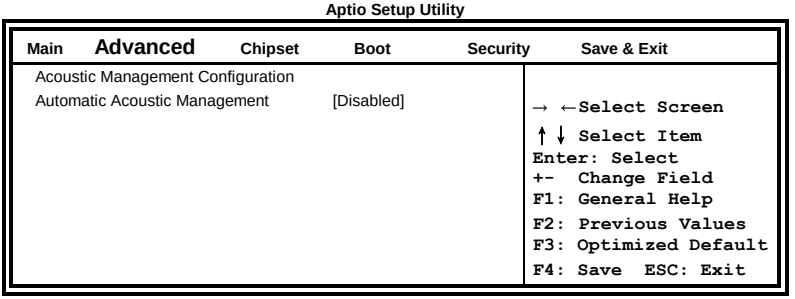
Set timer to wait before sending ASF_GET_BOOT_OPTIONS.

Activate Remote Assistance Process
Trigger CIRA boot.

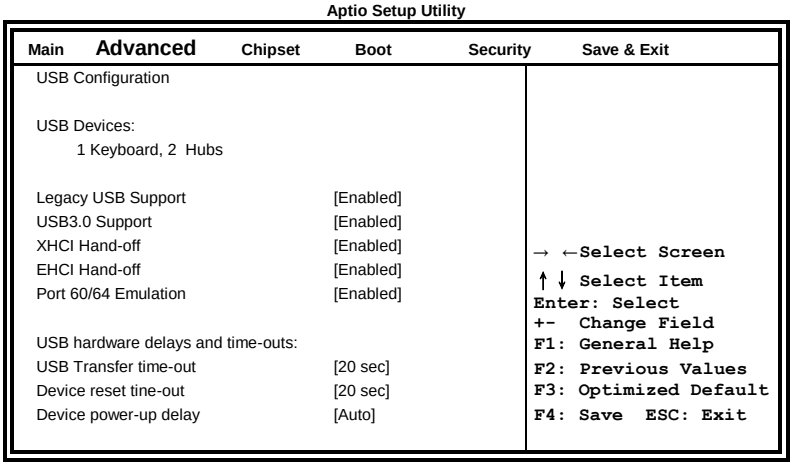
PET Progress
User can Enable/Disable PET Events progress to receive PET events or not.

Watchdog Timer
Enable/Disable Watchdog Timer.

Acoustic Management Configuration



USB Configuration



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.

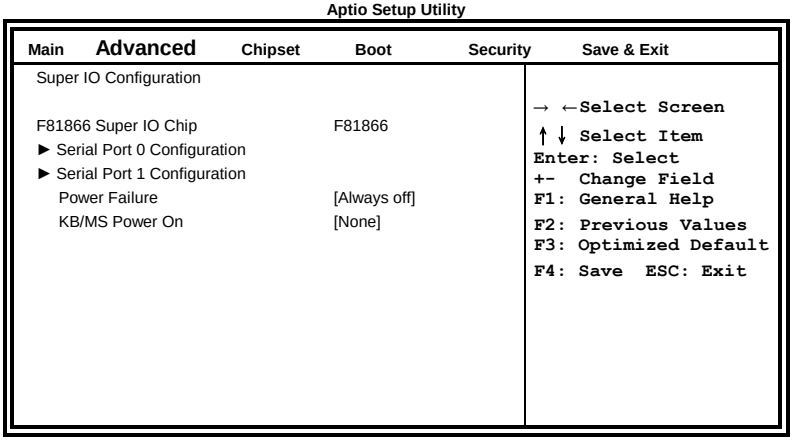
USB Transfer time-out

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

Device reset tine-out
USB mass Storage device start Unit command time-out.

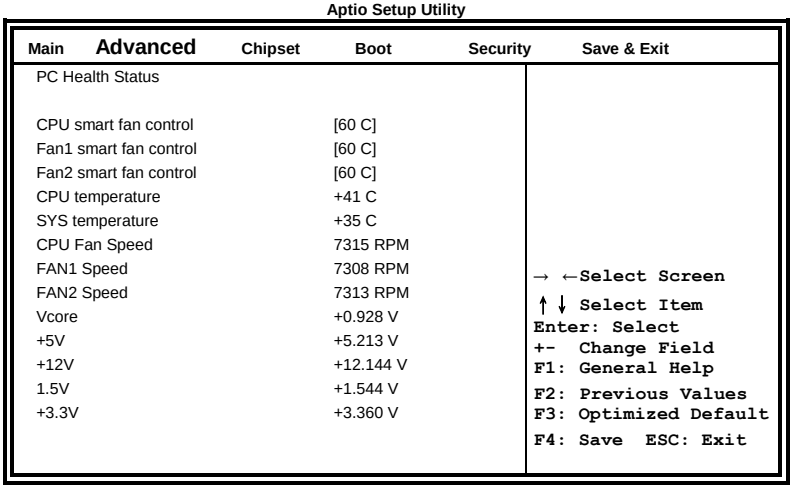
Device power-up delay
Maximum time the device will take before it properly reports itself to the Host Controller. 'Auto' uses default value: for a Root port it is 100ms, for a Hub port the delay is taken from Hub descriptor.

F81866 Super IO Configuration



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.

F81866 H/W Monitor



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/Fan1/Fan2 Smart Fan Control

This field enables or disables the smart fan feature. At a certain temperature, the fan starts turning. Once the temperature drops to a certain level, it stops turning again.

Serial Port Console Redirection

Aptio Setup Utility

Main	Advanced	Chipset	Boot	Security	Save & Exit

COM0		
Console Redirection		[Enabled]
► Console Redirection Settings		
		[Enabled]
COM1 (Pci Bus0, Dev0, Func0)		(Disabled)
Console Redirection	Port Is Disabled	

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

Console Redirection Settings

Aptio Setup Utility

Main	Advanced	Chipset	Boot	Security	Save & Exit

COM0	
Console Redirection Settings	
Terminal Type	[VT100+]
Bits per second	[115200]
Data Bits	[8]
Parity	[None]
Stop Bits	[1]
Flow Control	[None]
VT-UTF8 Combo Key Sup	[Enabled]
Recorder Mode	[Disabled]
Resolution 100x31	[Disabled]
Legacy OS Redirection	[80x24]
Putty KeyPad	[VT100]

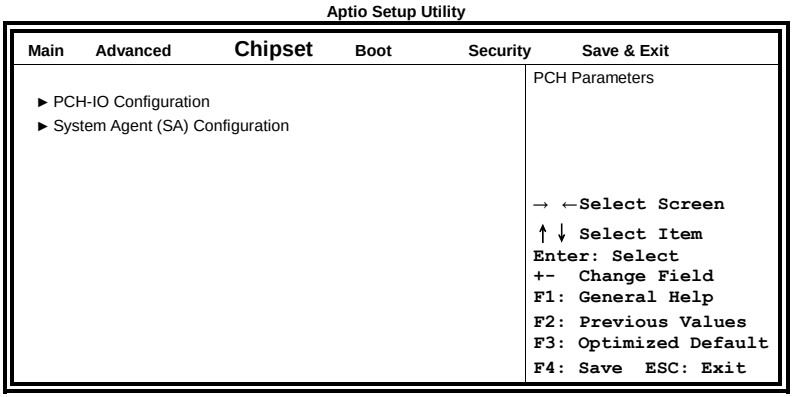
Emulation: ANSI: Extended ASCII char Set. VT100: ASCII char Set. VT100+: Extends VT100 to support color, Function keys, etc. VT-UTF8: Uses UTF8 Encoding to map Unicode Chars onto 1 or more

---

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

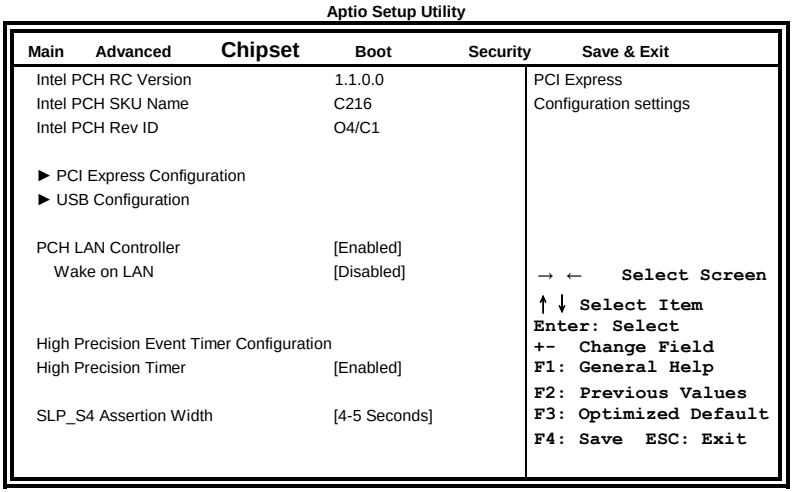
Chipset Settings

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



PCH-IO Configuration

This section allows you to configure the North Bridge Chipset.



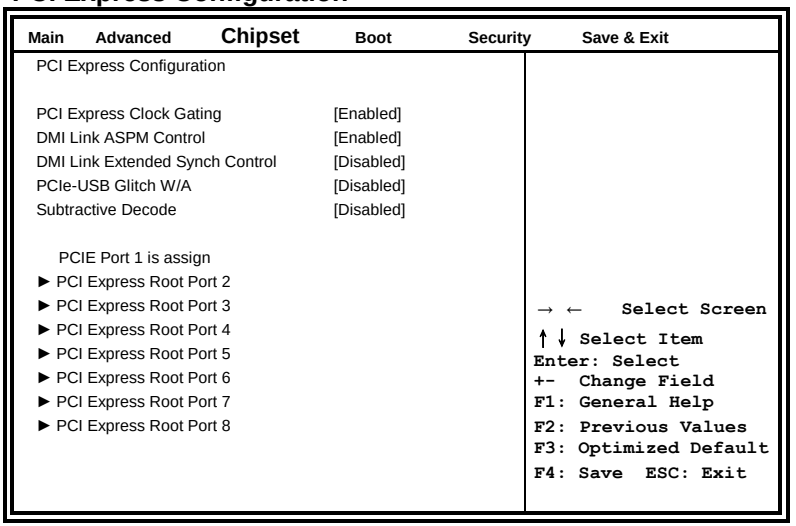
PCH LAN Controller

Enable or disable onboard NIC.

Wake on LAN
Enable or disable integrated LAN to wake the system. (The Wake On LAN cannot be disabled if ME is on at Sx state.)

SLP_S4 Assertion Width
Select a minimum assertion width of the SLP_S4# signal.

PCI Express Configuration



PCI Express Clock Gating
Enable or disable PCI Express Clock Gating for each root port.

DMI Link ASPM Control
The control of Active State Power Management on both NB side and SB side of the DMI link.

PCIe-USB Glitch W/A
PCIe-USB Glitch W/A for bad USB device(s) connected behind PCIE/PEG port.

USB Configuration

Main	Advanced	Chipset	Boot	Security	Save & Exit
USB Configuration					
XHCI Pre-Boot Driver			[Disabled]		
xHCI Mode			[Auto]		
HS Port #1 Switchable			[Enabled]		
HS Port #2 Switchable			[Enabled]		
HS Port #3 Switchable			[Enabled]		
HS Port #4 Switchable			[Enabled]		→ ← Select Screen
xHCI Streams			[Enabled]		↑ ↓ Select Item
EHCI1			[Enabled]		Enter: Select +- Change Field
EHCI2			[Enabled]		F1: General Help F2: Previous Values
USB Ports Per-Port Disable Control			[Disabled]		F3: Optimized Default F4: Save ESC: Exit

HS Port #1/2/3/4 Switchable

Allows for HS port switching between xHCI and EHCI. If disabled, port is routed to EHCI. If HS port is routed to xHCI, the corresponding SS port is enabled.

xHCI Streams

Enable or disable xHCI Maximum Primary Stream Array Size.

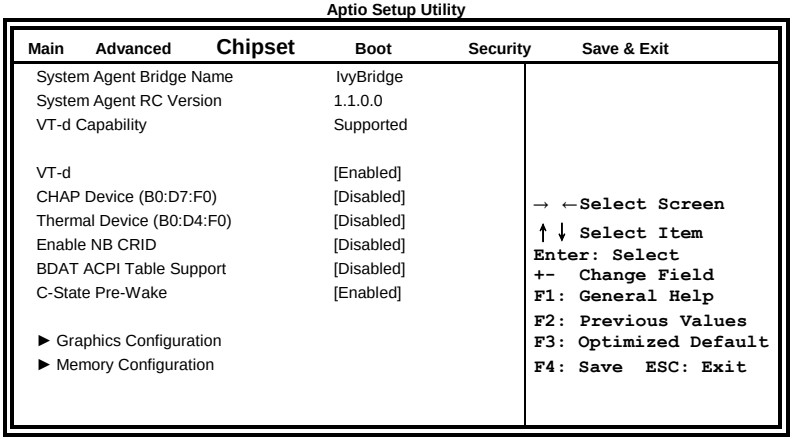
EHCI1/2

Control the USAB EHCI (USB 2.0) functions. One EHCI controller must always be enabled.

USB Ports Per-Port Disable Control

Control each of the USB ports (0~13) disabling.

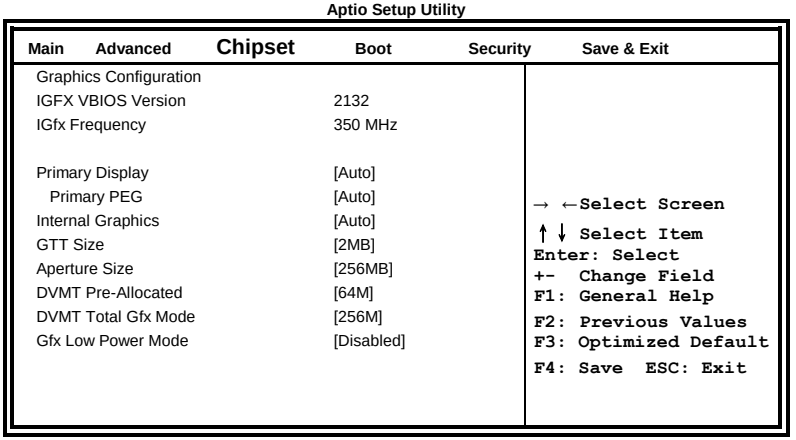
System Agent (SA) Configuration



VT-d

Check to enable VT-d function on MCH.

Graphics Configuration



Primary Display

Select which of IGFX/PEG/PCI graphics device should be primary display or select SG for switchable Gfx.

Internal Graphics

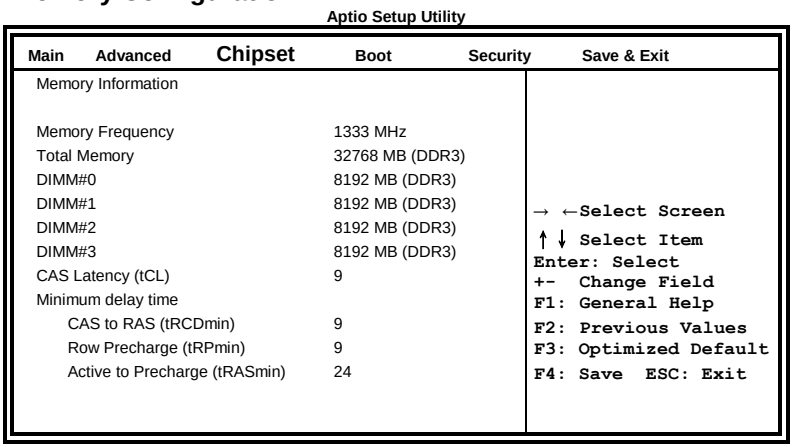
Keep IGD enabled based on the setup options.

DVMT Pre-Allocated
Select DVMT 5.0 Pre-Allocated (Fixed) graphics memory size used by the internal graphics device.

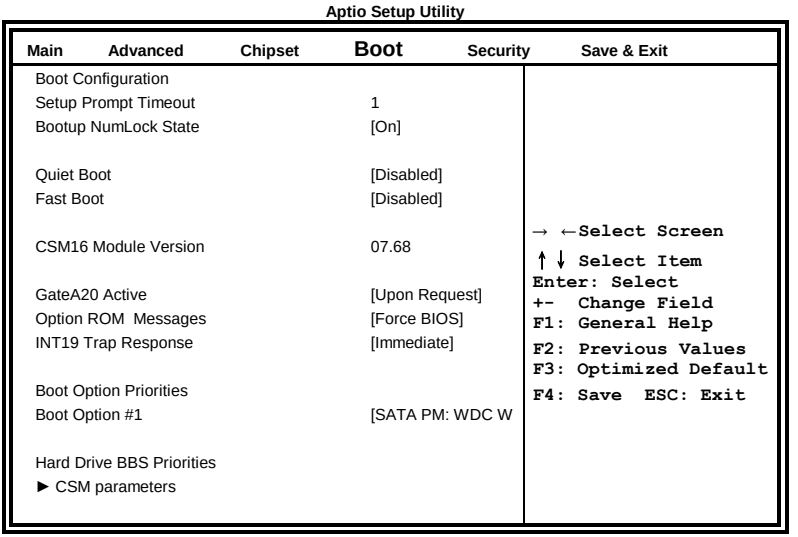
DVMT Total Gfx Mem
Select DVMT 5.0 total graphics memory size used by the internal graphics device.

Gfx Low Power Mode
This option is applicable for SFF only.

Memory Configuration



Boot Settings



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.

Fast Boot
Enables/Disables boot with initialization of a minimal set of devices required to launch active boot option. Has no effect for BBS boot options.

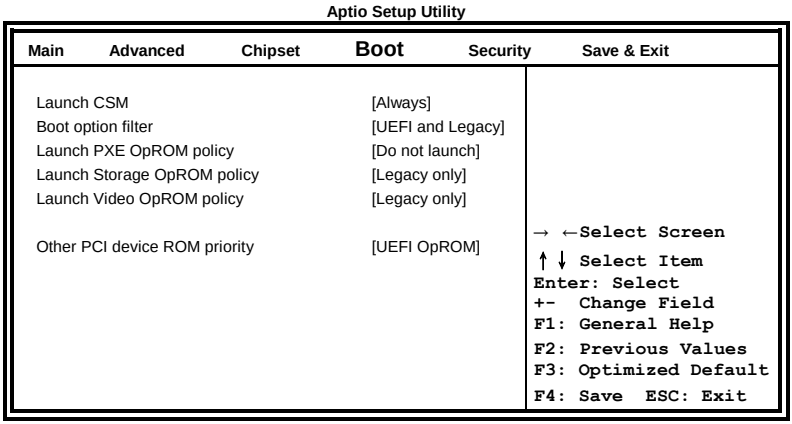
GateA20 Active
UPON REQUEST – GA20 can be disabled using BIOS services.
ALWAYS – do not allow disabling GA20; this option is useful when any RT code is executed above 1MB.

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

INT19 Trap Response
Enable: Allows Option ROMs to trap Int 19.

Boot Option Priorities
Sets the system boot order.

CSM parameters
This section allows you to configure the boot settings.



Boot option filter
This option controls what devices system can boot to.

Launch PXE OpROM policy
Controls the execution of UEFI and Legacy PXE OpROM.

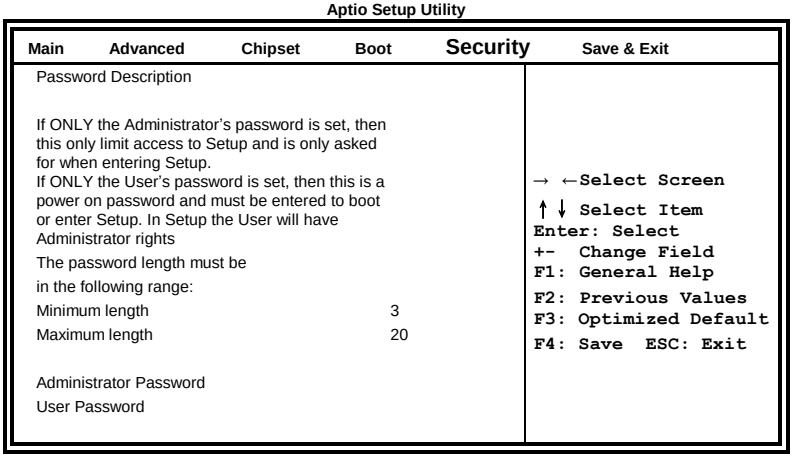
Launch Storatge OpROM policy
Controls the execution of UEFI and Legacy Storage OpROM.

Launch Video OpROM policy
Controls the execution of UEFI and Legacy Video OpROM.

Other PCI device ROM priority
For PCI devices other than Network, Mass storage or Video defines which OpROM to launch.

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.



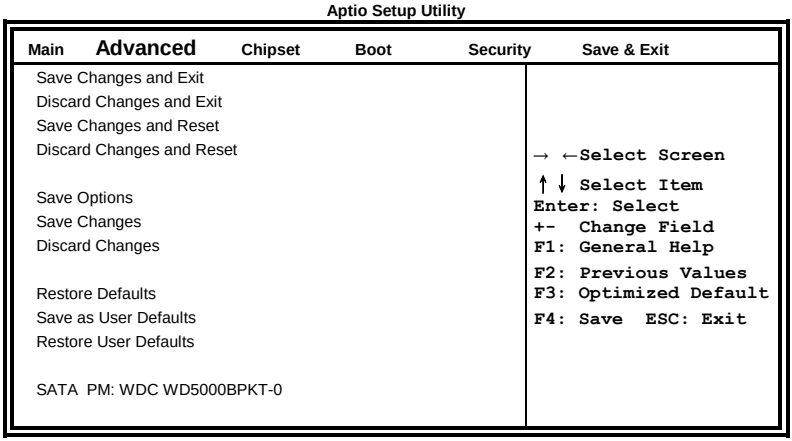
Administrator Password

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.

This page is intentionally left blank.

Drivers Installation

This section describes the installation procedures for software and drivers. 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:

IMPORTANT NOTE:
After installing your Windows operating system, 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.

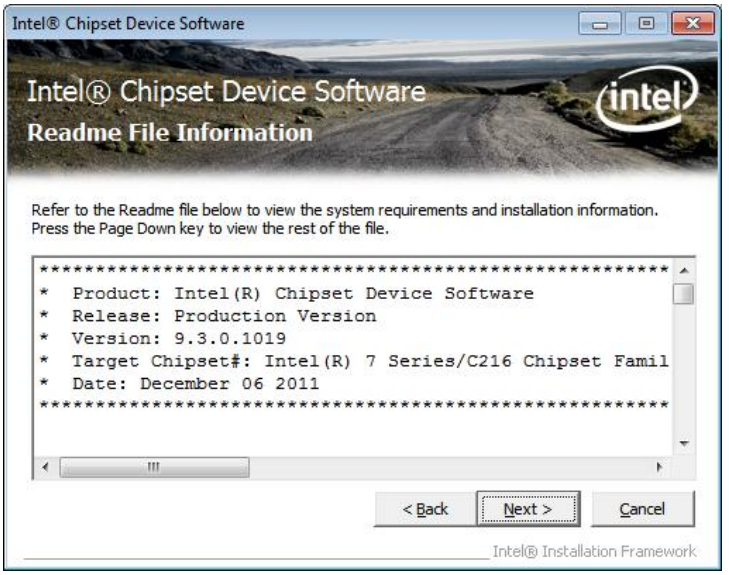
1. Insert the CD that comes with the board. Click ***Intel*** and then ***Intel(R) 7 Series Chipset Drivers***.

2. Click ***Intel(R) Chipset Software Installation Utility.***

3. When the Welcome screen to the Intel® Chipset Device Software appears, click ***Next*** to continue.

4. Click ***Yes*** to accept the software license agreement and proceed with the installation process.

5. On the Readme File Information screen, click ***Next*** to continue the installation.



6. The Setup process is now complete. Click ***Finish*** to restart the computer and for changes to take effect.

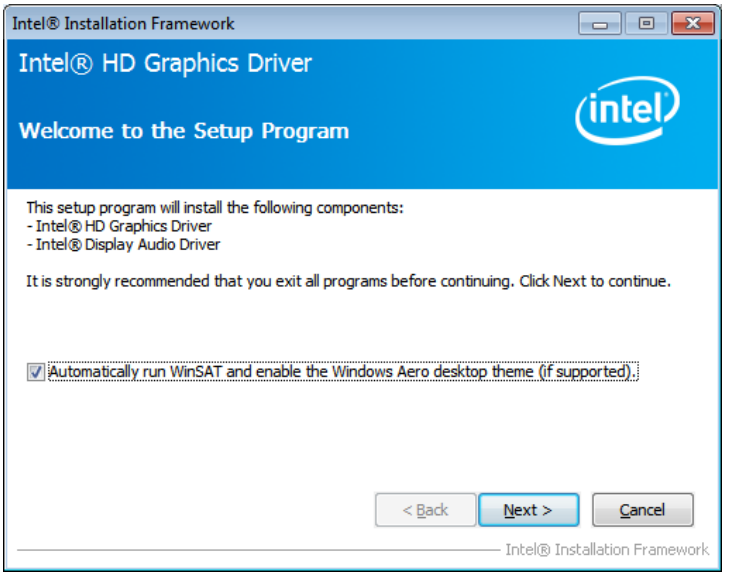
VGA Drivers Installation

NOTE: Before installing the *Intel(R) C216 Chipset Family Graphics Driver,* the Microsoft .NET Framework 3.5 SPI should be first installed.

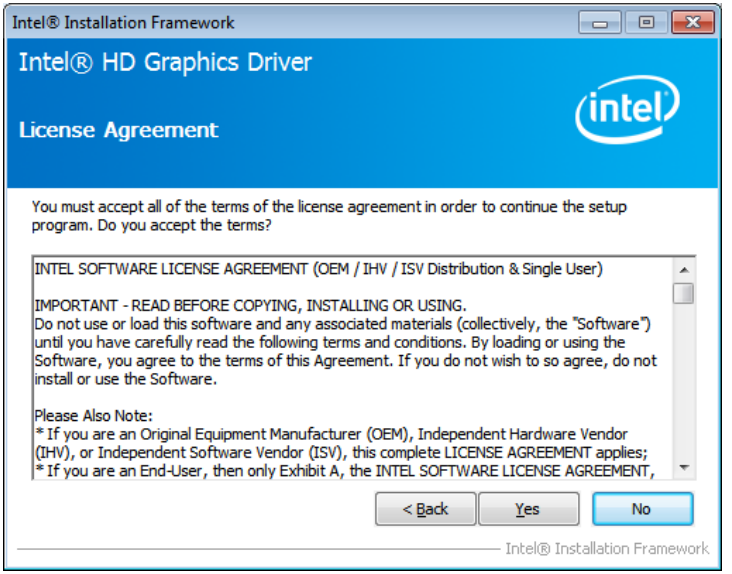
To install the VGA drivers, follow the steps below.
1. Insert the CD that comes with the board. Click ***Intel*** and then ***Intel(R) Q7 Series Chipset Drivers***.

2. Click ***Intel(R) C216 Chipset Family Graphics Driver.***

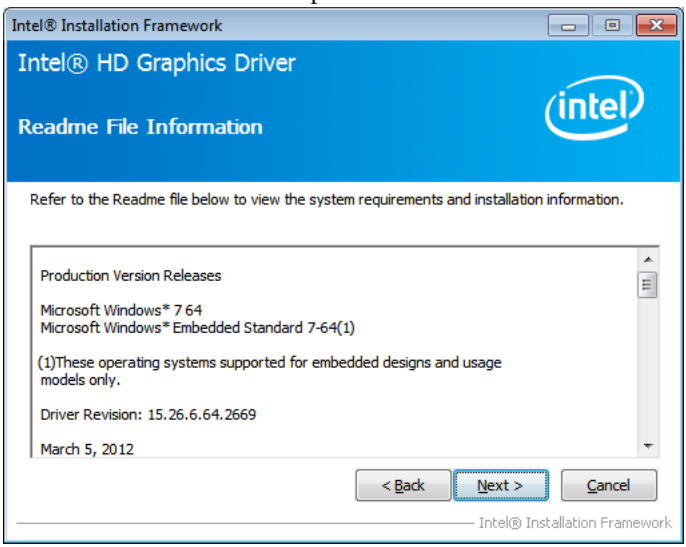
3. When the Welcome screen appears, click ***Next*** to continue.



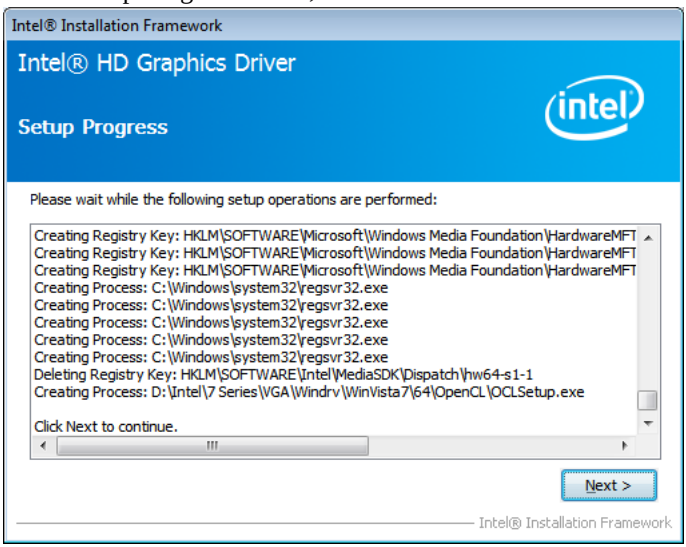
4. Click ***Yes*** to to agree with the license agreement and continue the installation.



5. On the Readme File Information screen, click ***Next*** to continue the installation of the Intel® Graphics Media Accelerator Driver.



6. On Setup Progress screen, click ***Next*** to continue.



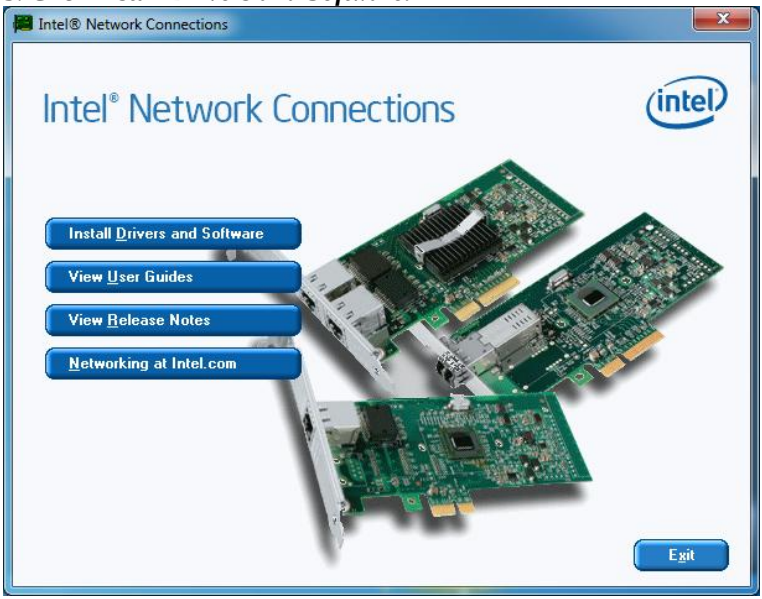
7. Setup complete. Click ***Finish*** to restart the computer and for changes to take effect.

LAN Drivers Installation

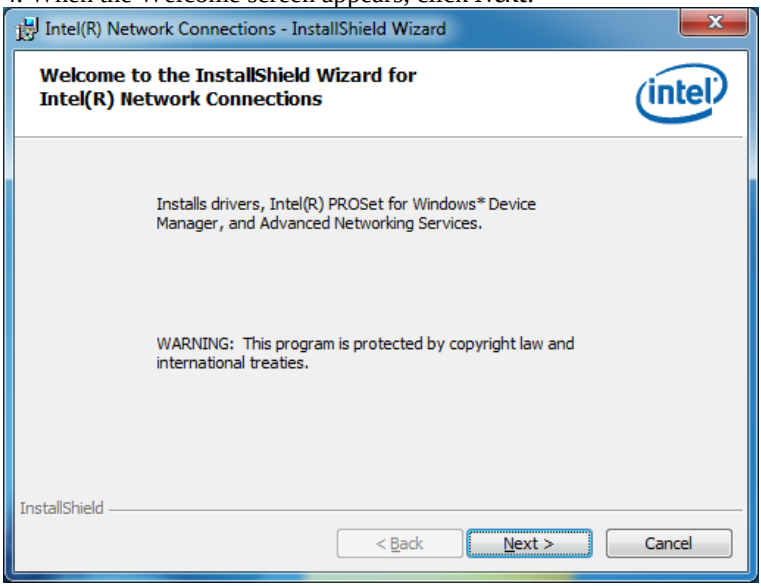
1. Insert the CD that comes with the board. Click ***Intel*** and then ***Intel(R) Q7 Series Chipset Drivers***.

2. Click ***Intel(R) PRO LAN Network Driver.***

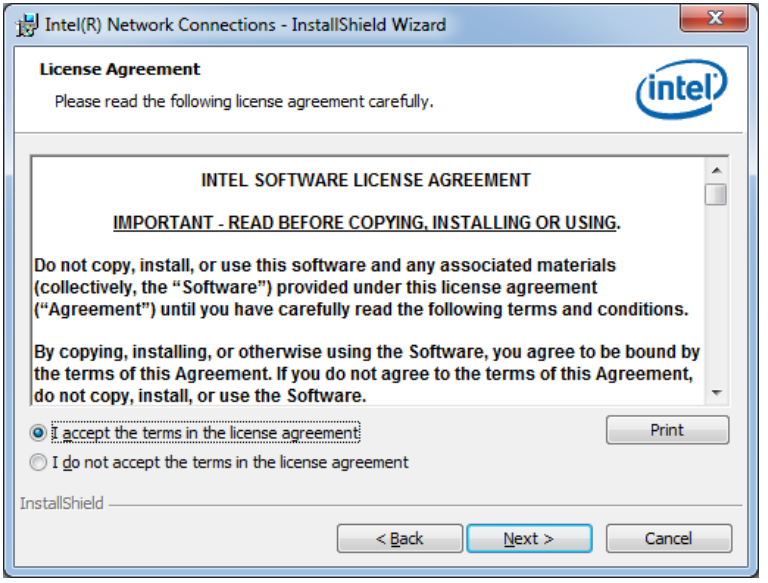
3. Click ***Install Drivers and Software.***



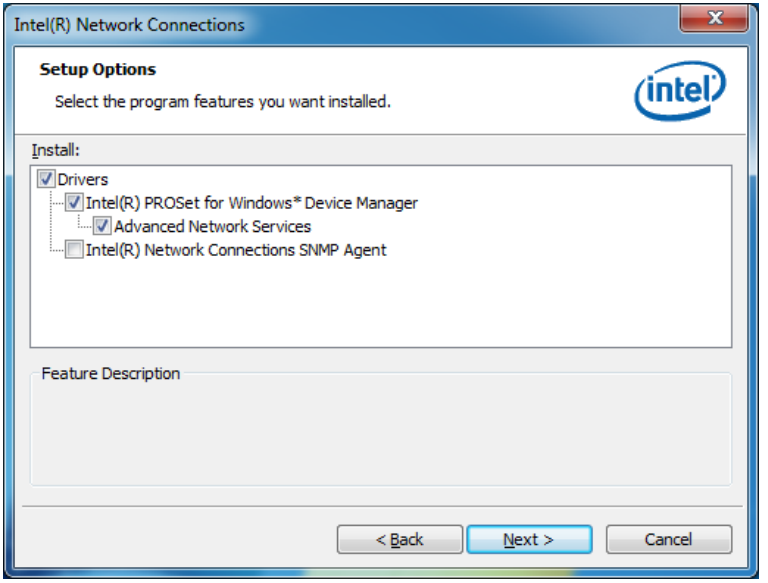
4. When the Welcome screen appears, click ***Next***.



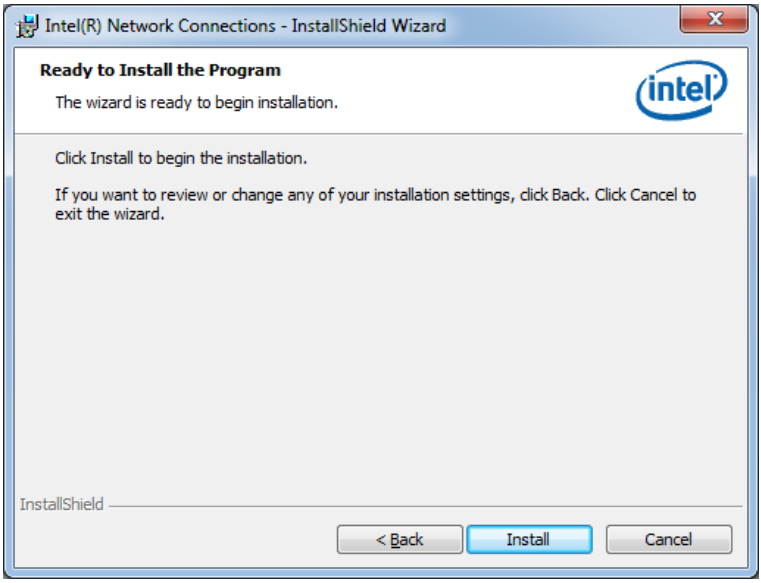
5. Click ***Next*** to to agree with the license agreement.



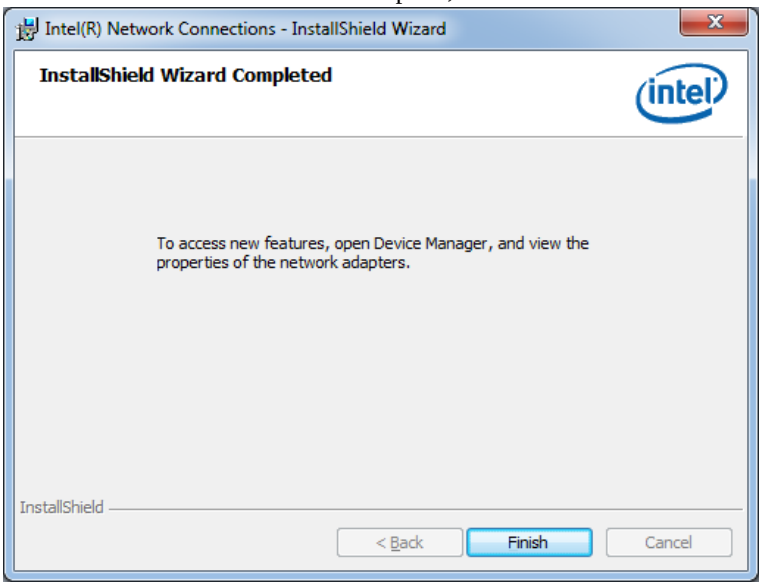
6. Click the checkbox for **Drivers** in the Setup Options screen to select it and click **Next** to continue.



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



8. When InstallShield Wizard is complete, click ***Finish***.



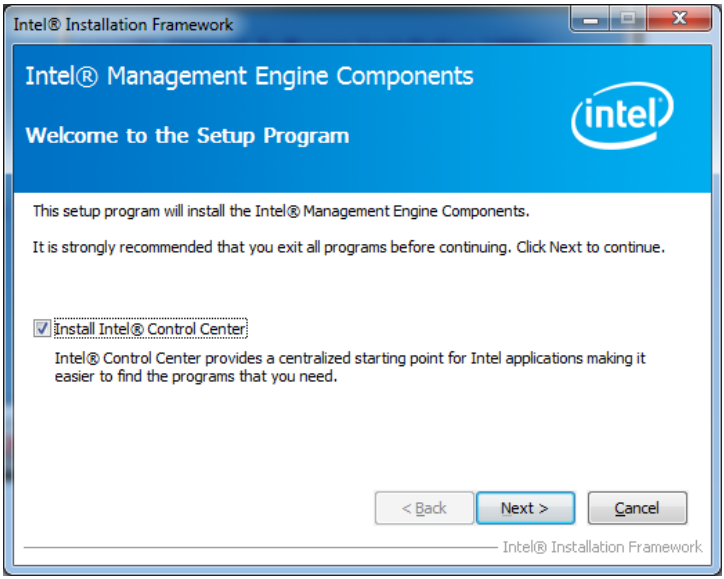
Intel® Management Engine Interface

The following application requires Microsoft .NET Framework 3.5 or later: Intel® Management Engine Components. Please install the latest version of Microsoft .NET Framework from Microsoft Download Center to run this application correctly.

Follow the steps below to install the Intel Management Engine.

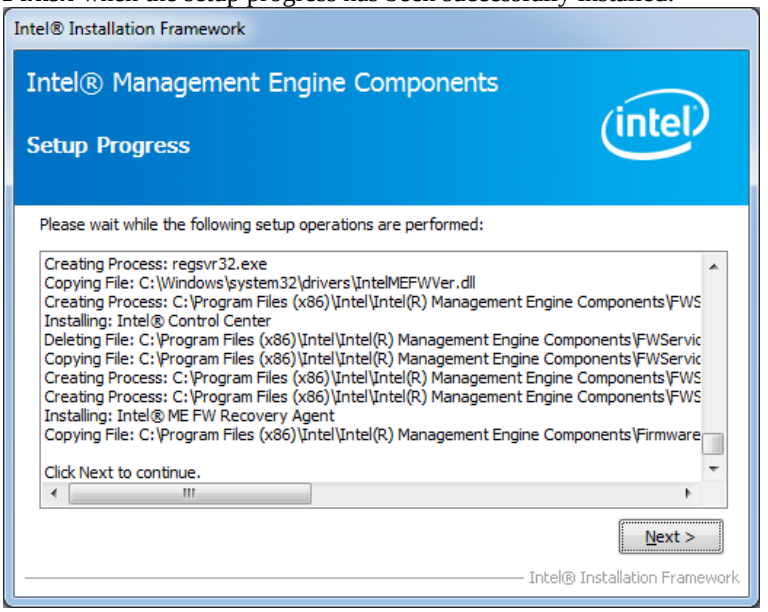
1. Insert the CD that comes with the board. Click ***Intel*** and then ***Intel(R) AMT 8.0 Drivers***.

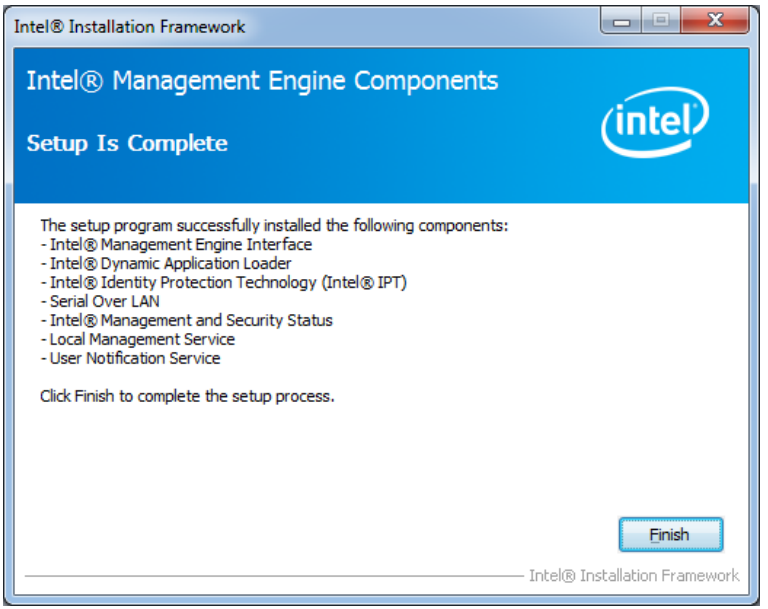
2. When the Welcome screen to the InstallShield Wizard for Intel® Management Engine Components, click the checkbox for **Install Intel® Control Center** & click ***Next***.



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

4. When the Setup Progress screen appears, click ***Next***. Then, click ***Finish*** when the setup progress has been successfully installed.



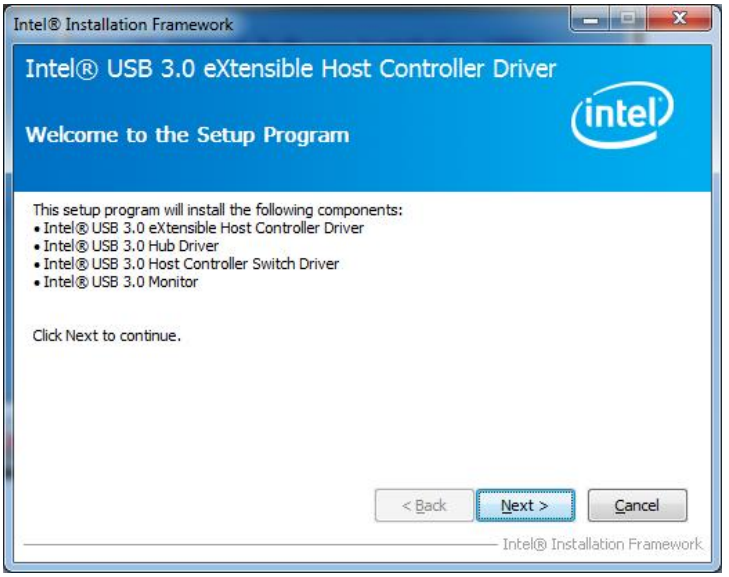


Intel® USB 3.0 Drivers

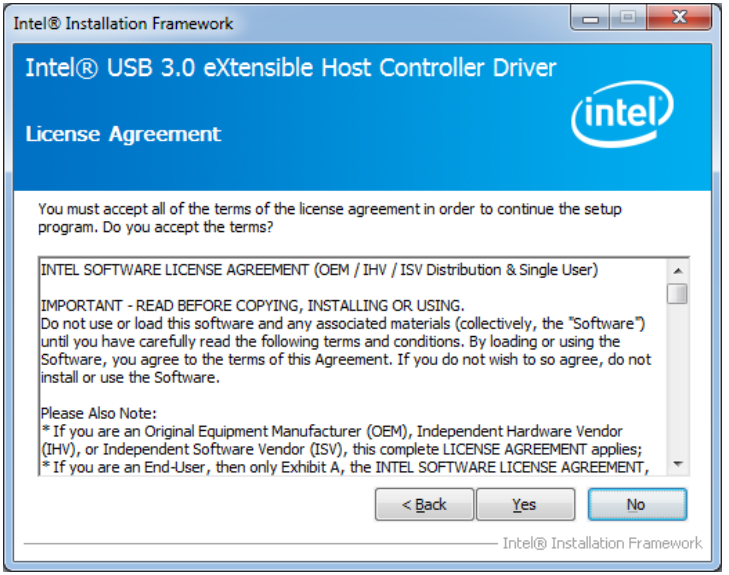
1. Insert the CD that comes with the board. Click ***Intel*** and then ***Intel(R) C216 Series Chipset Drivers***.

2. Click ***Intel(R) USB 3.0 Drivers.***

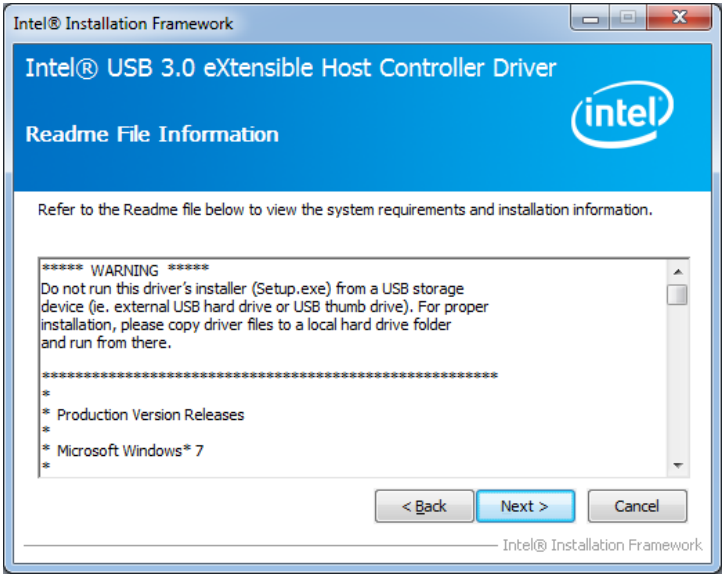
3. When the Welcome screen to the InstallShield Wizard for Intel® USB 3.0 eXtensible Host Controller Driver, click ***Next***.



4. Click ***Yes*** to to agree with the license agreement and continue the installation.



5. On the Readme File Information screen, click ***Next*** to continue the installation of the Intel® USB 3.0 eXtensible Host Controller Driver.



6. Setup complete. Click ***Finish*** to restart the computer and for changes to take effect.

Appendix

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
IRQ3	Serial Port #2
IRQ4	Serial Port #1
IRQ8	Real Time Clock
IRQ14	Primary IDE
IRQ15	Secondary IDE