

ACE-B5692 Gaming Board

**PCI I/O Board with discrete I/O, battery backup SRAM,
Timer, power-off intrusion Event Logger and Protect-U**

User's Manual

Manual Rev: 1.0

Book Number: ACE-B5692-09.05.22

Revision

Version	Date	Author	Description
0.1	2009/05/21	Bear Tsai	Draft
0.2	2009/05/22	Judy Tseng	Revise
1.0	2009/05/22	Bear Tsai	Release

@Copyright 2008**All Rights Reserved.**

Manual's first edition: **May 21, 2009**

For all the purpose of improving reliability, design, function and information in this document are subject to change without prior notice, which does not represent a commitment on the part of the manufacturer.

In no any events will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the using or inability to use the product or documentation, even if advised of the possibility of such damages.

The document contains proprietary information protected by copyright. All rights are reserved. No part of this Manual may be reproduced by any mechanical, electronic, or other means in any form without prior written permission of the manufacturer.

Trademarks

ACE-B5692 is a registered trademarks of Acrosser, IBM PC is a registered trademark of International Business Machines Corporation. Pentium is a registered trademark of Intel Technologies, Inc. Award is a registered trademark of Award Software International, Inc. Others

Product names mentioned herein are used for identification purpose only and may be trademarks and/or registered trademarks of their respective companies.

Table of Contents

CH1 Introduction	
1.1 Specifications	7
1.2 Package Contents	8
1.3 Block Diagram	9
CH2 H/W Information	
2.1 Locations (Top side)	10
2.2 Locations (Bottom side)	11
2.3 Connectors and Jumper Setting	13
2.4 Connectors and Jumper Setting	14
CH3 BIOS	
3.1 Main Setup	21
3.2 Advanced Chipset Setup	22
3.3 Power	24
3.4 PnP/PCI setup	25
3.5 Peripherals Setup	26
3.6 AGC	27
3.7 PC Health	29
3.8 Boot Setup	30
3.9 Exit Setup	31
CH4 AGC Register Description	
4.1 PCI Configuration Register	33
4.2 SRAM Memory Address Map	38
4.3 I/O-Interface Address Map	39
CH5 AGC Driver and Library	
5.1 Windows AGC Driver and Libraries	51
5.2 Linux AGC Driver and Libraries	73
5.3 Error Code Notes	86
CH6 AGA Library	
6.1 Windows AGA Driver and Libraries	88
6.2 Linux AGA Driver and Libraries	96
6.3 Error Code Notes	104

CH7 Electrical Characteristics

7.1 Basic Electrical Characteristics table	106
7.2 72 Pins Golden Fingers	107
7.3 20-pin Golden Finger	108
7.4 JAMMA Golden Finger	109
7.5 Port assignment.....	110
7.6 Spare GPIO connector(2x22Pin 2.0m.m Box header)	114

1 INTRODUCTION

Welcome to ACE-B5692 Gaming Computer. The ACE-B5692 incorporates the advanced Intel® GME965 Chipset. The chipset supports the Intel Core 2 Duo and Celeron M processors, while coming with a 533/667/800MHz Front Side Bus.

ACE-B5692 is a gaming board, which mainly designed for casino gambling machine, such as slot machine. Using low power Pentium M /Celeron M series processor with Acrosser Gaming Controller®/Acrosser Gaming Agent (AGA®) included, ACE-B5692 can satisfy diverse customer applications in different gaming fields.

ACE-B5692 has various physical interfaces in the front panel: 1dual USB Port, 1pcs 10/100/1000 LAN port, and 1 VGA port. In addition, the board provides the capacity for extend DDR2 DIMM module and CF card depending on the user's needs.

The ACE-B5692 is a Complete Platform that totally supports gaming application. It supposes Random Number Generator, Italian JAMMA (Japanese Amusement Machine Manufacturers' Association) and General Gaming interface, non-volatile Memory, and Security Protection.

1.1 Specifications

- Intel® Core 2 Duo Mobile Processor for Mobile Intel 965 Express Chipset Family
- Intel® GME965 & ICH8M chipset
- 30 bits interruptible digital inputs
 - 25 isolated inputs for button, coin, key & bill inputs (4N35, PC817, PC847)
 - 5 TTL level inputs
- 24 bits 500mA outputs, 3 bits 1000mA outputs
- 4 x 16-bit Interruptible Timer
- Three RS-232 and Two ccTalk serial ports/RS232 selectable
- Italian JAMMA and 72-pin golden finger interface
- Intrusion logger
- IButton & Protect-U, security
- Dual 512KB battery back-up SRAM with low-battery monitor
- Second real time clock
- Two 8-bit readable DIP switch
- 6 watts stereo amplifier
- True random number generator

1.2 Package Contents

Check the following items are included in the package.

- The quick manual.
- ACE-B5692.
- 1 software utility CD.

1.3 Black Diagram

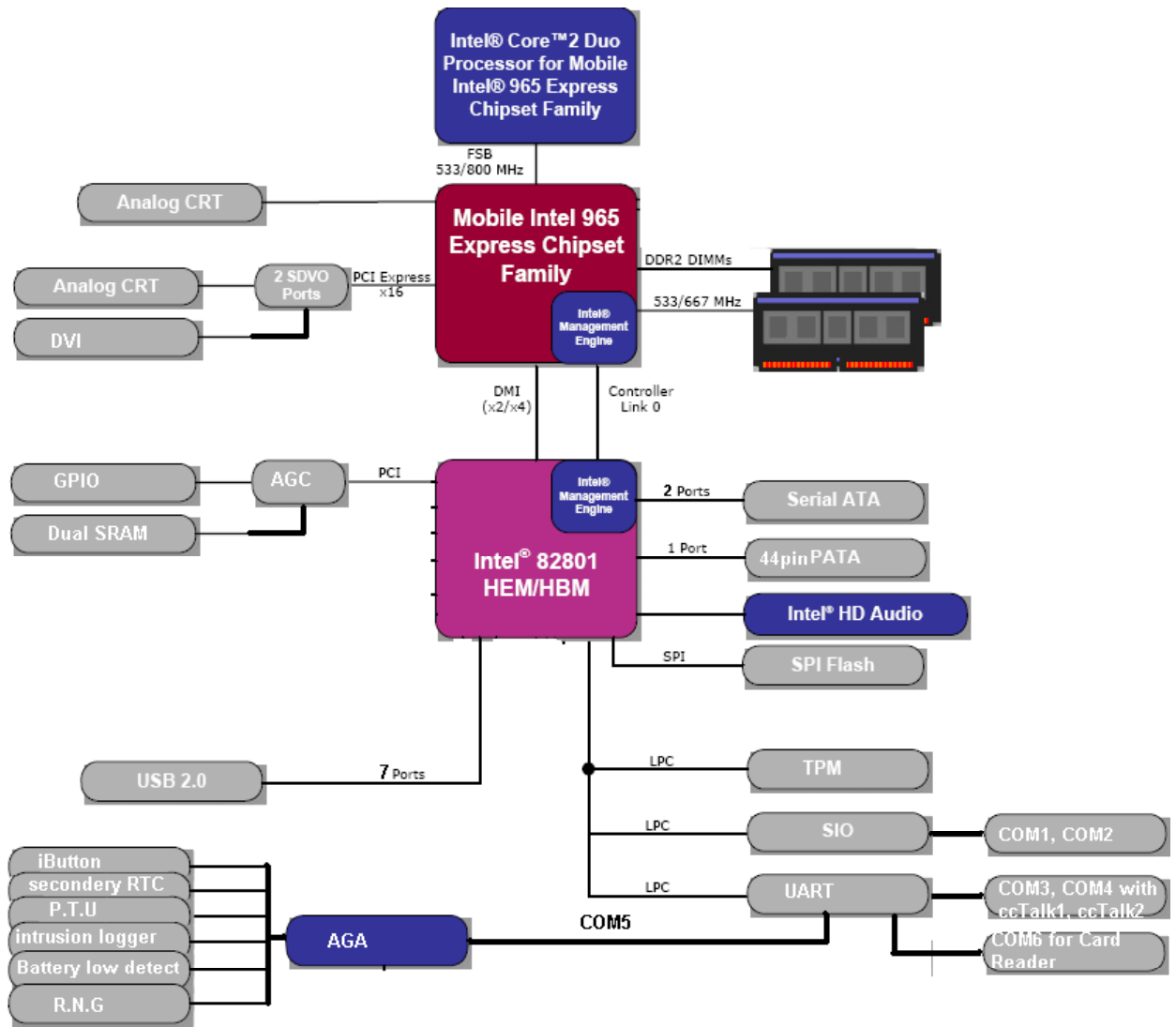


Figure 1: Black Diagram

2

H/W INFORMATION

This chapter describes the hardware information of ACE-B5692. First, we show the function diagram and the layout of ACE-B5692. Then illustrates the unpacking information that you shall be care about, as well as the jumper/switch settings for the ACE-B5692 configuration.

2.1 Locations (Top side)

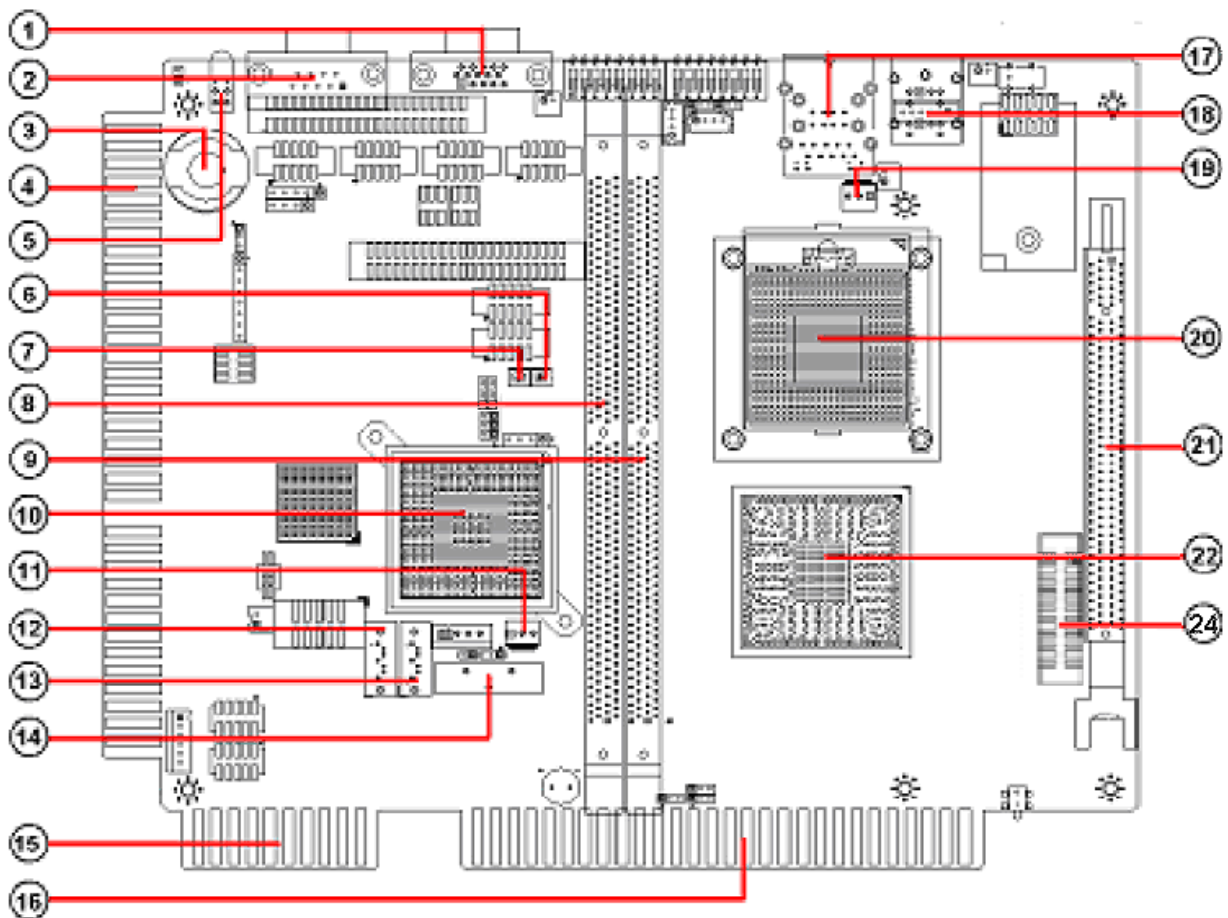


Figure 2: Locations (Top side)

2.2 Locations (Bottom side)

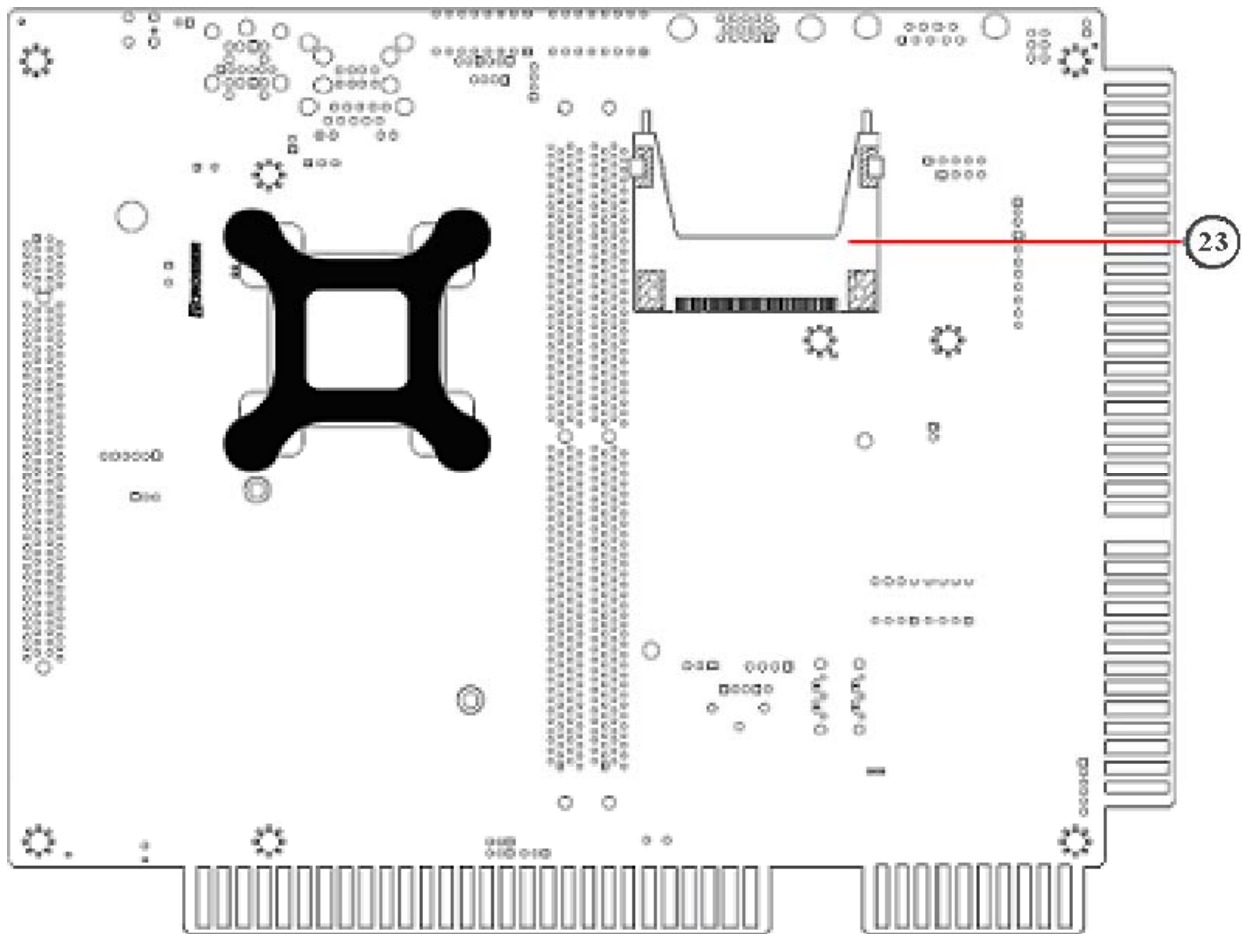


Figure 3: Locations (Bottom side)

1	SCN1 D-Sub 15-pin VGA connector	13	SATA1 Standard 7-pin SATA connector
2	COM1 D-Sub 9-pin RS232 connector	14	CN1 CR2032 Size Coin Battery
3	U83 iButton holder for DS1994 IC.	15	20 Pins Golden Fingers Work with 72 Pins Golden Fingers
4	72 Pins Golden Fingers General Gaming interface	16	JAMMA Interface The type of JAMMA is Italian
5	LED2 LED for Power(Yellow) & HD(Green) & and PS_ON(Red)	17	USB1 One RJ45 with two layer USB connector
6	BAT1 Battery connector for backup SRAM2	18	KM1(optional) Purple for keyboard and Green for mouse
7	BAT2 Battery connector for backup SRAM2/AGA	19	FAN1 3 pin Fan Connector for CPU
8	J2 ^{#1} Dual channel DDR2 DIMM socket	20	Processor Intel Core 2 Duo Mobile Processor
9	J1 Dual channel DDR2 DIMM socket	21	PCIE_1 PCI-Express X16
10	ICH8M I/O Controller Hub 8 M	22	GMCH Graphic Memory Control Hub Intel GME965
11	FAN2 3 pin Fan Connector for system	23	CN5 Standard CF Card slot
12	SATA3 Standard 7-pin SATA connector	24	DVI1 DVI-I pin header

#1 Note: If you use only one DDR2 DIMM module, please first insert the memory module into J2, don't insert it into J1.

2.3 Connectors and Jumper Setting

2.3.1 Locations (Top side)

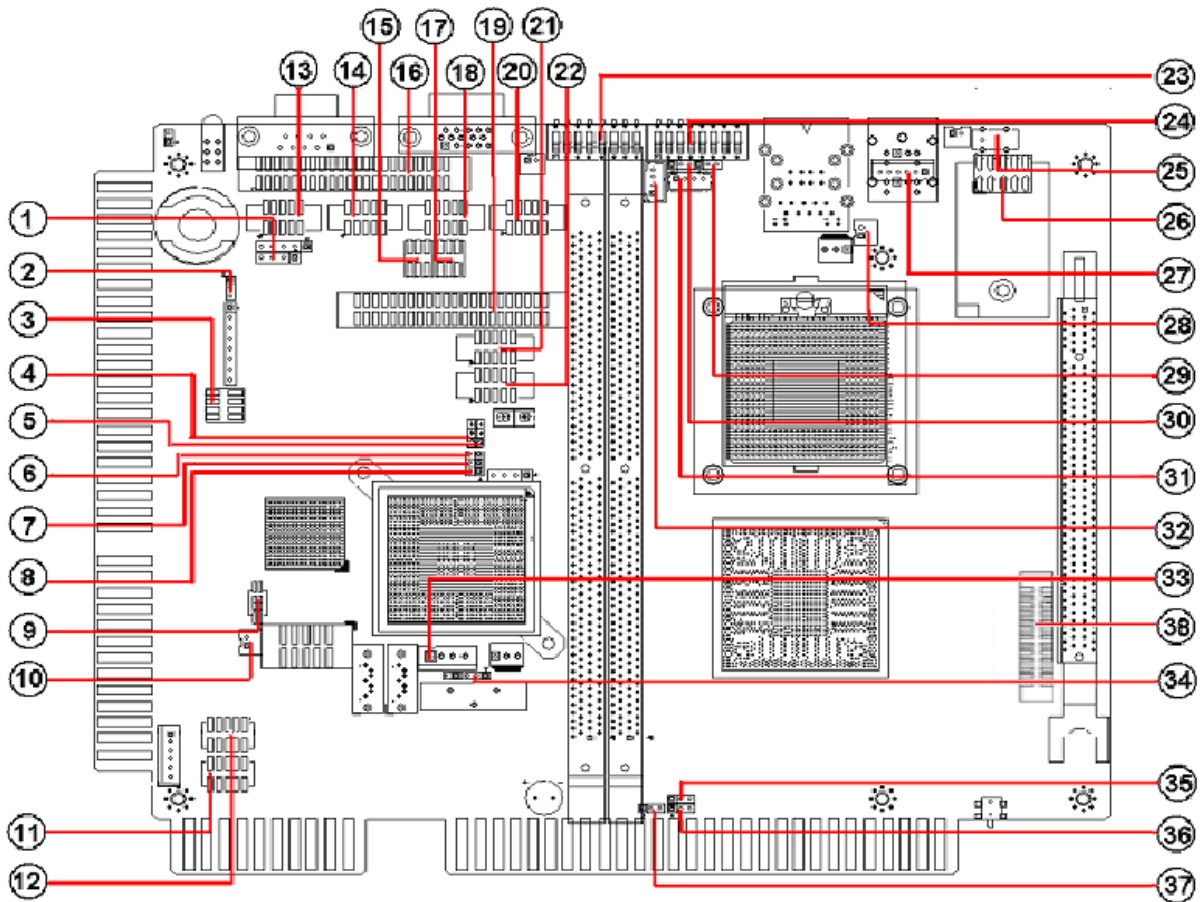


Figure 4: Locations (Top side)

2.4 Connectors and Jumper Setting

2.4.1 CN14 (I2C pin header)

	Pin	description
	1	Vcc
	2	CLK
	3	DATA
	4	GND

2.4.2 JP8 (CF voltage select)

	Pin	description
	1-2 short	+3.3V
	2-3 short	+5V

2.4.3 JP7 (beep/reset/power button/power LED)

	Pin	description	Pin	description
	1	+5V	2	PC BEEP
	3	GND	4	RESET
	5	GND	6	POWER BUTTON
	7	GND	8	+5V_STB

2.4.4 JP13 (SRAM2 clear pin header)

	Pin	description
	1-2 short	normal
	2-3 short	clear SRAM2

2.4.5 JP12 (SRAM1 clear pin header)

	Pin	description
	1-2 short	normal
	2-3 short	clear SRAM1

2.4.6 JP9 (CF Master/Slave pin header)

	Pin 1-2	description
	open	slave
	short	master

2.4.7 JP10 (IDE DMA33/66 pin header)

	Pin 1-2	description
	open	DMA33
	short	DMA66

2.4.8 JP11 (TPM physical presence pin header)

	Pin 1-2	description
	open	P.P
	short	normal

Note: The TPM function is option

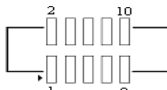
2.4.9 SW7 (Bill and Coin enable pre-set dip switch)

	Pin	State	description
	1(BILL)	ON	pre-set is "Low"
		OFF	pre-set is "High"
	2(COIN)	ON	pre-set is "Low"
OFF		pre-set is "High"	

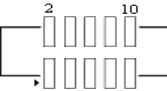
2.4.10 CN3 (System reset pin header)

	Pin1-2	description
	open	normal
	short	reset system

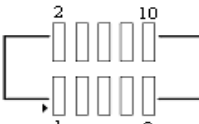
2.4.11 JP15(Audio with AMP or without AMP)

	Pin	description
	1-3, 2-4 short	with amplifier
	3-5, 4-6 short	without amplifier

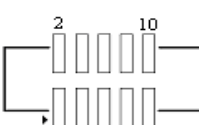
2.4.12 JP14(Audio with AMP or without AMP)

	Pin	description
	1-3, 2-4 short	with amplifier
	3-5, 4-6 short	without amplifier


2.4.13 CN15(COM2)

	Pin	description	Pin	description
	1	DCD	2	DSR
	3	RX	4	RTS
	5	TX	6	CTS
	7	DTR	8	RI
	9	GND	10	NC

2.4.14 CN16(COM3)

	Pin	description	Pin	description
	1	DCD	2	DSR
	3	RX	4	RTS
	5	TX	6	CTS
	7	DTR	8	RI
	9	GND	10	NC

2.4.15 SW5 (select COM3 is RS-232 or ccTalk mode)

	Pin	description
	1-2 ,2-4 short	RS-232 mode
	3-5 ,4-6 short	ccTalk mode

2.4.16 CN19 (Reel/GPIO connector)

	Pin	description	Pin	description
	1	MDAP(GPIO K0)	2	MDAN(GPIO K1)
	3	MDBP(GPIO K2)	4	MDBA(GPIO K3)
	5	GND	6	GND
	7	Lamp_Inn(GPIO K4)	8	Lamp_Out(GPIO K5)
	9	LAMP0(GPIO K6)	10	LAMP1(GPIO K7)
	11	LAMP2(GPIO L0)	12	LAMP3(GPIO L1)
	13	LAMP4(GPIO L2)	14	LED(GPIO L3)
	15	LATCH_A(GPIO L4)	16	LATCH_B(GPIO L5)
	17	GND	18	GND
	19	LATCH_C(GPIO L6)	20	LATCH_D(GPIO L7)
	21	LATCH_E(GPIO XA0)	22	LATCH_F(GPIO XA1)
	23	LATCH_G(GPIO XA2)	24	LATCH_H(GPIO XA3)
	25	FB_A(GPIO XA4)	26	FB_B(GPIO XA5)
	27	+5V	28	+5V
	29	FB_C(GPIO XA6)	30	FB_D(GPIO XA7)
	31	FB_E(GPIO XB0)	32	FB_F(GPIO XB1)
	33	+12V	34	+12V
	35	FB_G(GPIO XB2)	36	FB_H(GPIO XB3)
	37	+3.3V	38	+3.3V
39	GND	40	GND	
41	GND	42	GND	
43	LATCH_OE#	44	NC	

Note: GPIO is standard CMOS 3.3V Input/Output

2.4.17 SW6 (select COM4 is RS-232 or ccTalk mode)

	Pin	description
	1-2 ,2-4 short	RS-232 mode
	3-5 ,4-6 short	ccTalk mode

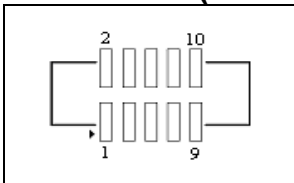
2.4.18 CN17(COM4)

	Pin	description	Pin	description
	1	DCD	2	DSR
	3	RX	4	RTS
	5	TX	6	CTS
	7	DTR	8	RI
	9	GND	10	NC

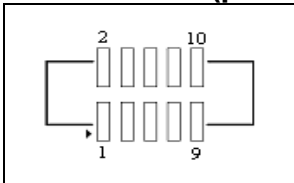
2.4.19 IDE1 (44Pin IDE connector)

	Pin	description	Pin	description
	1	RESET	2	GND
	3	D7	4	D8
	5	D6	6	D9
	7	D5	8	D10
	9	D4	10	D11
	11	D3	12	D12
	13	D2	14	D13
	15	D1	16	D14
	17	D0	18	D15
	19	GND	20	NC
	21	DREQ	22	GND
	23	IOW#	24	GND
	25	IOR#	26	GND
	27	IRDY	28	GND
	29	DACK#	30	GND
	31	IDEIRQ	32	NC
	33	A1	34	PDIAG
	35	A0	36	A2
	37	DCS1#	38	CS3
39	IDE_LED#	40	GND	
41	+5V	42	+5V	
43	GND	44	NC	

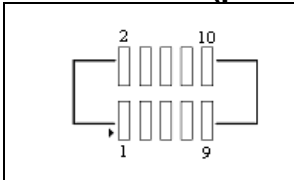
2.4.20 CN18(COM6 for Card Reader)

	Pin	description	Pin	description
	1	DCD	2	DSR
	3	RX	4	RTS
	5	TX	6	CTS
	7	DTR	8	RI
	9	GND	10	+5V

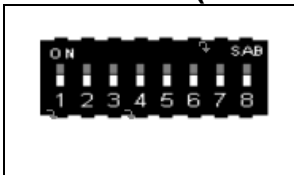
2.4.21 USB2 (pin header for 2 USB ports)

	Pin	description	Pin	description
	1	+5V	2	+5V
	3	USB_A-	4	USB_B-
	5	USB_A+	6	USB_B+
	7	GND	8	GND
	9	GND	10	GND


2.4.22 USB3 (pin header for 2 USB ports)

	Pin	description	Pin	description
	1	+5V	2	+5V
	3	USB_A-	4	USB_B-
	5	USB_A+	6	USB_B+
	7	GND	8	GND
	9	GND	10	GND


2.4.23 SW1 (software readable switch)

	Pin	description
	ON short	GND
	OFF short	+3.3V


2.4.24 SW2 (software readable switch)

	Pin	description
	ON short	GND
	OFF short	+3.3V


2.4.25 SW8 (intrusion log switch)

	description
	Intrusion log switch for AGC Port A bit7


2.4.26 USB4 (Pin header for bNAND USB dongle)

	Pin	description	Pin	description
	1	+5V	2	NC
	3	USBC-	4	NC
	5	USBC+	6	NC
	7	GND	8	NC
	9	NC	10	NC


2.4.27 KBMS1(keyboard/mouse pin header) --> optional

	Pin	description
	1	mouse data
	2	keyboard data
	3	GND
	4	+5V
	5	mouse clock
6	keyboard clock	


2.4.28 CN4 (keyboard lock)

	Pin1-2	description
	open	lock
	short	normal

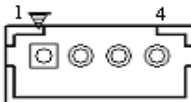
2.4.29 JP17(for cctalk1 voltage select)

	Pin	description
	1-2 short	+12V
	2-3 short	+24V

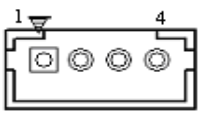
2.4.30 JP18(for cctalk2 voltage select)

	Pin	description
	1-2 short	+12V
	2-3 short	+24V

2.4.31 CCTALK1(COM3 ccTalk connector)

	Pin	description
	1	JP17 select voltage
	2	NC
	3	COM
	4	DATA


2.4.32 CCTALK2(COM4 ccTalk connector)

	Pin	description
	1	JP18 select voltage
	2	NC
	3	COM
	4	DATA

2.4.33 CN6 (SATA power connector)

	Pin	description
	1	+12V
	2	GND
	3	+3.3V
	4	+5V

2.4.34 JP4 (clear CMOS pin header)

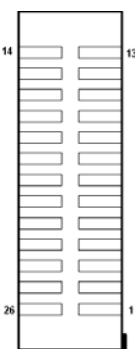
	Pin	description
	1-2 short	normal
	2-3 short	clear CMOS

2.4.35 JP1/2/3 (35,36,37) (FSB setting)

	JP1	JP2	JP3	description
	1-2 short	1-2 short	1-2 short	auto setting
	2-3 short	open	2-3 short	667Mhz
	open	2-3 short	2-3 short	533Mhz

2.4.36 DVI1 (DVI-I pin header)

#Note: Only VGA out, the DVI out is function Option (Grey)

	Pin	description	Pin	description
	1	GND	14	TDC0+
	2	TDC0-	15	GND
	3	TDC1+	16	TDC1-
	4	GND	17	TDC2+
	5	TDC2-	18	GND
	6	TLC+	19	TLC-
	7	DVI DETECT	20	SC DDC DVI
	8	+5V	21	SD DDC DVI
	9	RED	22	GND
	10	GREEN	23	GND
	11	BLUE	24	GND
	12	VSYNC	25	MONSCL
13	HSYNC	26	MONSDA	

3 BIOS SETTING

This chapter describes the BIOS menu and illustrates that how to perform common tasks setting and running. It also gives detail explanation of the elements each found in the BIOS menus. The following topics are:

- Main Setup
- Advanced Chipset Setup
- Power Step
- Peripherals Setup
- PnP/PCI Setup
- AGC Setup
- PC Health Setup
- Boot Setup
- Exit Setup

3.1 Main Setup

While you enter into the Award BIOS™ CMOS Setup Utility, the main menu will appear on your screen. Using the arrow keys to select the item and then using the <Pg Up> <Pg Dn> keys to change the value.

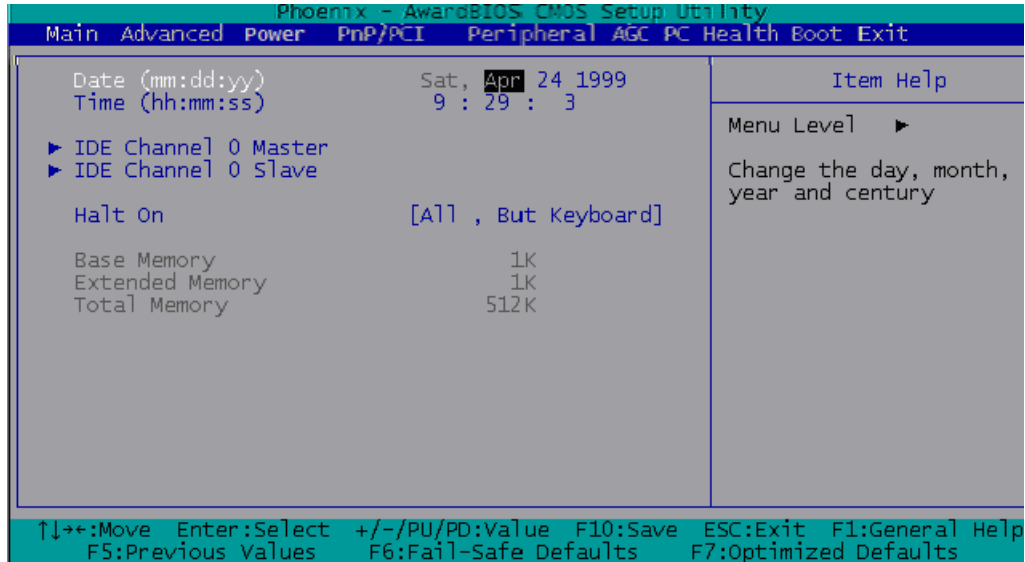


Figure 4: Main setup

Note: The control keys are listing at the bottom of menu. If you need any help in the item fields, please press the <F1> key, and it will display the relevant information.

Option	Choice	Description
Date Setup	N/A	Set the system date. Note that the 'Day' automatically changes when you set the date.
Time Setup	N/A	Set the system time.
IDE Channel 0 Master/Slave	N/A	The onboard IDE connectors provide a channel for connecting up to 2 IDE hard disks or other devices. The first is the "Master" and the second is "Slave", BIOS will auto-detect the IDE type.
Halt On	All Errors, No Errors, All but keyboard.	Select the situation in which you want the BIOS to stop the POST process and notify you.

3.2 Advanced Chipset Setup

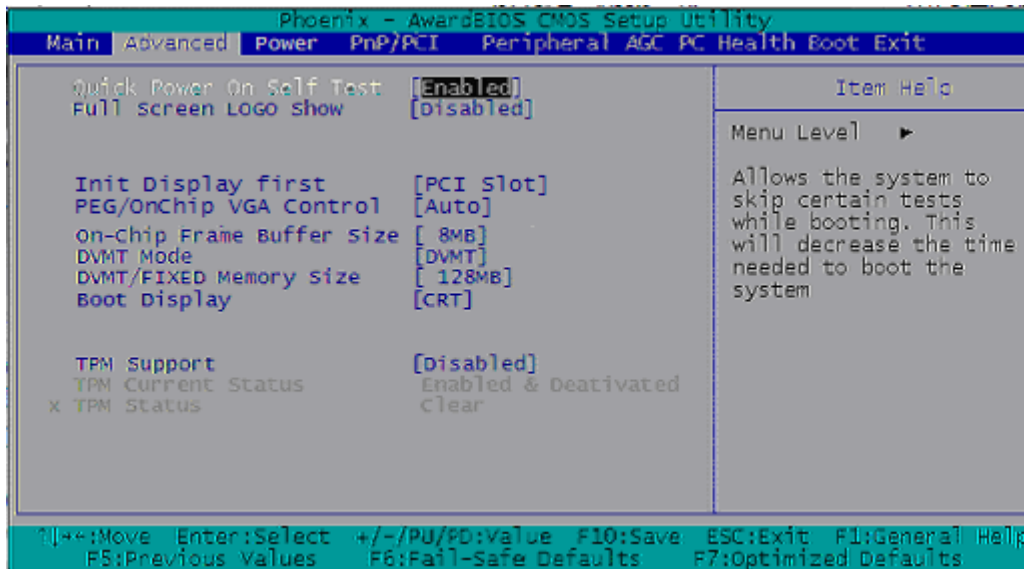


Figure 5: Advance chipset setup

Option	Choice	Description
Quick Power On Self Test	Enabled Disabled	This category speeds up Power On Self Test (POST) after you have powered up the computer. If it is set to Enable, BIOS will shorten or skip some check items during POST.
Full Screen Logo Show	Enabled Disabled	Select to show the OEM full screen logo if you have add-in BIOS.
Init Display First	PCI Slot Onboard	Select Init display first to VGA Card or Onboard VGA.
PEG/On chip VGA Control	Auto PEG Port On Chip	Forced or auto detecting Onboard VGA/ PCIE VGA Card.
On-Chip Frame Buffer Size	1 MB 8 MB	Pre-allocated main memory for onboard VGA frame buffer.
DVMT mode	FIXED DVMT Both	This item sets the mode for OS dynamic video memory technology (DVMT).
DVMT/FIXED Memory Size	128 MB 256 MB MAX.	This item sets the DVMT size
Boot Display	CRT CRT+EFPP1 (Option) CRT+CRT2	For User selected the onboard display combination.
TPM Support	Enabled Disabled	En/disable the TPM Function.

TPM Current Status	N/A	Report TPM Chip current status.
TPM Status	Clear Disabled & Deactivated Enabled & Actifed	To Clear TPM setting or Disabled/Enabled the TPM Chip.

3.3 Power

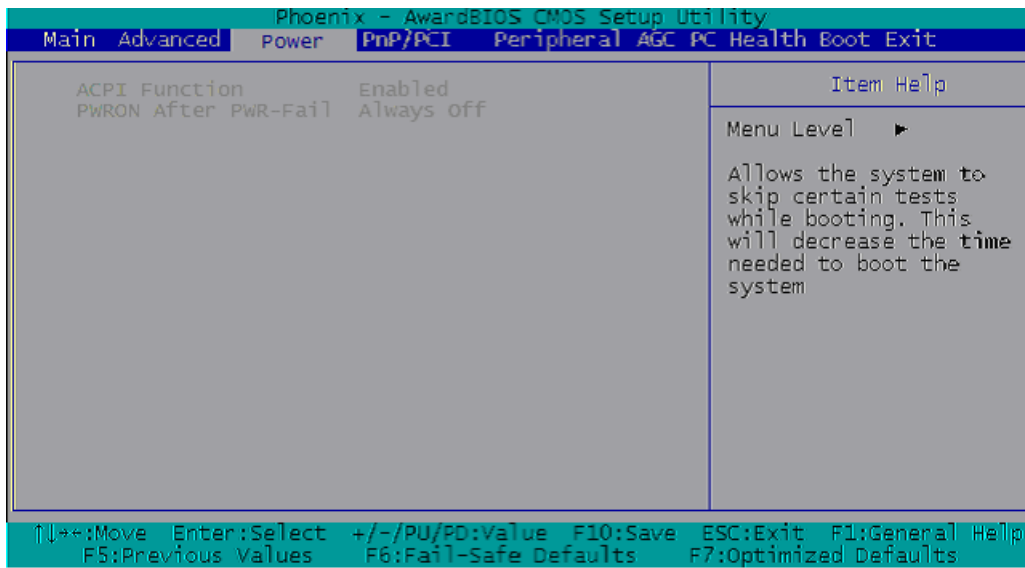


Figure 5: Power setup

Option	Choice	Description
ACPI Function		Default supports ACPI function.

3.4 PnP/PCI setup

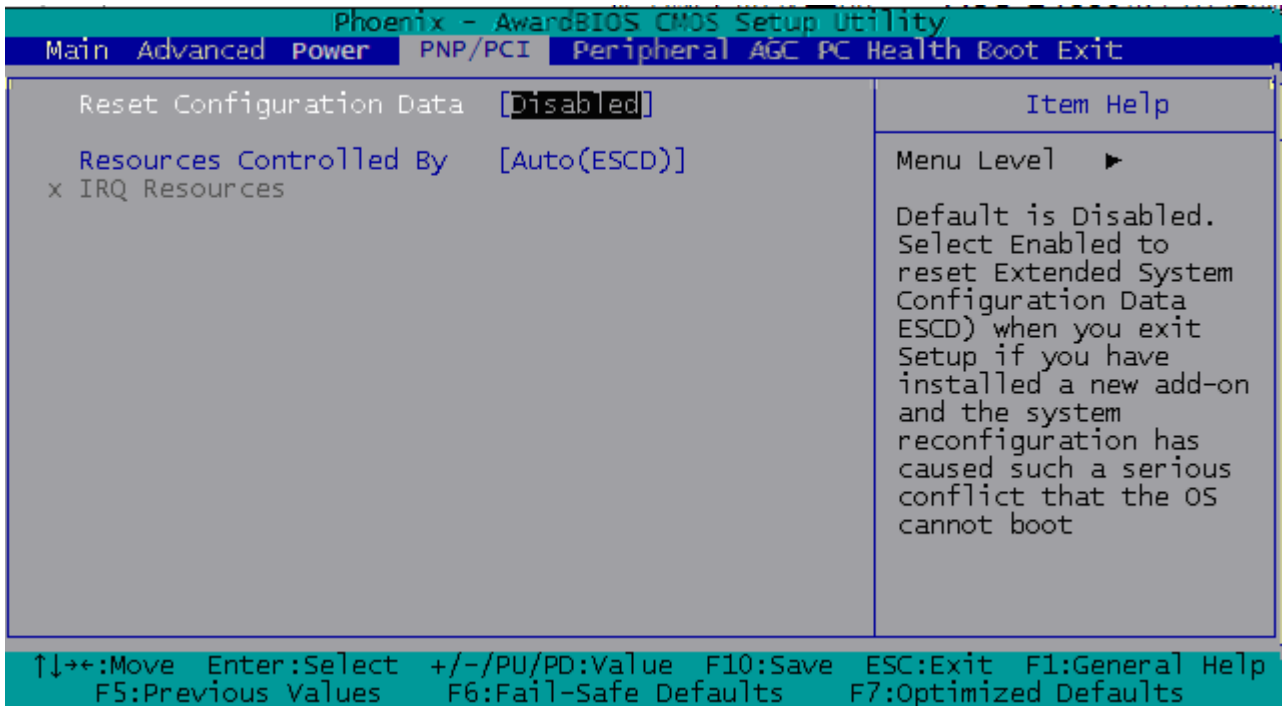


Figure 6: PnP/PCI setup

Option	Choice	Description
Reset Configuration Data	Enabled Disabled	Normally, you need leave this field Disabled. Select enabled to reset Extended System Configuration Data (ESCD) when you exit setup. If you have installed a new add-on and the system reconfiguration has caused such a serious conflict, then the operating system cannot boot.
Resources Controlled By	Auto (ESCD) Manual	The Award Plug and Play BIOS has the capacity to automatically configure all of the boot and Plug and Play compatible devices. However, this capability means absolutely nothing unless you use a Plug and Play operating system such as Windows 95. If you set this field to "manual," then you may choose specific resources by going into each of the submenus.
IRQ Resources	N/A	When resources are controlled manually, assign a type to each system interrupt, depending on the type of the device that uses the interrupt

3.5 Peripherals Setup

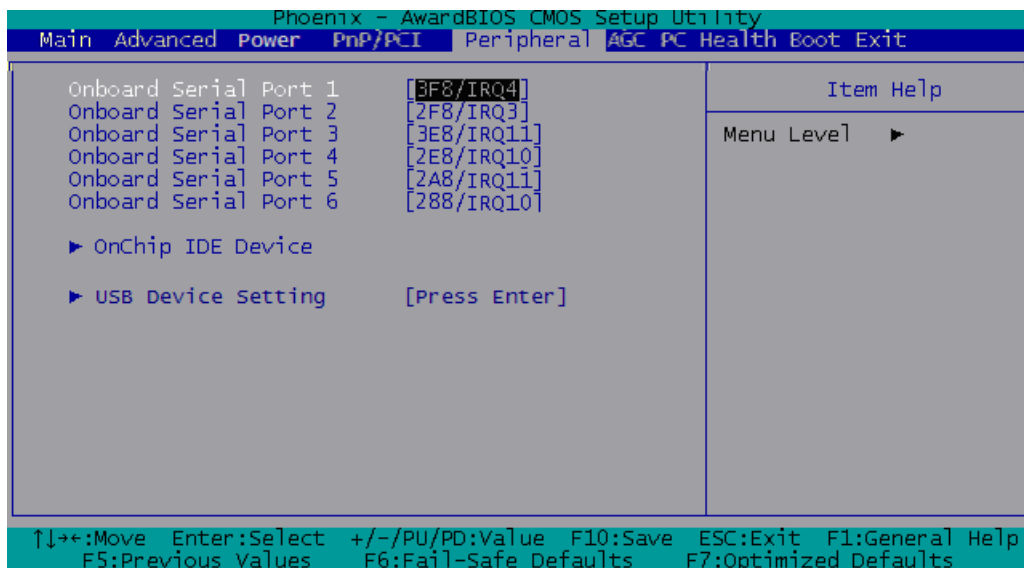


Figure 7: Peripherals setup

Option	Choice	Description
Onboard Serial Port 1	Serial Port 1: 3F8 / IRQ4 Serial Port 2: 2F8 / IRQ3 Serial Port 3: 3E8 / IRQ11 Serial Port 4: 2E8 / IRQ10 Serial Port 5: 2A8 / IRQ11 Serial Port 6: 288 / IRQ10	Select an address and the corresponding interrupt for each serial port.
Onboard Serial Port 2		
Onboard Serial Port 3		
Onboard Serial Port 4		
Onboard Serial Port 5		
Onboard Serial Port 6		

3.6 AGC

This section shows the determining parameters in the AGC Status.

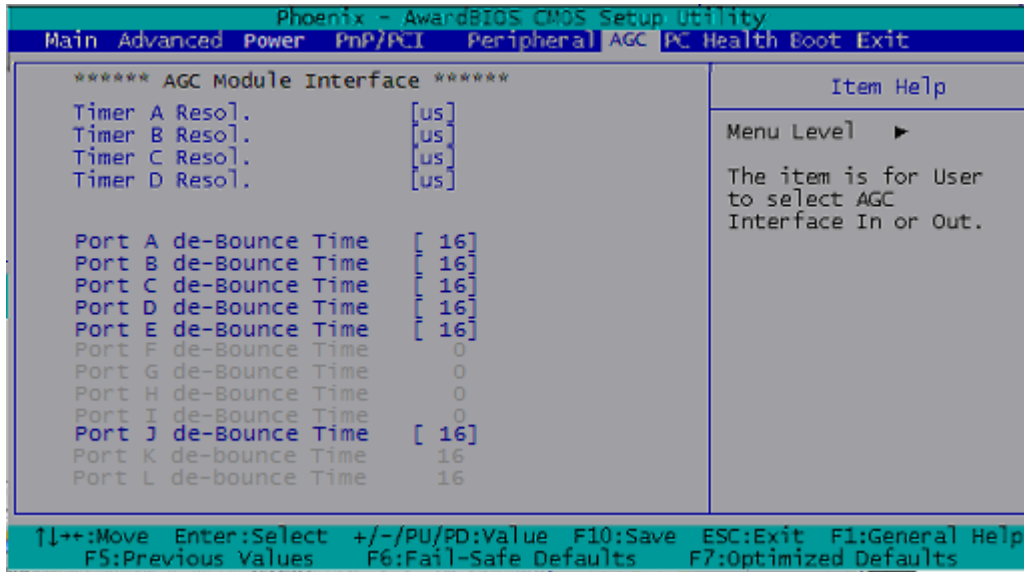


Figure 8: AGC setup

Option	Choice	Description
Timer A Resole.	Sec (Second) MS (Mini-Second) US (Micro-Second)	User can choose the AGC Timer-A resolution. If select "sec", it means 1 second unit. If select "ms", it means 1 mini-second unit. If select "us", it means 1 microsecond unit.
Timer B Resole.	Sec (Second) MS (Mini-Second) US (Micro-Second)	User can choose the AGC Timer-B resolution. If select "sec", it means 1 second unit. If select "ms", it means 1 mini-second unit. If select "us", it means 1 microsecond unit.
Timer C Resole.	Sec (Second) MS (Mini-Second) US (Micro-Second)	User can choose the AGC Timer-C resolution. If select "sec", it means 1 second unit. If select "ms", it means 1 mini-second unit. If select "us", it means 1 microsecond unit.

Timer D Resole.	Sec (Second) MS (Mini-Second) US (Micro-Second)	User can choose the AGC Timer-D resolution. If select "sec", it means 1 second unit. If select "ms", it means 1 mini-second unit. If select "us", it means 1 microsecond unit.
Port A Port B Port C Port D Port E	0 to 255	The value 0 means the de-bounce time is 1 sec/ms/us. The default Setting is 16 (That is 17 sec/ms/us).
Port F Port G Port H Port I	N/A	
Port J	0 to 255	The value 0 means the de-bounce time is 1 sec/ms/us. The default Setting is 16 (That is 17 sec/ms/us).
Port K Port L	N/A	

3.7 PC Health

This section shows the PC Health Status.

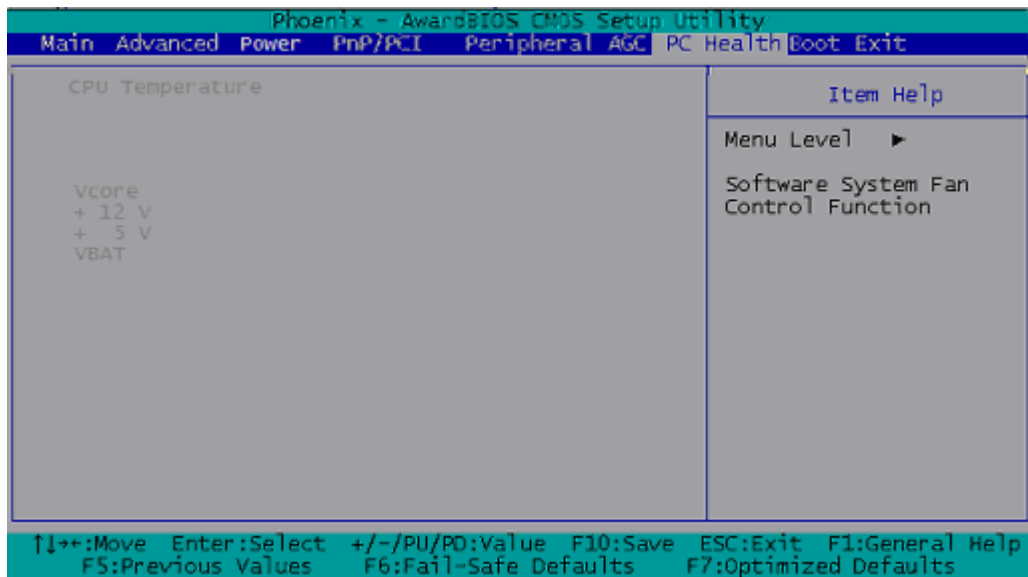


Figure 8: PC Health setup

3.8 Boot Setup

This section shows the determining parameters in the Boot Setup Status.

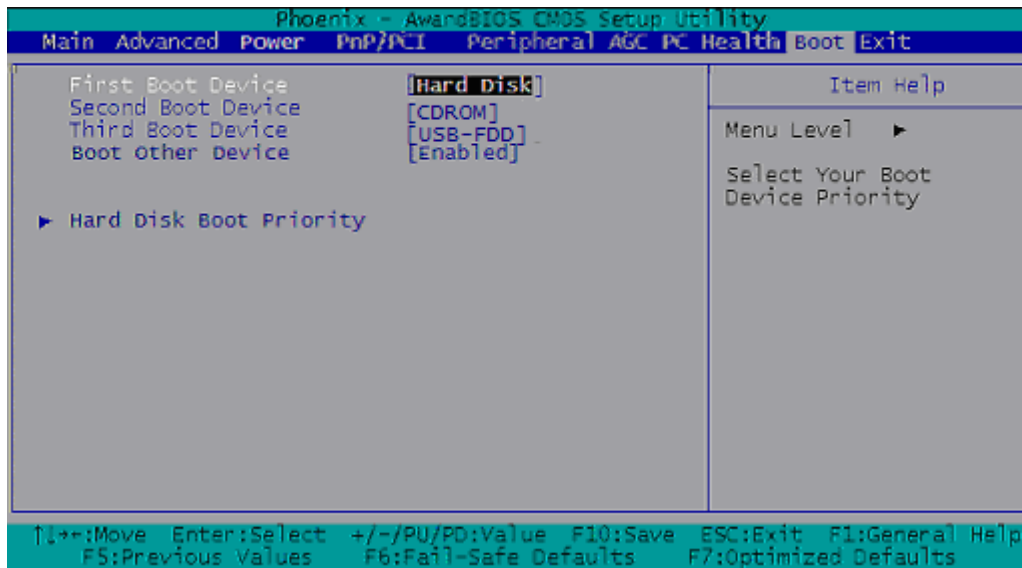


Figure 9: Boot setup

Option	Choice	Description
First Boot Device	Hard Disk CDROM USB-FDD USB-CDROM LAN Disabled	The BIOS attempts to load the operating system from the devices in the sequence selected in these items.
Second Boot Device	Hard Disk CDROM USB-FDD USB-CDROM LAN Disabled	The BIOS attempts to load the operating system from the devices in the sequence selected in these items.
Third Boot Device	Hard Disk CDROM USB-FDD USB-CDROM LAN Disabled	The BIOS attempts to load the operating system from the devices in the sequence selected in these items.
Hard Disk Boot priority		User can define the HDD boot priority. Default is USB Flash always boot first.

3.9 Exit Setup

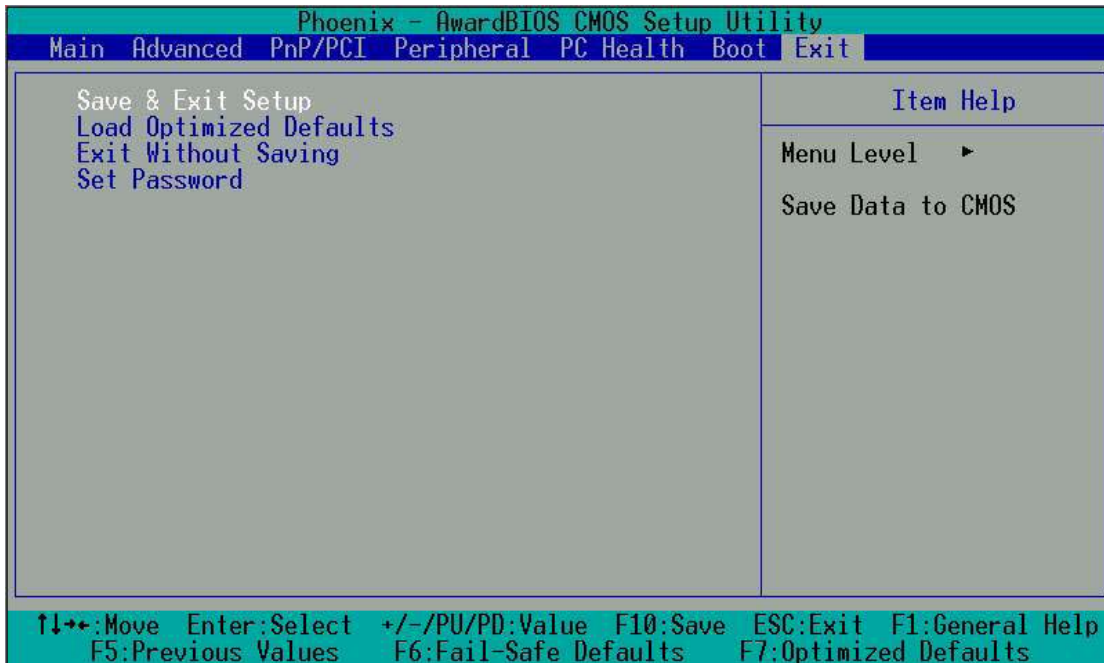


Figure 10: Exit setup

Option	Choice	Description
Save & Exit Setup	Pressing <Enter> on this item for confirmation: Save to CMOS and EXIT (Y/N)? Y	Press “Y” to store the selections made in the menus in CMOS – a special section of memory that stays on after you turn your system off. The next time you boot your computer, the BIOS configures your system according to the setup selections stored in CMOS. After saving the values the system is restarted again
Load Optimized Defaults	When you press <Enter> on this item you get a confirmation dialog box with a message like this: Load Optimized Defaults (Y/N)? N	Press ‘Y’ to load the default values that are factory-set for optimal-performance system operations.
Exit Without Saving	Pressing <Enter> on this item for confirmation: Quit without saving (Y/N)? Y	This allows you to exit setup without storing any changes in CMOS. The previous selections remain in effect. This shall exit the setup utility and restart your computer.

Set Password	Pressing <Enter> on this item for confirmation: ENTER PASSWORD:	<p>When a password has been enabled, you will be prompted to enter your password when you try to enter setup. This prevents unauthorized persons from changing any part of your system configuration.</p> <p>Type the password, up to eight characters in length, and press <Enter>. The password typed now will clear any previous password from the CMOS memory. You will be asked to confirm the password. Type the password again and press <Enter>. You may also press <Esc> to abort the selection and not enter a password.</p> <p>To disable a password, just press <Enter> when you are prompted to enter the password. A message will confirm that the password will be disabled. Once the password is disabled, the system will boot and you can enter Setup freely.</p>
---------------------	---	---

4

AGC REGISTER DESCRIPTION

This chapter describes register function inside an AGC chip. Programming application's software, user must have some information about the AGC chip registers.

4.1 PCI Configuration Register

PCI CFG Register Offset Address	32 bit Register				PCI Readable	PCI Writable
	31 24	23 16	15 8	7 0		
00h	Device ID		Vendor ID		Yes	No
04h	Status		Command		Yes	No
08h	Class Code			Revision ID	Yes	No
0Ch	BIST	Header Type	Latency Timer	Cache Line Size	Yes	No
10h	PCI Base Address 0 for Memory Mapped Configuration Registers				Yes	Yes
14h	PCI Base Address 1 for I/O Mapped Configuration Registers				Yes	Yes
18h	PCI Base Address 2 (Not Supported)				No	No
1Ch	PCI Base Address 3 (Not Supported)				No	No
20h	PCI Base Address 4 (Not Supported)				No	No
24h	PCI Base Address 5 (Not Supported)				No	No
28h	Cardbus CIS Pointer (Not Supported)				Yes	No
2Ch	Subsystem ID		Subsystem Vendor ID		Yes	No
30h	PCI Base Address for Local Expansion ROM (Not Supported)				Yes	No
34h	Reserved				No	No
38h	Reserved				No	No
3Ch	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	Yes	Yes / No

Vendor ID Register (00h : 01h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
15 : 0	Vendor ID. Identifies manufacturer of the device.	Yes	No	1204h

Device ID Register (02h : 03h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 16	Device ID. Identifies particular device.	Yes	No	5692h

Command Register (04h : 05h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	I/O Space. Value of 1 allows device to respond to I/O space accesses.	Yes	No	1
1	Memory Space. Value of 1 allows device to respond to memory space accesses	Yes	No	1
2	Master Enable. Value of 0 disables device from generating bus master accesses. Not Supported	Yes	No	0
3	Special Cycle. Not Supported.	Yes	No	0
4	Memory Write/Invalidate. Not Supported.	Yes	No	0
5	VGA Palette Snoop. Not Supported.	Yes	No	0
6	Parity Error Response. Not Supported.	Yes	No	0
7	Wait Cycle Control. Not Supported.	Yes	No	0
8	SERR# Enable. Not Supported.	Yes	No	0
9	Fast Back-to-Back Enable. Not Supported.	Yes	No	0
15 : 10	Reserved	Yes	No	0

Status Register (06h : 07h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
22 : 16	Reserved	Yes	No	0
23	Fast Back-to-Back Capable. Not Supported.	Yes	No	0
24	Master Data Parity Error Detected. Not supported	Yes	No	0
26 : 25	DEVSEL Timing. Value of 01 is Slow .	Yes	No	10
27	Target Abort. 1 if Device has Signal Target Abort.	Yes	Yes	0
28	Received Target Abort. Not Supported.	Yes	No	0
29	Received Master Abort. Not Supported.	Yes	No	0
30	Signaled System Error. Not Supported.	Yes	No	0
31	Detected Parity Error. Not Supported.	Yes	No	0

Revision ID Register (08h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
7 : 0	Revision ID. Identifies particular device.	Yes	No	0000

Class Code Register (09h : 0Bh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
15 : 8	Specific Register Level Programming Interface (00h). No interface defined.	Yes	No	00
23 : 16	Subclass Encoding (80h). Other bridge device.	Yes	No	80h
31 : 24	Base Class Encoding. Other bridge Device.	Yes	No	06h

Cache Line Size Register (0Ch)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
7 : 0	System Cache Line Size (in units of 32-bit words). Can be written and read; however, the value has no effect on operation of chip.	Yes	No	0

Latency Timer Register (0Dh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset

15 : 8	PCI Latency Timer. Not Supported.	Yes	No	0
--------	--	-----	----	---

Header Type Register (0Eh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
22 : 16	Configuration Layout Type. Specifies layout of bits 10h through 3Fh in configuration space. Only one encoding 0 is defined. All other encodings are reserved.	Yes	No	0
23	Header Type. Value of 1 indicates multiple functions. Value of 0 indicates a single Function.	Yes	No	0

Built-In Self Test Register (0Fh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 24	Built-In Self Test. Value of 0 indicates that device has passed its test. Not Supported.	Yes	No	0

Base Address 0 Registers for Memory Accesses to Local Configuration (10h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Memory Space Indicator. Value of 0 indicates register maps into Memory space. Value of 1 indicates register maps into I/O space.	Yes	No	0
2 : 1	Location of register: 00 = Locate anywhere in 32 bit memory address space 01 = Locate below 1 MB memory address space 10 = Locate anywhere in 64 bit memory address space 11 = Reserved	Yes	No	0
3	Prefetchable. Value of 1 indicates no side effect on reads.	Yes	No	0
6 : 4	Memory Base Address. Memory base address for access to local configuration registers. (Default 1024 Kbytes)	Yes	No	0
31 : 7	Memory Base Address. Memory base address for access to local configuration registers.	Yes	Yes	0

Base Address 1 Register for I/O Accesses to Local Configuration (14h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Memory Space Indicator. Value of 0 indicates register maps into Memory space. Value of 1 indicates register maps into I/O space.	Yes	No	1
1	Reserved	Yes	No	0
6 : 2	I/O Base Address. Base address for I/O access to local configuration registers (default 128 bytes).	Yes	No	0
31 : 7	I/O Base Address. Base address for I/O access to local configuration registers	Yes	Yes	0

Base Address 2 Registers (18h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 0	Not Supported	Yes	No	0

Base Address 3 Registers (1Ch)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 0	Not Supported	Yes	No	0

Base Address 4 Registers (20h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 0	Not Supported	Yes	No	0

Base Address 5 Registers (24h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 0	Not Supported	Yes	No	0

Cardbus CIS Pointer Registers (28h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 0	Card bus Information Structure Pointer for PCMCIA. Not Supported.	Yes	No	0

Subsystem Vendor ID Registers (2Ch)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
15 : 0	Subsystem Vendor ID (Unique add-in board Vendor ID)	Yes	Yes	00

Subsystem ID Registers (2Eh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 16	Subsystem ID. (Unique add-in board Device ID)	Yes	Yes	00

Base Address for Local Expansion ROM Registers (30h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 0	Not Supported	Yes	No	0

Interrupt Line Registers (3Ch)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
7 : 0	Interrupt Line Routing Value indicates which system interrupt controller(s) input the interrupt line of device is connected to.	Yes	Yes	0

Interrupt Pin Registers (3Dh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
15 : 8	Interrupt Pin Register indicates the interrupt pin that the device uses. The following values are decoded: 0 = No Interrupt Pin 1 = INTA# 2 = INTB# 3 = INTC# 4 = INTD# Note: supports only one PCI interrupt (INTA#).	Yes	No	2

Min Gnt Registers (3Eh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
23 : 16	Min Gnt. Specifies needed length of Burst period for the device, assuming a clock rate of 33 MHz. Value is a multiple of 1/4 as increments. Not Supported.	Yes	No	0

Max Lat Registers (3Fh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
31 : 24	Max Lat. Specifies how often the device must gain access to PCI bus. Value is a multiple of 1/4 as increments. Not Supported.	Yes	No	0

4.2 SRAM Memory Address Map

The following table shows the SRAM Memory Address map (max. 1024 KB) and their offset addresses, relative to the “**PCI Base Address 0**”. To access SRAM memory, user must use Byte-Access command.

Memory Offset Address	32 bit Data width				Software Readable	Software Writable
	31 24	23 16	15 8	7 0		
00h	Byte 3	Byte 2	Byte 1	Byte 0	Yes	Yes
04h	Byte 7	Byte 6	Byte 5	Byte 4	Yes	Yes
08h	Byte 11	Byte 10	Byte 9	Byte 8	Yes	Yes
....	Yes	Yes
....	Yes	Yes
FFFF4h	Byte 8183	Byte 8182	Byte 8181	Byte 8180	Yes	Yes
FFFF8h	Byte 8187	Byte 8186	Byte 8185	Byte 8184	Yes	Yes
FFFFCh	Byte 8191	Byte 8190	Byte 8189	Byte 8188	Yes	Yes

4.3 I/O-Interface Address Map

The following table shows the I/O Address map, including descriptions and their offset addresses relative to the **“PCI Base Address1”**.

I/O Offset Address	32 bit Register				Software Readable	Software Writable
	31 24	23 16	15 8	7 0		
00h	Reserved			SRAM operation mode	Yes	Yes
04h	Reserved			DIP Switch	Yes	No
08h	Reserved		Interrupt & Timer Enable Register		Yes	Yes
0Ch	Reserved		I/O & Timer Interrupt Source Registers		Yes	No
10h	Reserved			Port BCD Mode	Yes	Yes
14h	Reserved			Port A Data	Yes	Yes
18h	Reserved			Port B Data	Yes	Yes
1Ch	Reserved			Port C Data	Yes	Yes
20h	Reserved			Port D Data	Yes	Yes
24h	Reserved			Port EFGH Mode	Yes	Yes
28h	Reserved			Port E Data	Yes	Yes
2Ch	Reserved			Port F Data	Yes	Yes
30h	Reserved			Port G Data	Yes	Yes
34h	Reserved			Port H Data	Yes	Yes
38h	Reserved			Port IJ Mode	Yes	Yes
3Ch	Reserved			Port I Data	Yes	Yes
40h	Reserved			Port J Data	Yes	Yes
44h	Reserved			Reserved	Yes	Yes
48h	Reserved			Reserved	Yes	Yes
4Ch			TIMER-A Register		Yes	Yes
50h			TIMER-B Register		Yes	Yes
54h			TIMER-C Register		Yes	Yes
58h			TIMER-D Register		Yes	Yes
5Ch				Timer Resolution	Yes	Yes
60h	Port D de-bounce	Port C de-bounce	Port B de-bounce	Port A de-bounce	Yes	Yes
64h	Port H de-bounce	Port G de-bounce	Port F de-bounce	Port E de-bounce	Yes	Yes
68h	Reserved	Reserved	Port J de-bounce	Port I de-bounce	Yes	Yes

SRAM Bank Select (00h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	SRAM mode select. '0' => Independent mode. You can use memory base address FDB00000~FDBFFFFFF total 1024KB space. '1' => replicate mode. Write data in the base address FDB00000~FDB7FFFF total 512KB space, you can read same data in the FDB80000~FDBFFFFFF. It had backup function But FDB80000~FDBFFFFFF space is read only.	Yes	Yes	0
7 : 1	Reserved	Yes	Yes	0

DIP Switch (04h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	DIP Switch1	Yes	No	0 / 1
1	DIP Switch2	Yes	No	0 / 1
2	DIP Switch3	Yes	No	0 / 1
3	DIP Switch4	Yes	No	0 / 1
4	DIP Switch5	Yes	No	0 / 1
5	DIP Switch6	Yes	No	0 / 1
6	DIP Switch7	Yes	No	0 / 1
7	DIP Switch8	Yes	No	0 / 1

Interrupt & Timer Enable Register (08h & 09h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Port A Interrupt Enable bit. '0' = No support Interrupt from Port A as Input; '1' = Support Interrupt from Port A as Input.	Yes	Yes	0
1	Port B Interrupt Enable bit. '0' = No support Interrupt from Port B as Input; '1' = Support Interrupt from Port B as Input.	Yes	Yes	0
2	Port C Interrupt Enable bit. '0' = No support Interrupt from Port C as Input; '1' = Support Interrupt from Port C as Input.	Yes	Yes	0
3	Port D Interrupt Enable bit. '0' = No support Interrupt from Port D as Input; '1' = Support Interrupt from Port D as Input.	Yes	Yes	0
4	Port E Interrupt Enable bit. '0' = No support Interrupt from Port E as Input; '1' = Support Interrupt from Port E as Input.	Yes	Yes	0
5	Port F Interrupt Enable bit. '0' = No support Interrupt from Port F as Input; '1' = Support Interrupt from Port F as Input.	Yes	Yes	0
6	Port G Interrupt Enable bit. '0' = No support Interrupt from Port G as Input; '1' = Support Interrupt from Port G as Input.	Yes	Yes	0
7	Port H Interrupt Enable bit. '0' = No support Interrupt from Port H as Input; '1' = Support Interrupt from Port H as Input.	Yes	Yes	0
8	Port I Interrupt Enable bit. '0' = No support Interrupt from Port I as Input; '1' = Support Interrupt from Port I as Input.	Yes	Yes	0
9	Port J Interrupt Enable bit. '0' = No support Interrupt from Port J as Input; '1' = Support Interrupt from Port J as Input.	Yes	Yes	0
10	Reserved	Yes	Yes	0
11	Reserved	Yes	Yes	0
12	Timer-A Enable bit. '0' = Timer-A disable; '1' = Timer-A Enable.	Yes	Yes	0
13	Timer-B Enable bit. '0' = Timer-B disable; '1' = Timer-B Enable.	Yes	Yes	0
14	Timer-C Enable bit. '0' = Timer-C disable; '1' = Timer-C Enable.	Yes	Yes	0
15	Timer-D Enable bit. '0' = Timer-D disable; '1' = Timer-D Enable.	Yes	Yes	0

Interrupt Source Register (0Ch & 0Dh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Interrupt Status in Port A. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port A as Input.	Yes	No	0
1	Interrupt Status in Port B. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port B as Input.	Yes	No	0
2	Interrupt Status in Port C. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port C as Input.	Yes	No	0
3	Interrupt Status in Port D. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port D as Input.	Yes	No	0
4	Interrupt Status in Port E. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port E as Input.	Yes	No	0
5	Interrupt Status in Port F. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port F as Input.	Yes	No	0
6	Interrupt Status in Port G. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port G as Input.	Yes	No	0
7	Interrupt Status in Port H. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port H as Input.	Yes	No	0
8	Interrupt Status in Port I. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port I as Input.	Yes	No	0
9	Interrupt Status in Port J. 0 = No Interrupt, 1 = Interrupt active. To clear this bit, must be wrote any data to Port J as Input.	Yes	No	0
10	Reserved	Yes	No	0
11	Reserved	Yes	No	0
12	Timer-A Interrupt status. 0 = No Interrupt from Timer-A, 1 = Timer-A Interrupt activ. To clear this bit, must be wrote to Timer-A register.	Yes	No	0
13	Timer-A Interrupt status. 0 = No Interrupt from Timer-B, 1 = Timer-B Interrupt activ. To clear this bit, must be wrote to Timer-B register.	Yes	No	0
14	Timer-A Interrupt status. 0 = No Interrupt from Timer-C, 1 = Timer-C Interrupt activ. To clear this bit, must be wrote to Timer-C register.	Yes	No	0
15	Timer-A Interrupt status. 0 = No Interrupt from Timer-D, 1 = Timer-D Interrupt activ. To clear this bit, must be wrote to Timer-D register.	Yes	No	0

Port BCD Mode (10h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Port A (8 bit). 0 = Input Mode, Output Mode disable	Yes	No	0
1	Port B (8 bit). 0 = Input Mode, 1 = Output Mode	Yes	Yes	0
2	Port C (8 bit). 0 = Input Mode, 1 = Output Mode	Yes	Yes	0
3	Port D (8 bit). 0 = Input Mode, 1 = Output Mode	Yes	Yes	0
7 : 4	Reserved	Yes	No	0

Port A Data (14h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port A	Yes	Only to clear Interrupt	0 / 1
1	Bit 2 of Port A	Yes	Only to clear Interrupt	0 / 1
2	Bit 3 of Port A	Yes	Only to clear Interrupt	0 / 1
3	Bit 4 of Port A	Yes	Only to clear Interrupt	0 / 1
4	Bit 5 of Port A	Yes	Only to clear Interrupt	0 / 1
5	Bit 6 of Port A	Yes	Only to clear Interrupt	0 / 1
6	Bit 7 of Port A	Yes	Only to clear Interrupt	0 / 1
7	Bit 8 of Port A	Yes	Only to clear Interrupt	0 / 1

Port B Data (18h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port B	Yes	Yes (only in Output Mode)	0 / 1
1	Bit 2 of Port B	Yes	Yes (only in Output Mode)	0 / 1
2	Bit 3 of Port B	Yes	Yes (only in Output Mode)	0 / 1
3	Bit 4 of Port B	Yes	Yes (only in Output Mode)	0 / 1
4	Bit 5 of Port B	Yes	Yes (only in Output Mode)	0 / 1
5	Bit 6 of Port B	Yes	Yes (only in Output Mode)	0 / 1
6	Bit 7 of Port B	Yes	Yes (only in Output Mode)	0 / 1
7	Bit 8 of Port B	Yes	Yes (only in Output Mode)	0 / 1

Port C Data (1Ch)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port C	Yes	Yes (only in Output Mode)	0 / 1
1	Bit 2 of Port C	Yes	Yes (only in Output Mode)	0 / 1
2	Bit 3 of Port C	Yes	Yes (only in Output Mode)	0 / 1
3	Bit 4 of Port C	Yes	Yes (only in Output Mode)	0 / 1
4	Bit 5 of Port C	Yes	Yes (only in Output Mode)	0 / 1

5	Bit 6 of Port C	Yes	Yes (only in Output Mode)	0 / 1
6	Bit 7 of Port C	Yes	Yes (only in Output Mode)	0 / 1
7	Bit 8 of Port C	Yes	Yes (only in Output Mode)	0 / 1

Port D Data (20h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port D	Yes	Yes (only in Output Mode)	0 / 1
1	Bit 2 of Port D	Yes	Yes (only in Output Mode)	0 / 1
2	Bit 3 of Port D	Yes	Yes (only in Output Mode)	0 / 1
3	Bit 4 of Port D	Yes	Yes (only in Output Mode)	0 / 1
4	Bit 5 of Port D	Yes	Yes (only in Output Mode)	0 / 1
5	Bit 6 of Port D	Yes	Yes (only in Output Mode)	0 / 1
6	Bit 7 of Port D	Yes	Yes (only in Output Mode)	0 / 1
7	Bit 8 of Port D	Yes	Yes (only in Output Mode)	0 / 1

Port EFGH Mode (24h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Port E (8 bit). 0 = Input Mode, 1 = Output Mode	Yes	Yes	0
1	Port F (8 bit). 0 = Input Mode, 1 = Output Mode	Yes	Yes	0
2	Port G (8 bit). 0 = Input Mode, 1 = Output Mode	Yes	Yes	0

3	Port H (8 bit). 0 = Input Mode, 1 = Output Mode	Yes	Yes	0
7 : 3	Reserved	Yes	No	0

Port E Data (28h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port E	Yes	Yes (only in Output Mode)	0 / 1
1	Bit 2 of Port E	Yes	Yes (only in Output Mode)	0 / 1
2	Bit 3 of Port E	Yes	Yes (only in Output Mode)	0 / 1
3	Bit 4 of Port E	Yes	Yes (only in Output Mode)	0 / 1
4	Bit 5 of Port E	Yes	Yes (only in Output Mode)	0 / 1
5	Bit 6 of Port E	Yes	Yes (only in Output Mode)	0 / 1
6	Bit 7 of Port E	Yes	Yes (only in Output Mode)	0 / 1
7	Bit 8 of Port E	Yes	Yes (only in Output Mode)	0 / 1

Port F Data (2Ch)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port F	Yes	Yes (only in Output Mode)	0 / 1
1	Bit 2 of Port F	Yes	Yes (only in Output Mode)	0 / 1
2	Bit 3 of Port F	Yes	Yes (only in Output Mode)	0 / 1
3	Bit 4 of Port F	Yes	Yes (only in Output Mode)	0 / 1
4	Bit 5 of Port F	Yes	Yes (only in Output Mode)	0 / 1
5	Bit 6 of Port F	Yes	Yes (only in Output Mode)	0 / 1
6	Bit 7 of Port F	Yes	Yes (only in Output Mode)	0 / 1

7	Bit 8 of Port F	Yes	Yes (only in Output Mode)	0 / 1
---	-----------------	-----	---------------------------	-------

Port G Data (30h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port G	Yes	Yes (only in Output Mode)	0 / 1
1	Bit 2 of Port G	Yes	Yes (only in Output Mode)	0 / 1
2	Bit 3 of Port G	Yes	Yes (only in Output Mode)	0 / 1
3	Bit 4 of Port G	Yes	Yes (only in Output Mode)	0 / 1
4	Bit 5 of Port G	Yes	Yes (only in Output Mode)	0 / 1
5	Bit 6 of Port G	Yes	Yes (only in Output Mode)	0 / 1
6	Bit 7 of Port G	Yes	Yes (only in Output Mode)	0 / 1
7	Bit 8 of Port G	Yes	Yes (only in Output Mode)	0 / 1

Port H Data (34h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port H	Yes	Yes (only in Output Mode)	0 / 1
1	Bit 2 of Port H	Yes	Yes (only in Output Mode)	0 / 1
2	Bit 3 of Port H	Yes	Yes (only in Output Mode)	0 / 1
3	Bit 4 of Port H	Yes	Yes (only in Output Mode)	0 / 1

4	Bit 5 of Port H	Yes	Yes (only in Output Mode)	0 / 1
5	Bit 6 of Port H	Yes	Yes (only in Output Mode)	0 / 1
6	Bit 7 of Port H	Yes	Yes (only in Output Mode)	0 / 1
7	Bit 8 of Port H	Yes	Yes (only in Output Mode)	0 / 1

Port IJ Mode (38h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Port I (8 bit). 0 = Input Mode, 1 = Output Mode	Yes	Yes	1
1	Port J (8 bit). 0 = Input Mode, 1 = Output Mode	Yes	Yes	0
2	Reserved	Yes	Yes	0
3	Reserved	Yes	Yes	0
7 : 4	Reserved	Yes	No	0

Port I Data (3Ch)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port I	Yes	Yes (only in Output Mode)	0 / 1
1	Bit 2 of Port I	Yes	Yes (only in Output Mode)	0 / 1
2	Bit 3 of Port I	Yes	Yes (only in Output Mode)	0 / 1
3	Bit 4 of Port I	Yes	Yes (only in Output Mode)	0 / 1
4	Bit 5 of Port I	Yes	Yes (only in Output Mode)	0 / 1
5	Bit 6 of Port I	Yes	Yes (only in Output Mode)	0 / 1
6	Bit 7 of Port I	Yes	Yes (only in Output Mode)	0 / 1
7	Bit 8 of Port I	Yes	Yes (only in Output Mode)	0 / 1

Port J Data (40h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0	Bit 1 of Port J	Yes	Yes (only in Output Mode)	0 / 1
1	Bit 2 of Port J	Yes	Yes (only in Output Mode)	0 / 1
2	Bit 3 of Port J	Yes	Yes (only in Output Mode)	0 / 1
3	Bit 4 of Port J	Yes	Yes (only in Output Mode)	0 / 1
4	Bit 5 of Port J	Yes	Yes (only in Output Mode)	0 / 1

5	Bit 6 of Port J	Yes	Yes (only in Output Mode)	0 / 1
6	Bit 7 of Port J	Yes	Yes (only in Output Mode)	0 / 1
7	Bit 8 of Port J	Yes	Yes (only in Output Mode)	0 / 1

Timer-A Register (4Ch & 4Dh)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0 - 15	16 bit Timer-A up to 65536 sec/ms/us . If this register is written, the Timer-A will count down and if "0" state is reached, it will generate an interrupt.	Yes	Yes, only if Timer-A is enabled	0

Timer-B Register (50h & 51h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0 - 15	16 bits Timer-B up to 1 to 65535 mS . If this register is written, the Timer-B will count down and if "0" state is reached, it will generate an interrupt.	Yes	Yes, only if Timer-B is enabled	0

Timer-C Register (54h & 55h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0 - 15	16 bits Timer-C up to 1 to 65535 mS . If this register is written, the Timer-C will count down and if "0" state is reached, it will generate an interrupt.	Yes	Yes, only if Timer-C is enabled	0

Timer-D Register (58h & 59h)

Bit Field	Description	Software Readable	Software Writable	Value after Reset
0 - 15	16 bits Timer-D up to 1 to 65535 ms . If this register is written, the Timer-D will count down and if "0" state is reached, it will generate an interrupt.	Yes	Yes, only if Timer-D is enabled	0

Timer Resolution Register (5Ch)

Acrosser default setting register value to be 55H (ms resolution) for using timer-A ~ timer-D.

Note: The resolutions for second & micro-second are reserved in further use.

De-bounce Time Register (60h ~6Ah)

These function only support after version 1.3. Setting de-bounce time for each I/O port.

De-bounce range in 1~256 ms. The value 00 means the de-bounce time is 1ms. Default setting is 10.

5 AGC DRIVER AND LIBRARY

This chapter describes driver, library and software utility for ACE-B5692. Acrosser provides drivers for the following operating systems: Windows 2000, Windows XP and Linux. We also support the test-program under Command Prompt Mode.

- Driver: The driver of ACE-B5692 supports general I/O, SRAM access, Timer, and Interrupt control in different OS (Windows/Linux). For convenient use of this driver, we have provided an AGC-Library that is described as follow.
- Library: For convenient use of ACE-B5692, we provide an AGC-Library to support general I/O, SRAM access, Timer, and Interrupt control in ACE-B5692. It must be collocated with the driver.
- Utility: For convenient use of ACE-B5692, we have provided some sample codes to control ACE-B5692 through the AGC-Library, we plus a DOS test application to easily understand ACE-B5692's hardware characteristics.

5.1 Windows AGC Driver and Libraries

- Operating System: Microsoft Windows XP SP2 / Microsoft Windows 2000 SP4
- Coding Environment: Microsoft Visual C++ .NET / Visual C++ 2005
- Installation setup: Vc2005 MFC Install
- File Description:
 - ARB5692.sys → AGC driver
 - AGC_5692_LIB.lib → Static Library of AGC driver
 - AGC_5692_LIB.dll → Dynamic Library of AGC driver

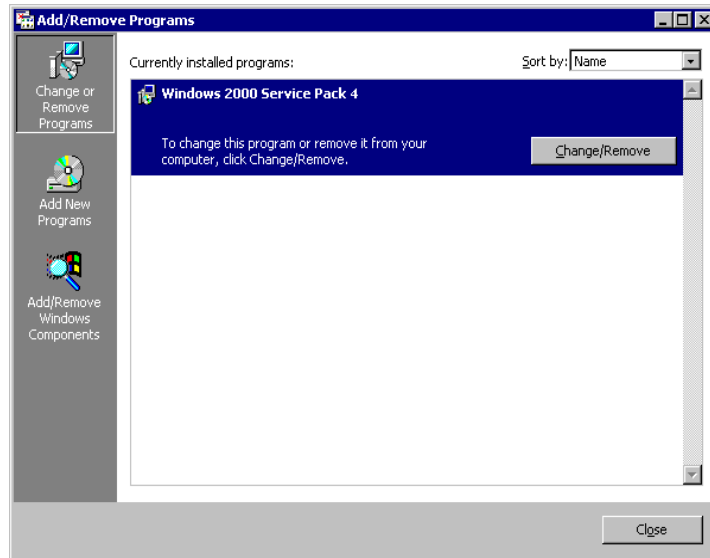
5.1.1 Windows AGC Driver Installation

Please install the Microsoft .NET Framework Version 2.0 Redistributable package and Microsoft Visual C++ 2005 Redistributable Package before you install the AGC and AGA driver. If you use Windows 2000, please upgrade it to Service Pack 4, and then install the following item 1, 2 and 3. You shall install all the following patches:

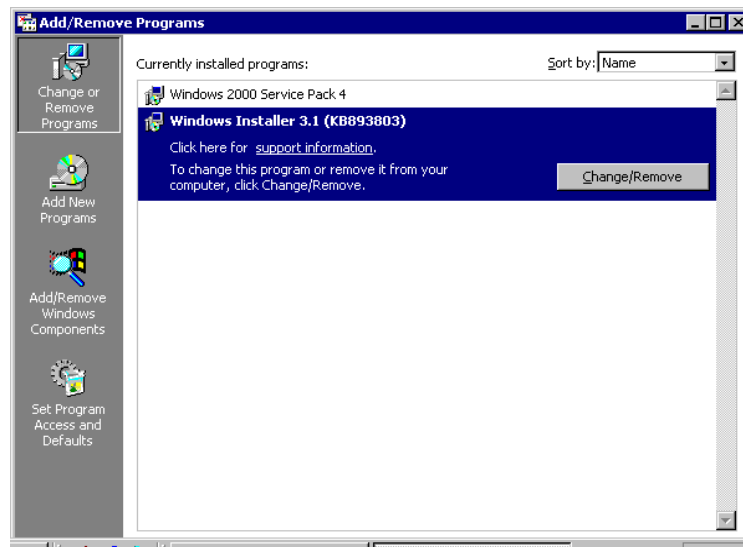
1. Windows Installer 3.1 Redistributable
2. Microsoft .NET Framework Version2.0 Redistributable package
3. Microsoft Visual C++ 2005 Redistributable Package

5.1.1.1 Install AGC driver in WIN2000

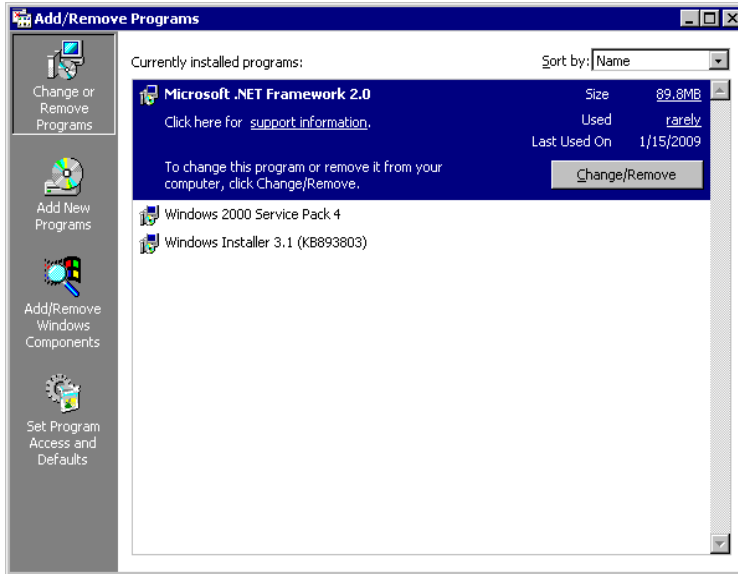
- a. Make sure you are already upgraded your Windows 2000 in Service Pack.



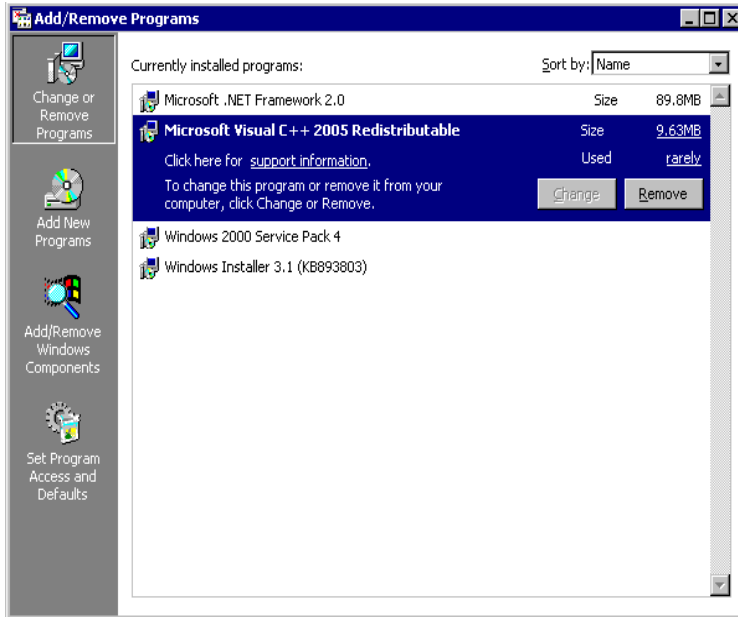
- b. Install the Windows Installer3.1.



c. Then install the Microsoft.NET Framework2.0.



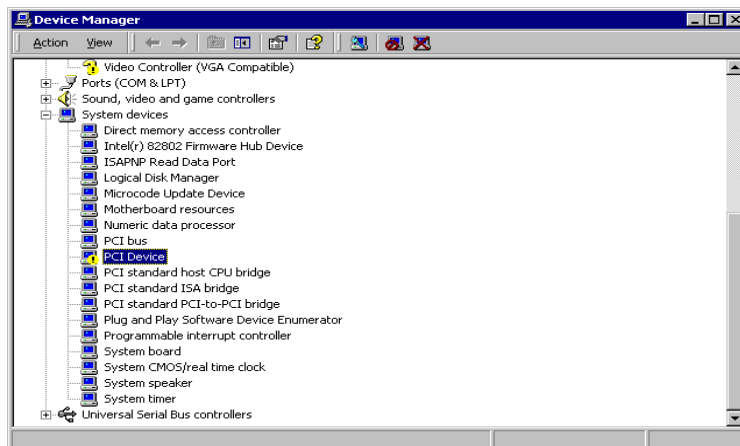
d. Install the Microsoft Visual C++ 2005 Redistributable.



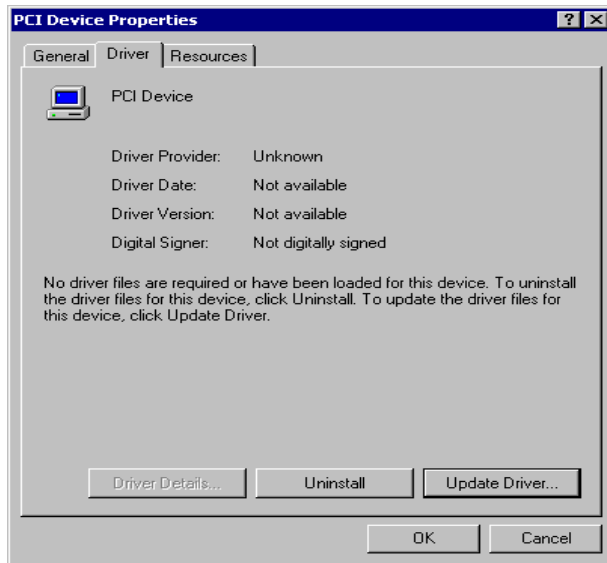
- e. Make sure install all patches according to above order. And install Acrosser driver with following steps: Click ARB5692.INF icon and click mouse right bottom to select install.



- f. In the Device Manager window, you will find a PCI Device with “!” mark.



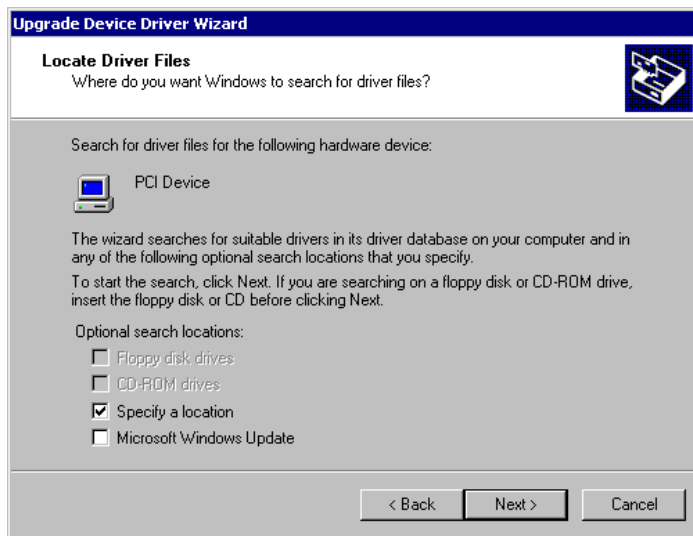
- g. Select that PCI Device and check its Properties. Select "Driver" and click Update Driver.



- h. Select "Search for a suitable driver for my device" and then click next.



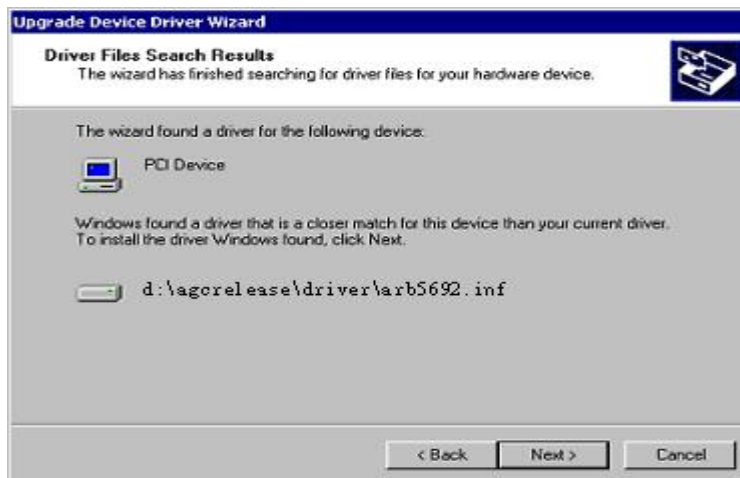
- i. Select "Specify a location" and click next.



- j. Select the folder where the driver located and then click "OK".



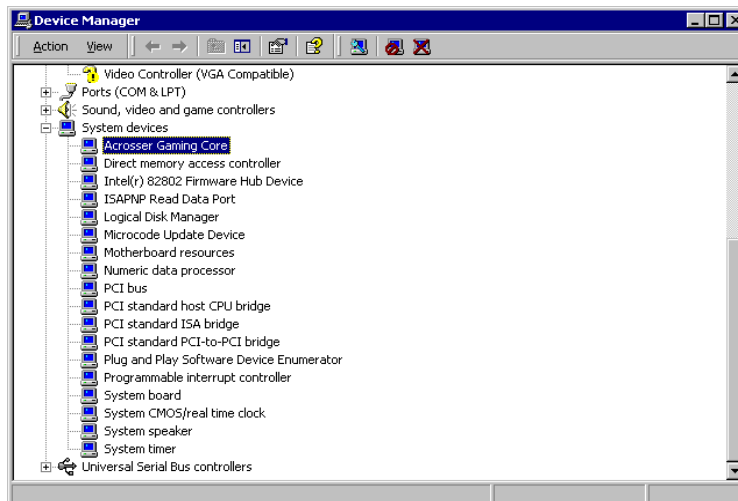
- k. Select the correct folder of arB5692.inf, you will find following message. Click "Next".



- l. Windows will start the installation of driver5692.



- m. Then you will find the Acrosser Gaming Core in Device Manager.

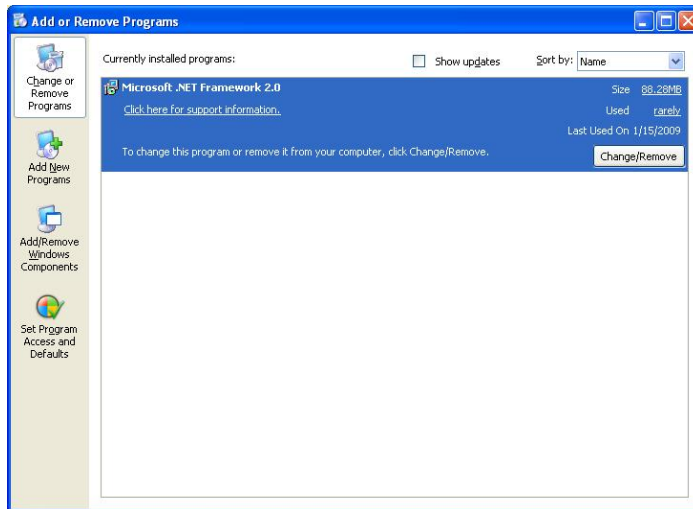


- n. Final, you can use "AGC_5692_AP.exe to find the I/O and memory devices.

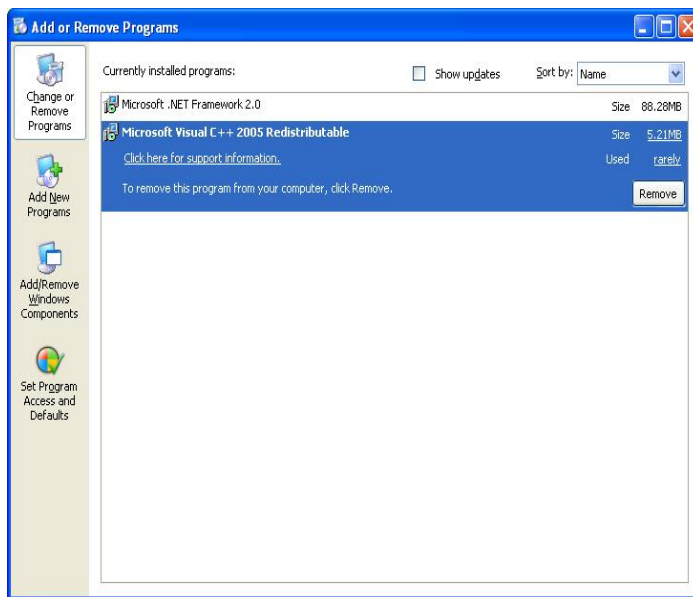
```
Acrosser Gaming I/O Board Test Program (Windows 2000/XP)
Open Card 1 ...
[I/O Base Address = DE00 ] [Memory Base Addr = FDB00000 ]
[Swan Mode = 0]
===== MENU =====
<0> Exit
<1> Set Port Type
<2> Set Port Data
<3> Get Port Data
<4> Switch Memory Mode
<5> Write Memory For Byte
<6> Read Memory For Byte
<7> Set InterruptEnable Register
<8> Get InterruptEnable Register
<9> Set Timer Register
<10> Get Timer Register
<11> Get DipSwitch1 Ualue
<12> Get DipSwitch2 Ualue
<13> Memory Swan Write test
<14> Memory Swan Read test
<15> Set Port Debounce Time
<16> Get Port Debounce Time
<17> Get Time Resolution Switch
<18> Set Time Resolution Switch
<19> Get Interrupt Buffer Count
<20> Get Interrupt Buffer data
<21> Clear Interrupt Buffer data
<22> Get Interruptport Timer Buffer Count
<23> Get Interruptport Timer Buffer data
<24> Clear Interruptport Timer Buffer data
>>
```

5.1.1.2 Install AGC driver in Windows XP

- a. First of all, please install the Microsoft.NET Framework2.0.



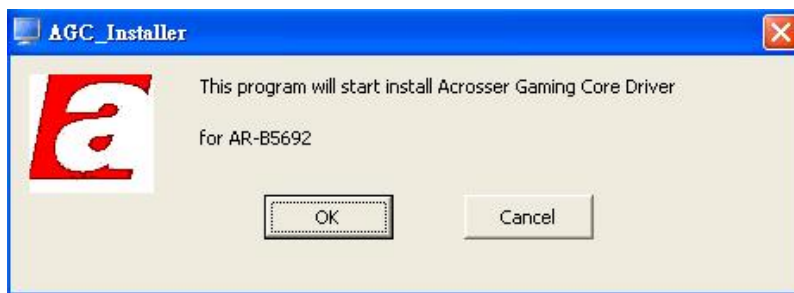
- b. Then install the Microsoft Visual C++ 2005 Redistributable.



- c. Double click the **setup.exe**.



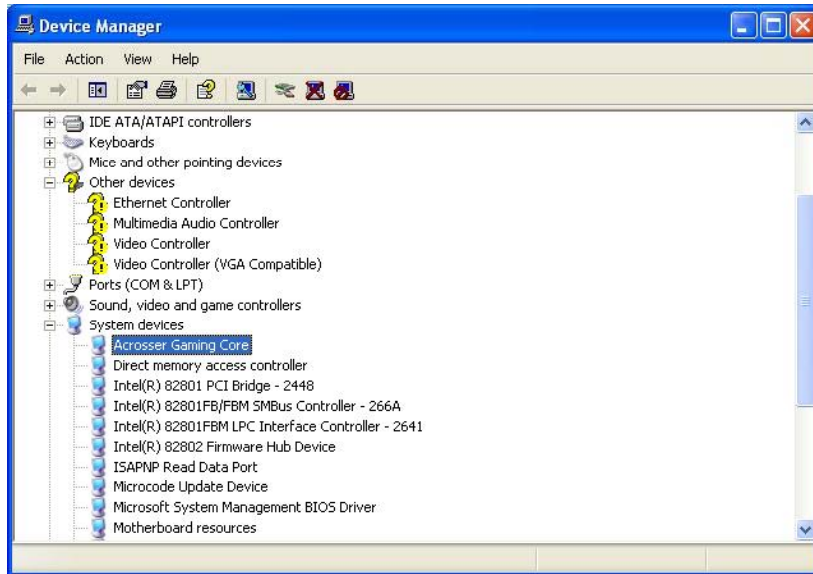
- d. An installer will show up. Please click “OK” to continue the installation.



- e. After the installation the “Driver Install successful” will show up. Please click “OK”.



f. You will find Acrosser Gaming Core in Device Manager.



g. Final, you can use “AGC_5692_AP.exe to find the I/O and memory devices.

```

Acrosser Gaming I/O Board Test Program (Windows 2000/XP)
Open Card 1 ...
I/O Base Address = DE00 1 [Memory Base Addr = FDB00000 1
ISram Mode = 0]
===== MENU =====
<0> Exit
<1> Set Port Type
<2> Set Port Data
<3> Get Port Data
<4> Switch Memory Mode
<5> Write Memory For Byte
<6> Read Memory For Byte
<7> Set InterruptEnable Register
<8> Get InterruptEnable Register
<9> Set Timer Register
<10> Get Timer Register
<11> Get DipSwitch1 Value
<12> Get DipSwitch2 Value
<13> Memory Sram Write test
<14> Memory Sram Read test
<15> Set Port Debounce Time
<16> Get Port Debounce Time
<17> Get Time Resolution Switch
<18> Set Time Resolution Switch
<19> Get Interrupt Buffer Count
<20> Get Interrupt Buffer data
<21> Clear Interrupt Buffer data
<22> Get Interruptport Timer Buffer Count
<23> Get Interruptport Timer Buffer data
<24> Clear Interruptport Timer Buffer data
>>
    
```

5.1.1.3 Windows AGC-API Library

5.1.1.3.1 Register Card

@Description

This function is used to register the ACE-B5692. ACE-B5692 has to be registered by this function before other functions are called.

@Syntax

VC/VC++

```
unsigned int WXP_RegisterCard(short *CardNum)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.2 Release Card

@Description

This function is used to release the registered ACE-B5692.

@Syntax

VC/VC++

```
unsigned int WXP_ReleaseCard(short CardNum)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.3 GetIOBaseAddr

@Description

This function is used to get the I/O base address of ACE-B5692.

@Syntax

VC/VC++

```
unsigned int WXP_GetIOBaseAddr(short CardNum, unsigned int*  
IOBaseAddr)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

IoBaseAddr: the I/O base address of ACE-B5692.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.4 GetMemBaseAddr**@Description**

This function is used to get the memory base address of ACE-B5692.

@Syntax

VC/VC++

```
unsigned int WXP_GetMemBaseAddr(short CardNum, unsigned int* MemBaseAddr)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized

MemBaseAddr: The memory base address of ACE-B5692

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.5 SetPortType**@Description**

This function is used to set the port type as INPUT or OUTPUT.

@Syntax

VC/VC++

```
unsigned int WXP_SetPortType(short CardNum, short PortNum, unsigned char Type)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

PortNum: Port number from "A" to "L".(0 ~ 11)

Type: 0 = "Input" or 1 = "Output".

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.6 ReadPort**@Description**

This function is used to read the data of each port.

@Syntax

VC/VC++

```
unsigned int WXP_ReadPort(short CardNum, short PortNum, unsigned char* Value)
```

@Argument

CardNum: The number of ACE-B5692 card initialized.

PortNum: Port number from "A" to "L".(0~11)

Value: The data of the port.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.7 WritePort**@Description**

This function is used to write the card of each port.

@Syntax

VC/VC++

unsigned int WXP_WritePort(short CardNum , short PortNum,
unsigned char Value)

@Argument

CardNum: The number of ACE-B5692 card initialized.

PortNum: Port number from "A" to "L".(0~11)

Value: The data will be written to the output port.

@Return Code

Please refer to Chapter 5.3 Error Code

5.1.1.3.8 SetCallbackFunTable**@Description**

Set the call back function table to ACE-B5692 which number is the CardNo. When then interrupt arises, the call back function table set to this CardNo will be involved.

@Syntax

VC/VC++

unsigned int WXP_SetCallbackFunTable(short CardNum ,
_AGC_Callback_FunTab_t *)

@Argument

i16 CardNo: The card number of ACE-B5692 card initialized. Range from 1~5.

AGC_Callback_FunTab_t* funtab: Call back functions table.

@Return Code

No Chapter 5.3 Error Code.

5.1.1.3.9 SetInterruptEnableReg**@Description**

This function is used to set interrupt enable register.

@Syntax

VC/VC++

unsigned int WXP_SetInterruptEnableReg(short CardNum , unsigned short IntEnValue);

@Argument

CardNum: The card number of ACE-B5692 card initialized.

IntrEnValue: Interrupt enable value.

FORMAT:

HI BYTE											LO BYTE				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD	TC	TB	TA	PL	PK	PJ	PI	PH	PG	PF	PE	PD	PC	PB	PA
S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

S = 1:Enable 0: Disable, TX =TIMER, PX=PORT

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.10 GetInterruptEnableReg

@Description

This function is used to get interrupt enable register.

@Syntax

VC/VC++

unsigned int LNX_GetInterruptEnableReg(short CardNum , unsigned short *IntEnValue)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

IntrEnValue: Interrupt enable value.

FORMAT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD	TC	TB	TA	TL	PK	PJ	PI	PH	PG	PF	PE	PD	PC	PB	PA

TX =TIMER , PX=PORT

***NOTE: ACE-B5692 just get PA~PL , TA~TD back**

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.11 GetPortInterruptCounter

@Description

The interrupts from port A~L would be saved in driver's buffer which max size is 2048. This function is used to get how many interrupt

events has been saved in buffer.

@Syntax

VC/VC++

unsigned int WXP_GetInterruptCounter (short CardNum , unsigned short* Count)

@Argument

CardNum: The card number of ACE-B5692 card initialized. Range from 1~5.

Count: The number of interrupt event which has been saved in the buffer.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.12 GetPortInterruptBuffer**@Description**

The interrupts from port A~L would be saved in driver's buffer which max size is 2048. This function is used to get one event from the buffer.

@Syntax

VC/VC++

unsigned int WXP_GetInterruptBuffer(short CardNum , short* PortNum, unsigned char* Value, unsigned short* RemainCount)

@Argument

CardNum: The card number of ACE-B5692 card initialized. Range from 1~5.

PortNum: The port number that the event belong.

Value: The value read from port.

RemainCount: The number of interrupt event which remain in the buffer.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.13 SetTimerReg**@Description**

This function is used to set timer register.

@Syntax

VC/VC++

unsigned int WXP_SetTimerReg(short CardNum , short TimerNum,
unsigned short TimerValue)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

TimerNum: Timer number.

TimerValue: Timer Value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.14 GetTimerReg**@Description**

This function is used to get timer register.

@Syntax

VC/VC++

unsigned int WXP_GetTimerReg (short CardNum , short TimerNum,
unsigned short *TimerValue)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

TimerNum: Timer Number.

TimerValue: Timer Value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.15 SetTimerResolution**@Description**

There are four timers on ACE-B5692, each timer uses two bits to set their resolution. This function selects timer resolution as micro-second(μ s), millisecond(ms) and second(sec).

@Syntax

VC/VC++

unsigned int WXP_SetTimerResolution(short CardNum , short
TimerNum, unsigned char Resolution)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

TimerNum: Timer number. Range from (0~3).

Resolution: The resolution selection. From 0~2(00 = sec, 01 = ms, 10 = μ s).

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.16 GetTimerResolution**@Description**

There are four timers on ACE-B5692, each timer uses two bits to set their resolution. This function is used to get current timer resolution.

@Syntax

VC/VC++

unsigned int WXP_GetTimerResolution(short CardNum , short TimerNum, unsigned char* Resolution)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

TimerNum: Timer number. Range from (0~3).

Resolution: The resolution selection. From 0~2(00 = sec, 01 = ms, 10 = μ s).

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.17 SetDebounce**@Description**

This function is used to set the debounce time for a port.

@Syntax

VC/VC++

unsigned int WXP_SetDebounce (short CardNum , short PortNum, unsigned char Value)

@Argument

CardNum: The card number of ACE-B5692rd initialized.

PortNum: Port number from "A" to "L", so the range is from 0~11.

Value: The de-bounce time 0~255.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.18 GetDebounce**@Description**

Get the de-bounce time form a port.

@Syntax

VC/VC++

unsigned int WXP_GetDebounce(short CardNum , short PortNum, unsigned char* Value)

@Argument

i16 CardNo: The card number of ACE-B5692 initialized. Range from 1~5.

i16 PortNum: Port number from "A" to "L", so the range is from 0~11.

byte Value: The de-bounce time 0~255.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.19 SwitchMemSram**@Description**

This function is used to switch memory Mode.

@Syntax

VC/VC++

unsigned int WXP_SwitchMemBank (CardNum, ModeNum)

@Argument

CardNum: The number of ACE-B5692 card initialized.

ModeNum: Mode Number.(0 or 1), 0= 1024kByte Mode , 1 = Rewrite 512kByte Mode.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.20 WriteMemByte**@Description**

This function is used to write memory type.

@Syntax

VC/VC++

unsigned int WXP_WriteMemByte(short CardNum , unsigned int Offset, unsigned char Value)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

Offset: Memory offset.

Value: Memory Value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.21 ReadMemByte**@Description**

This function is used to read memory byte.

@Syntax

VC/VC++

unsigned int WXP_ReadMemByte(short CardNum , unsigned int Offset, unsigned char* Value)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

Offset: Memory offset.

Value: Memory value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.22 ReadMemBlock**@Description**

This function is used to read memory Block.

@Syntax

VC/VC++

unsigned int WXP_ReadMemBlock(short CardNum , unsigned int Offset, unsigned char *Value, unsigned int Size)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

Offset: Memory offset.

Value: Memory value array.

Size: Memory Size.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.23 WriteMemBlock**@Description**

This function is used to write memory Block.

@Syntax

VC/VC++

unsigned int WXP_WriteMemBlock(short CardNum , unsigned int Offset, unsigned char *Value, unsigned int Size)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

MemOffset: Memory offset.

MemValue: Memory value array.

Size: Memory Size.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.24 GetDipSwitch1**@Description**

This function is used to get dip switch 1 state value.

@Syntax

VC/VC++

unsigned int WXP_GetDipSwitch1 (CardNum, &DipSwitchValue)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

DipSwitchValue: Dip Switch1 Value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3.25 GetDipSwitch2**@Description**

This function is used to get dip switch 2 state value.

@Syntax

VC/VC++

unsigned int WXP_GetDipSwitch2 (CardNum, &DipSwitchValue)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

DipSwitchValue: Dip Switch2 Value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3. 26 GetTimerInterruptCounter**@Description**

The interrupts from Timer A~D would be saved in driver's buffer which max size is 1024. This function is used to get how many interrupts.

@Syntax

C/C++

unsigned int WXP_GetInterruptTimerCounter(short CardNum ,
unsigned short* Count)

@Argument

CardNum: The card number of ACE-B5692 card initialized. Range from 1~5.

Count: The number of interrupt.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.1.1.3. 27 GetTimerInterruptBuffer**@Description**

The interrupts from Timer A~D would be saved in driver's buffer which max size is 1024.

@Syntax

VC/VC++

unsigned int WXP_GetInterruptTimerBuffer (short CardNum , short* TimerNum, , unsigned short* RemainCount)

@Argument

CardNum: The card number of ACE-B5692 card initialized. Range from 1~5.

TimerNum: The Timer number .

RemainCount: The number of interrupt.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2 Linux AGC Driver and Libraries

Operating System: Kernel 2.6.27.5-117.fc10.i686

Coding Environment: Gcc

- File Discription:
 - arb5692.ko → AGC driver
 - libAGC_5692_LIB.a → Static Library of AGC driver
 - libAGC_5692_LIB.so → Dynamic Library of AGC driver

5.2.1 Linux AGC Driver Load

- a. Check if AGC driver is existed.

```
[root@localhost Driver]# ls
arb5692.ko  scripts
[root@localhost Driver]#
```

- b. Check if load/unload script is existed.

```
[root@localhost scripts]# ls
modld modul
[root@localhost scripts]#
```

- c. Run mold to load module and check if AGC driver module loads.

```
[root@localhost scripts]# lsmod
Module              Size  Used by
arb5692             12588  0
i915                 53380  2
drm                 158260  3 i915
sco                  12932  2
bridge              43668  0
```

- d. Then you can use "AGC_5692_AP_s " to access the I/O and memory.

```
Acrosser Gaming I/O Board Test Program Fedora core
Open Card 1 ... 0 [IO Base Address = DE00 ] [Memory Base Addr = FDB00000 ]
[Sram Mode = 0]
===== MENU =====
(0) Exit
(1) Set Port Type
(2) Set Port Data
(3) Get Port Data
(4) Switch Memory Write Mode
(5) Write Memory For Byte
(6) Read Memory For Byte
(7) Set InterruptEnable Register
(8) Get InterruptEnable Register
(9) Set Timer Register
(10) Get Timer Register
(11) Get DipSwitch1 Value
(12) Get DipSwitch2 Value
(13) Memory block Write test
(14) Memory block Read test
(15) Set Port Debounce Time
(16) Get Port Debounce Time
(17) Get Time Resolution Switch
(18) Set Time Resolution Switch
(19) Get Interrupt Port Buffer Count
(20) Get Interrupt Port Buffer data
(21) Clear Interrupt Port Buffer data
(22) Get Interruptport Timer Buffer Count
(23) Get Interruptport Timer Buffer data
(24) Clear Interruptport Timer Buffer data
>>
```

*AGC_5692_AP_s uses static library.

*AGC_5692_AP_d uses dynamic library.

*If you run AGC_5692_AP_d, please make sure libAGC_5692_LIB.so is located in your dynamic link library folders.

5.2.2 Linux AGC-API Library

5.2.2.1 RegisterCard

@Description

This function is used to register of the ACE-B5692. ACE-B5692 has to be registered by this function before other functions are called.

@Syntax

C/C++

```
unsigned int LNX_RegisterCard(short *CardNum)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.2 ReleaseCard

@Description

This function is used to release the registered ACE-B5692.

@Syntax

C/C++

unsigned int LNX_ReleaseCard(short CardNum)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.3 GetIOBaseAddr

@Description

This function is used to get the I/O base address of ACE-B5692.

@Syntax

C/C++

unsigned int LNX_GetIOBaseAddr(short CardNum,unsigned int* IOBaseAddr)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

IoBaseAddr: the I/O base address of ACE-B5692.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.4 GetMemBaseAddr

@Description

This function is used to get the memory base address of ACE-B5692.

@Syntax

C/C++

unsigned int LNX_GetMemBaseAddr(short CardNum,unsigned int* MemBaseAddr)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

MemBaseAddr: The memory base address of ACE-B5692.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.5 SetPortType

@Description

This function is used to set the port type as INPUT or OUTPUT.

@Syntax

C/C++

unsigned int LNX_SetPortType(short CardNum , short PortNum,
unsigned char Type)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

PortNum: Port number from "A" to "L".(0 ~ 11)

Type: 0 = "Input" or 1 = "Output".

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.6 ReadPort

@Description

This function is used to read the data of each port.

@Syntax

C/C++

unsigned int LNX_ReadPort(short CardNum , short PortNum,
unsigned char* Value)

@Argument

CardNum: The number of ACE-B5692 card initialized.

PortNum: Port number from "A" to "L".(0~11)

Value: The data of the port.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.7 WritePort

@Description

This function is used to write the card of each port.

@Syntax

C/C++

unsigned int LNX_WritePort(short CardNum , short PortNum,
unsigned char Value)

@Argument

CardNum: The number of ACE-B5692 card initialized.

PortNum: Port number from "A" to "L".(0~11)

Value: The data will be written to the output port.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.8 SetCallbackFunTable

@Description

Set the call back function table to ACE-B5692 which number is the CardNo. When then interrupt arises, the call back function table set to this CardNo will be involved.

@Syntax

C/C++

```
unsigned int LNX_SetCallbackFunTable(short CardNum ,
_AGCCallbackFunTab_t*)
```

@Argument

i16 CardNo: The card number of ACE-B5692 card initialized. Range from 1~5.

AGCCallbackFunTab_t* funtab: Call back functions table.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.9 SetInterruptEnableReg

@Description

This function is used to set interrupt enable register.

@Syntax

C/C++

```
unsigned int LNX_SetInterruptEnableReg(short CardNum , unsigned
short IntEnValue)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

IntrEnValue: Interrupt enable value.

FORMAT:

HI BYTE											LO BYTE				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD	TC	TB	TA	PL	PK	PJ	PI	PH	PG	PF	PE	PD	PC	PB	PA
S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

S = SET , TX =TIMER , PX=PORT

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.10 GetInterruptEnableReg

@Description

This function is used to get interrupt enable register.

@Syntax

C/C++

```
unsigned int LNX_GetInterruptEnableReg(short CardNum , unsigned short *IntEnValue)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

IntrEnValue: Interrupt enable value.

FORMAT :

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD	TC	TB	TA	TL	PK	PJ	PI	PH	PG	PF	PE	PD	PC	PB	PA

TX =TIMER , PX=PORT

***NOTE: ACE-B5692 just get PA~PL , TA~TD back.**

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.11 GetPortInterruptportCounter

@Description

The interrupts from port A~L would be saved in driver's buffer which maximum size is 2048. This function is used to get how many interrupt events has been saved in buffer.

@Syntax

C/C++

```
unsigned int LNX_GetInterruptportCounter(short CardNum , unsigned short* Count)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized. Range from 1~5.

Count: The number of interrupt event which has been saved in the

buffer.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.12 GetPortInterruptportBuffer**@Description**

The interrupts from port A~L would be saved in driver's buffer which maximum size is 2048. This function is used to get one event from the buffer.

@Syntax

C/C++

```
unsigned int LNX_GetInterruptportBuffer(short CardNum , short*  
PortNum, unsigned char* Value, unsigned short* RemainCount)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized. Range from 1~5.

PortNum: The port number that the event belong.

Value: The value read from port.

RemainCount: The number of interrupt event which remain in the buffer.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.13 SetTimerReg**@Description**

This function is used to set timer register.

@Syntax

C/C++

```
unsigned int LNX_SetTimerReg(short CardNum , short TimerNum,  
unsigned short TimerValue)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

TimerNum: Timer number.

TimerValue: Timer Value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.14 GetTimerReg

@Description

This function is used to get timer register.

@Syntax

C/C++

```
unsigned int LNX_GetTimerReg(short CardNum , short TimerNum,  
unsigned short *TimerValue)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

TimerNum: Timer Number.

TimerValue: Timer Value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.15 SetTimerResolution**@Description**

There are four timers on ACE-B5692. Each timer uses two bits to set their resolution. This function select timer resolution as micro-second(μ s), millisecond(ms) and second(sec).

@Syntax

C/C++

```
unsigned int LNX_SetTimerResolution(short CardNum , short  
TimerNum, unsigned char Resolution)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized.

TimerNum: Timer number. Range from (0~3).

Resolution: The resolution selection. From 0~2(00 = sec, 01 = ms, 10 = μ s).

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.16 GetTimerResolution**@Description**

There are four timers on ACE-B5692. Each timer uses two bits to set their resolution. This function is used to get current timer resolution.

@Syntax

C/C++

```
unsigned int LNX_GetTimerResolution(short CardNum , short  
TimerNum, unsigned char* Resolution)
```


@Argument

CardNum: The card number of ACE-B5692 card initialized.

TimerNum: Timer number. Range from (0~3).

Resolution: The resolution selection. From 0~2(00 = sec, 01 = ms, 10 = μ s).

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.17 SetDebounce**@Description**

This function is used to set the debounce time for a port.

@Syntax

C/C++

unsigned int LNX_SetDebounce(short CardNum , short PortNum, unsigned char Value)

@Argument

CardNum: The card number of ACE-B5692rd initialized.

PortNum: Port number from "A" to "L", so the range is from 0~11.

Value: The de-bounce time 0~255.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.18 GetDebounce**@Description**

Get the de-bounce time form a port.

@Syntax

C/C++

unsigned int LNX_GetDebounce(short CardNum , short PortNum, unsigned char* Value)

@Argument

i16 CardNo: The card number of ACE-B5692 initialized. Range from 1~5.

i16 PortNum: Port number from "A" to "L", so the range is from 0~11.

byte Value: The de-bounce time 0~255.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.19 SwitchMemSram**@Description**

This function is used to switch memory Mode.

@Syntax

C/C++

unsigned int LNX_SwitchMemBank (CardNum, ModeNum)

@Argument

CardNum: The number of ACE-B5692 card initialized.

ModeNum: Mode Number.(0 or 1),0= 1024kByte Mode , 1 = Rewrite

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.20 WriteMemByte**@Description**

This function is used to write memory type.

@Syntax

C/C++

unsigned int LNX_WriteMemByte(short CardNum , unsigned int
Offset, unsigned char Value)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

Offset: Memory offset.

Value: Memory Value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.21 ReadMemByte**@Description**

This function is used to read memory byte.

@Syntax

C/C++

unsigned int LNX_ReadMemByte(short CardNum , unsigned int
Offset, unsigned char* Value)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

Offset: Memory offset.

Value: Memory value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.22 ReadMemBlock**@Description**

This function is used to read memory Block.

@Syntax

C/C++

unsigned int LNX_ReadMemBlock(short CardNum , unsigned int Offset, unsigned char *Value, unsigned int Size)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

Offset: Memory offset.

Value: Memory value array.

Size: Memory Size.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.23 WriteMemBlock**@Description**

This function is used to write memory Block.

@Syntax

C/C++

unsigned int LNX_WriteMemBlock(short CardNum , unsigned int Offset, unsigned char *Value, unsigned int Size)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

MemOffset: Memory offset.

MemValue: Memory value array.

Size: Memory Size.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.24 GetDipSwitch1**@Description**

This function is used to get dip switch1 state value.

@Syntax

C/C++

unsigned int LNX_GetDipSwitch1 (CardNum, &DipSwitchValue)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

DipSwitchValue: Dip Switch1 Value.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2.25 GetDipSwitch2

@Description

This function is used to get dip switch2 state value.

@Syntax

C/C++

unsigned int LNX_GetDipSwitch2 (CardNum, &DipSwitchValue)

@Argument

CardNum: The card number of ACE-B5692 card initialized.

DipSwitchValue: Dip Switch2 Value

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2. 26 GetTimerInterruptCounter

@Description

The interrupts from Timer A~D would be saved in driver's buffer which maximum size is 1024. This function is used to get how many interrupt.

@Syntax

C/C++

unsigned int LNX_GetTimerInterruptCounter(short CardNum ,
unsigned short* Count)

@Argument

CardNum: The card number of ACE-B5692 card initialized. Range from 1~5.

Count: The number of interrupt.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.2.2. 27 GetTimerInterruptBuffer**@Description**

The interrupts from Timer A~D would be saved in driver's buffer which maximum size is 1024.

@Syntax

C/C++

```
unsigned int LNX_GetTimerInterruptBuffer(short CardNum , short*  
TimerNum, , unsigned short* RemainCount)
```

@Argument

CardNum: The card number of ACE-B5692 card initialized. Range from 1~5.

TimerNum: The Timer number.

RemainCount: The number of interrupt.

@Return Code

Please refer to Chapter 5.3 Error Code.

5.3 Error Code Notes

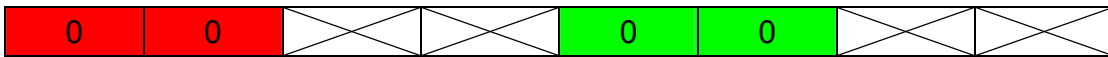
ERROR_MASK_API

0xFF000000 API Mask FFxxxxxx, from 0~255

ERROR_MASK_GEN

0x0000FF00 General error code mask, Mask xxxxFFxx, range(0~255)

For return error format



ERROR_MASK_API

- 0x20 Error code for API RegisterCard
- 0x21 Error code for API ReleaseCard
- 0x22 Error code for API GetMemBaseAddr
- 0x23 Error code for API GetIOBaseAddr
- 0x24 Error code for API ReadPort
- 0x25 Error code for API WritePort
- 0x26 Error code for API SetPortType
- 0x27 Error code for API ReadMemByte
- 0x28 Error code for API WriteMemByte
- 0x29 Error code for API SwitchMemBank
- 0x2A Error code for API GetInterruptEnableReg
- 0x2B Error code for API SetInterruptEnableReg
- 0x2C Error code for API GetTimerReg
- 0x2D Error code for API SetTimerReg
- 0x2E Error code for API SetCallbackFunTable
- 0x2F Error code for API ReadMemBlock
- 0x30 Error code for API WriteMemBlock
- 0x31 Error code for API GetTimerResolution
- 0x32 Error code for API SetTimerResolution
- 0x33 Error code for API GetInterruptBuffer
- 0x34 Error code for API SetDebounce
- 0x35 Error code for API GetDebounce
- 0x36 Error code for API HardMeter
- 0xFF Error code for API UNKNOWN

ERROR_MASK_GEN

- 0x0C AGC device open error
- 0x0D AGC device number error

0x0E HardMeter Device No Responce
0xFF Others unknow errors

6 AGA DRIVER AND LIBRARY

This chapter describes driver, library and software utility for ACE-B5692. Acrosser provides drivers for the following operating systems: Windows XP and Linux. We also support the test program under command prompt mode.

6.1 Windows AGA Driver and Libraries

- Operating System: Microsoft Windows XP SP2 / Microsoft Windows 2000 SP4
- Coding Environment: Microsoft Visual C++ .NET Visual C++ 2005
- File Discription:
 - AGA_5692_LIB.lib → Static Library of AGA driver
 - AGA_5696_LIB.dll → Dynamic Library of AGA driver

6.1.1 Windows AGA-API Library

6.1.1.1 AGA_InitLib

@Description

Set com port.

@Syntax

VC/VC++

unsigned int WXP_AGA_InitLib (unsigned short usCOM_Num)

@Argument

usCOM_Num: This is the serial port number used for communicating with ACE-B5692 in your computer. For example, if you use the COM1 to communicate with ACE-B5692, this value should be "1".

@Return Code

Please refer to Chapter 6.3 Error Code

6.1.1.2 AGA_DeInitLib

@Description

This function is used to release the DeInitLib ACE-B5692.

@Syntax

VC/VC++

unsigned int WXP_AGA_DeInitLib()

@Argument

NONE

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.3 AGA_Get_RTC**@Description**

Set up AGA RTC Time

@Syntax

VC/VC++

unsigned int WXP_AGA_Get_RTC(struct tm* pTm)

@Argument

PTm: This is the pointer of the structure for returned RTC data.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.4 AGA_Set_RTC

This API is used to read the current RTC data stored in real time clock.

It uses standard structure defined in <time.h>.

@Syntax

VC/VC++

unsigned int WXP_AGA_Set_RTC(unsigned short Type,struct tm*
pTm)**@Argument**

Type: user mode or system mode user =1 , system = 2.

Struct tm: Time for pTm Time format.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.5 AGA_Get_IntrLog**@Description**

Read Record Lo or Hi form the IntrLog Record.

@Syntax

VC/VC++

unsigned int WXP_AGA_Get_IntrLog(unsigned char
*LogBuf,unsigned short LogLen,unsigned short *ReadLen)**@Argument**

LogBuf : Read log buffer

LogLen: Read record times
ReadLen : Return data times

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.6 AGA_Cls_IntrLog**@Description**

Clean Record Lo or Hi to the IntrLog Record.

@Syntax

VC/VC++
unsigned int WXP_AGA_Cls_IntrLog()

@Argument

no.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.7 AGA_Get_BattLog1**@Description**

Read AGA Record from the battery log Record

@Syntax

VC/VC++
unsigned int WXP_AGA_Get_BattLog1(unsigned char
*LogBuf,unsigned short LogLen,unsigned short *ReadLen)

@Argument

LogBuf : Read log buffer.
LogLen: Read record times.
ReadLen : Return data times.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.8 AGA_Cls_BattLog1**@Description**

Clean AGA Record to the battery log Record.

@Syntax

VC/VC++
unsigned int WXP_AGA_Cls_BattLog1()

@Argument

no.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.9 AGA_Get_BattLog2**@Description**

Read AGA Record from the battery log Record.

@Syntax

VC/VC++

unsigned int WXP_AGA_Get_BattLog1(unsigned char
*LogBuf,unsigned short LogLen,unsigned short *ReadLen)

@Argument

LogBuf : Read log buffer.

LogLen: Read record times.

ReadLen : Return data times.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.10 AGA_Cls_BattLog2**@Description**

Clean AGA Record to the battery log Record.

@Syntax

VC/VC++

unsigned int WXP_AGA_Cls_BattLog1()

@Argument

no.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.11 AGA_GET_AGAVER**@Description**

Read AGA Version.

@Syntax

VC/VC++

unsigned int WXP_AGA_GET_AGAVER(unsigned char *ver)

@Argument

ver : AGA Version.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.12 AGA_Set_Protect-U**@Description**

Your application software can sets the data (manufacture code and serial number) into Protect-U by calling the ACE-B5692 Protect-U Library API “ Lib_Set_PTU ”, then this API will return the count value in Protect-U.

@Syntax

VC/VC++

```
unsigned int WXP_AGA_Set_Protect-U(unsigned char *Manu_Code,  
unsigned char *Ser_Num, unsigned short *PTU_Counter)
```

@Argument

Manu_Code: This is the 16-character hexadecimal “manufacture code” you are going to program into the Protect-U chip.

Ser_Num: This is the 7-character hexadecimal “serial number” you are going to program into the Protect-U chip.

PTU_Counter: Return counter.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.13 AGA_Check_Protect-U**@Description**

This API is used to validate the consistency of Protect-U. It will return an error code and Protect-U counter value. Please keep the counter value for the next **time** check. The API needs input of manufacture code, serial number, previous counter number and allowed counter errors. The API will **uses** the input data to require a set of encrypted data from Protect-U and decrypt an encrypted data. The decrypted data contains a new counter value which is normally the previous counter value plus one. The checked result will be returned in an error code. The API will return a “Correct” only if manufacture code and serial number are all correct and the new counter value is within the assigned error range.

@Syntax

VC/VC++

```
unsigned int WXP_AGA_Check_PTU(unsigned char *Manu_Code,  
unsigned char *Ser_Num, unsigned short PTU_Counter, unsigned  
short Err_Range, unsigned short *New_PTU_Counter)
```

@Argument

Manu_Code: This is the 16-character hexadecimal “manufacture code” you should previously programmed into the Protect-U chip.

Ser_Num: This is the 7-character hexadecimal “serial number” you should previously programmed into the Protect-U chip.

PTU_Counter: This is the 2-character count number which you get from “AGA_Set_PTU” or last “AGA_Check_PTU” check.

Err_Range: This is an integer value for the acceptable count value difference between this check and last check (the input

PTU_Counter). The normal difference should be 1. This parameter enables to define a wider-range difference.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.14 AGA_Write_IntrMask**@Description**

The mask value of intrusion log is used to set the monitor bits for intrusion log. To set the mask of intrusion log we can call API AGA_Set_IntrMask.

@Syntax

VC/VC++

```
unsigned int WXP_AGA_Read_IntrMask(unsigned char *Mask)
```

@Argument

Mask: We use the Mask value to set monitor bits for intrusion log.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.15 AGA_Read_IntrMask**@Description**

Get the mask value of intrusion log.

@Syntax

VC/VC++

```
unsigned int WXP_AGA_Write_IntrMask(unsigned char Mask)
```

@Argument

Mask: We use the Mask value pointer to set the current monitor bits value for intrusion log.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.16 AGA_Get_RNG**@Description**

ACE-B5692 has a 16-bit hardware real random number generator.

@Syntax

VC/VC++

unsigned int WXP_AGA_Get_RNG(unsigned short *Buf)

@Argument

Buf: This is a pointer assigned by you. This pointer should point to the starting point of the buffer where you want the API to store the returned log.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.17 AGA_Write_iBtn**@Description**

To write a byte of data or any command to iButton.

@Syntax

VC/VC++

unsigned int WXP_AGA_Write_iBtn(unsigned char Data)

@Argument

Data: A byte of data or command we want to write to or send to iButton.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.18 AGA_Read_iBtn**@Description**

To read a byte of data or any command to iButton.

@Syntax

VC/VC++

unsigned int WXP_AGA_Read_iBtn(unsigned char *Data)

@Argument

Data: A byte of data or command we want to write to or send from iButton.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.19 AGA_RESET_iBtn

@Description

To resent a byte of data or any command to iButton.

@Syntax

VC/VC++

unsigned int WXP_AGA_Reset_iBtn()

@Argument

no.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.20 AGA_Read_I2C_Byte**@Description**

Read data from assigned IO address of specified I2C device through AGA.

@Syntax

VC/VC++

unsigned int WXP_AGA_Read_I2C_Byte(unsigned char
i2c_device_addr, unsigned char i2c_addr, unsigned char *i2c_Rdata)

@Argument

i2c_device_addr: the address of the target I2C device.

i2c_addr: the IO address of target I2C device.

i2c_Rdata: Data read from target I2C device.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.1.1.21 AGA_Write_I2C_Byte**@Description**

Write data to assigned IO address of specified I2C device through AGA.

@Syntax

VC/VC++

unsigned int WXP_AGA_Write_I2C_Byte(unsigned char
i2c_device_addr, unsigned char i2c_addr, unsigned char i2c_Wdata)

@Argument

i2c_device_addr: the address of the target I2C device.

i2c_addr: the IO address of target I2C device.

i2c_Wdata: Data write to target I2C device.

@Return Code

Please refer to Chapter 6.3 Error Code.

6. 2 Linux AGA Driver and Libraries

Operating System: Kernel 2.6.27.5-117.fc10.i686.

Coding Environment: Gcc.

- File Discription:
 - AGA_5692_LIB.a → Static Library of AGA device.
 - AGA_5692_LIB.so → Dynamic Library of AGA device.
 - AGA_5692_AP → The test program of AGA device.

6.2.1 Linux AGA-API Library

6.2.1.1 AGA_InitLib

@Description

Set com port.

@Syntax

C/C++

unsigned int LNX_AGA_InitLib(unsigned short usCOM_Num)

@Argument

usCOM_Num: This is the serial port number used for communicating with ACE-B5692 in your computer. For example, if you use the COM1 to communicate with ACE-B5692, this value should be "1".

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.2 AGA_DelInitLib

@Description

This function is used to release the DelInitLib ACE-B5692.

@Syntax

C/C++

unsigned int LNX_AGA_DelInitLib()

@Argument

NONE.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.3 AGA_Get_RTC

@Description

Set up AGA RTC Time.

@Syntax

C/C++

```
unsigned int LNX_AGA_Get_RTC(struct tm* pTm)
```

@Argument

PTm: This is the pointer of the structure for returned RTC data.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.4 AGA_Set_RTC

This API is used to read the current RTC data stored in real time clock.

It uses standard structure defined in <time.h>.

@Syntax

C/C++

```
unsigned int LNX_AGA_Set_RTC(unsigned short Type,struct tm*  
pTm)
```

@Argument

Type: user mode or system mode user =1 , system = 2.

Struct tm: Time for pTm Time format.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.5 AGA_Get_IntrLog**@Description**

Read Record Lo or Hi form the IntrLog Record.

@Syntax

C/C++

```
unsigned int LNX_AGA_Get_IntrLog(unsigned char  
*LogBuf,unsigned short LogLen,unsigned short *ReadLen)
```

@Argument

LogBuf : Read log buffer.

LogLen: Read record times.

ReadLen : Return data times.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.6 AGA_Cls_IntrLog**@Description**

Clean Record Lo or Hi to the IntrLog Record.

@Syntax

C/C++

unsigned int LNX_AGA_Cls_IntrLog()

@Argument

no.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.7 AGA_Get_BattLog1**@Description**

Read AGA Record from the battery log Record.

@Syntax

C/C++

unsigned int LNX_AGA_Get_BattLog1(unsigned char
*LogBuf,unsigned short LogLen,unsigned short *ReadLen)

@Argument

LogBuf : Read log buffer.

LogLen: Read record times.

ReadLen : Return data times .

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.8 AGA_Cls_BattLog1**@Description**

Clean AGA Record to the battery log Record.

@Syntax

C/C++

unsigned int LNX_AGA_Cls_BattLog1()

@Argument

no.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.9 AGA_Get_BattLog2**@Description**

Read AGA Record from the battery log Record.

@Syntax

C/C++

```
unsigned int LNX_AGA_Get_BattLog1(unsigned char
*LogBuf,unsigned short LogLen,unsigned short *ReadLen)
```

@Argument

LogBuf : Read log buffer.

LogLen: Read record times.

ReadLen : Return data times.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.10 AGA_Cls_BattLog2**@Description**

Clean AGA Record to the battery log Record.

@Syntax

C/C++

```
unsigned int LNX_AGA_Cls_BattLog1()
```

@Argument

no.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.11 AGA_GET_AGAVER**@Description**

Read AGA Version.

@Syntax

C/C++

```
unsigned int LNX_AGA_GET_AGAVER(unsigned char *ver)
```

@Argument

ver : AGA Version.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.12 AGA_Set_Protect-U**@Description**

Your application software can set the data (manufacture code and serial number) into Protect-U by calling the ACE-B5692 Protect-U Library API "Lib_Set_PTU", and this API will return the count value in Protect-U.

@Syntax

C/C++

unsigned int LNX_AGA_Set_Protect-U(unsigned char *Manu_Code,
unsigned char *Ser_Num, unsigned short *PTU_Counter)

@Argument

Manu_Code: This is the 16-character hexadecimal "Manufacture code" you are going to program into the Protect-U chip.

Ser_Num: This is the 7-character hexadecimal "serial number" you are going to program into the Protect-U chip.

PTU_Counter: Return counter

@Return Code

Please refer to Chapter 6.3 Error Code..

6.2.1.13 AGA_Check_Protect-U**@Description**

This API is used to validate the consistency of Protect-U. It will return an error code and Protect-U counter value. Please keep the counter value for the next **time** check. This API needs input of manufacture code, serial number, previous counter number and allowed counter errors. This API will use the input data to require a set of encrypted data from Protect-U and decrypt an encrypted data. The decrypted data contains a new counter value which is normally the previous counter value plus 1. The checked result will be returned in an error code. The API will return a "Correct" only if manufacture code and serial number are all correct and the new counter value is within the assigned error range.

@Syntax

C/C++

unsigned int LNX_AGA_Check_PTU(unsigned char *Manu_Code,
unsigned char *Ser_Num, unsigned short PTU_Counter, unsigned
short Err_Range, unsigned short *New_PTU_Counter)

@Argument

Manu_Code: This is the 16-character hexadecimal "Manufacture code" you previously programmed into the Protect-U chip.

Ser_Num: This is the 7-character hexadecimal "serial number" you previously programmed into the Protect-U chip.

PTU_Counter: This is the 2-character count number which you **get** from "AGA_Set_PTU" or last "AGA_Check_PTU" check.

Err_Range: This is an integer value for the acceptable count value difference between this check and last check (the input PTU_Counter). The normal difference should be 1. This parameter enables you to define a wider-range difference.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.14 AGA_Write_IntrMask**@Description**

The mask value of intrusion log is used to set the monitor bits for intrusion log. To set the mask of intrusion log we can call API AGA_Set_IntrMask.

@Syntax

C/C++

```
unsigned int LNX_AGA_Read_IntrMask(unsigned char *Mask)
```

@Argument

Mask: We use the Mask value to set monitor bits for intrusion log.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.15 AGA_Read_IntrMask**@Description**

Get the mask value of intrusion log.

@Syntax

C/C++

```
unsigned int LNX_AGA_Write_IntrMask(unsigned char Mask)
```

@Argument

Mask: We use the Mask value pointer to set the current monitor bits value for intrusion log.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.16 AGA_Get_RNG**@Description**

ACE-B5692 has a 16-bit hardware real random number generator.

@Syntax

C/C++

```
unsigned int LNX_AGA_Get_RNG(unsigned short *Buf)
```

@Argument

Buf: This is a pointer assigned by you. This pointer should point to the starting point of the buffer where you want the API to store the returned log.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.17 AGA_Write_iBtn**@Description**

To write a byte of data or any command to iButton.

@Syntax

C/C++

unsigned int LNX_AGA_Write_iBtn(unsigned char Data)

@Argument

Data: A byte of data or command we want to write to or send to iButton

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.18 AGA_Read_iBtn**@Description**

To read a byte of data or any command to iButton.

@Syntax

C/C++

unsigned int LNX_AGA_Read_iBtn(unsigned char *Data)

@Argument

Data: A byte of data or command we want to write to or send from iButton.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.19 AGA_RESET_iBtn**@Description**

To resent a byte of data or any command to iButton.

@Syntax

C/C++

Unsigned int LNX_AGA_Reset_iBtn()

@Argument

no.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.20 AGA_Read_I2C_Byte**@Description**

Read data from assigned IO address of specified I2C device through AGA.

@Syntax

C/C++

```
unsigned int LNX_AGA_Read_I2C_Byte(unsigned char  
i2c_device_addr, unsigned char i2c_addr, unsigned char *i2c_Rdata)
```

@Argument

i2c_device_addr: the address of the target I2C device.

i2c_addr: the IO address of target I2C device.

i2c_Rdata: Data read from target I2C device.

@Return Code

Please refer to Chapter 6.3 Error Code.

6.2.1.21 AGA_Write_I2C_Byte**@Description**

Write data to assigned IO address of specified I2C device through AGA.

@Syntax

C/C++

```
unsigned int LNX_AGA_Write_I2C_Byte(unsigned char  
i2c_device_addr, unsigned char i2c_addr, unsigned char i2c_Wdata)
```

@Argument

i2c_device_addr: the address of the target I2C device.

i2c_addr: the IO address of target I2C device.

i2c_Wdata: Data write to target I2C device.

@Return Code

Please refer to Chapter 6.3 Error Code.

6. 3 Error Code Notes

ERROR_MASK_API

0xFF000000 API Mask FFxxxxxx, from 0~255

ERROR_MASK_AGA

0x00FF0000 AGA internal function code mask, Mask xxFFxxxx, range(0~255)

ERROR_MASK_GEN

0x0000FF00 General error code mask, Mask xxxxFFxx, range(0~255)

For return error format



ERROR_MASK_API

- 0x01 Error code for API AGA_Set_PTU
- 0x02 Error code for API AGA_Check_PTU
- 0x03 Error code for API AGA_Get_RTC
- 0x04 Error code for API AGA_Set_RTC
- 0x05 Error code for API AGA_Get_IntrLog
- 0x06 Error code for API AGA_Cls_IntrLog
- 0x07 Error code for API AGA_Get_BattLog1
- 0x08 Error code for API AGA_Cls_BattLog1
- 0x09 Error code for API AGA_Get_BattLog2
- 0x0A Error code for API AGA_Cls_BattLog2
- 0x0B Error code for API AGA_Get_BattLog3
- 0x0C Error code for API AGA_Cls_BattLog3
- 0x0D Error code for API AGA_Get_RNG
- 0x0E Error code for API AGA_Cls_iBtn
- 0x0F Error code for API AGA_Read_iBtn
- 0x10 Error code for API AGA_Write_iBtn
- 0x11 Error code for API AGA_Read_IntrMa
- 0x12 Error code for API AGA_Write_IntrM
- 0x13 Error code for API AGA_Read_I2C_By
- 0x14 Error code for API AGA_Write_I2C_B
- 0x15 Error code for API AGA_Get_CaseOpe
- 0x16 Error code for API AGA_Cls_CaseOpe
- 0x17 Error code for API AGA_Get_Reset_L

- 0x18 Error code for API AGA_Cls_Reset_L
- 0x19 Error code for API AGA_GET_AGAVER

ERROR_MASK_AGA

- 0x00 No errors
- 0xE0 Handshake error with Serial Port (Wait 0x55)
- 0xE1 (Read) Check PTU duty cycle Hi signal fail
- 0xE2 (Read) Check PTU duty cycle Low signal fail
- 0xE3 (Read) Check PTU Header 1st Low fail
- 0xE3 (Read) Check PTU Header 2nd Low fail
- 0xE3 (Read) Check PTU Header 3rd Low fail
- 0xE4 (Read) Check PTU Data bit high signal fail
- 0xE5 (Read) Check PTU Data bit Low signal fail
- 0xE6 (Write) Verify Fail after program PTU.
- 0xE7 Check I2C slave address ACK fail
- 0xE8 Check I2C word address ACK fail
- 0xE9 Check I2C data byte ACK fail
- 0xEA iButton device No Presence
- 0xEE Loops Timeout fail

ERROR_MASK_GEN

- 0x01 Serial Port open error
- 0x02 Serial Port close error
- 0x03 Serial Port read error
- 0x04 Serial Port write error
- 0x05 Serial Port handshaking error with AGA (Wait 0xAA)
- 0x06 The arguments pass to API is NULL
- 0x07 Manufacture code formate error
- 0x08 Serial number formate error
- 0x09 Count number out of range
- 0x0A Time formate error
- 0x0B Others Input formate error
- 0xFF Others unknow errors

7 ELECTRICAL CHARACTERISTICS

7.1 Basic Electrical Characteristics table

Electrical Characteristics					
Symbol	Parameter / Condition	Value			Unit
		Min.	Typ.	Max.	
I.IH	Isolation Input Voltage high level threshold	2.2	12	20	V
I.IL	Isolation input voltage low level threshold	-10		1.2	V
I.IC	Isolation Input current			25	mA
O.C	Open Drain Output Voltage			20	V
O.C	Open Drain Sink current		500	550	mA
DBIOH	Operating voltage high level threshold	3	5	5.5	V
DBIOL	Operation voltage low level threshold	-0.3		0.8	V
DBIO	Operating current		20	35	mA
+12V	Power In to ACE-B5692	11.4	12	12.6	V
+5V	Power In to ACE-B5692	4.85	5	5.25	V
C.C talk	Communication pin high threshold	3.0	5.0	5.5	V
C.C talk	Communication pin low level Threshold	-0.3		0.8	V
RS232	Maximum Working baud rate			115.2	Kbps
AGA	Maximum Working baud rate		19.2		Kbps
Blight	Backlight operating voltage	11.4	12	12.6	V
T.P.C	Total power consumption in ACE-B5692 without External device @ Pentium –M 2 0Ghz		70		W

7.2 72 Pins Golden Fingers

Component Side				Solder Side		
I/O	Port/Bit	Function	Pin	Function	Port/Bit	I/O
			1			
		Speaker Base+	2	Speaker Base+		
O		SPEAKER LEFT +	3	Speaker Left -		
I.I	B0	Button 1	4	Speaker Right-		
I.I	B1	Button 2	5	SPEAKER RIGHT+		
I.I	B2	Button 3	6	Door SW2	A1	I
I.I	B3	Button 4	7	Door SW3	A2	I
I.I	B4	Button 5	8	Door SW4	A3	I
I.I	B5	Button 6	9	Door SW5	A4	I
I.I	B6	Button 7	10	Touch-Cal Key-Lock	D3	I.I
I.I	B7	Button 8	11	Spare Key-Lock	D4	I.I
I.I	C0	Button 9	12	Coin-Enable	I0	O.C.
I.I	C1	Button10	13	Bill-Enable	I1	O.C.
			14			
I.I	D0	Dissolve Key-Lock	15	Button 15	C6	I.I
I.I	C2	Button11	16	Button 16	C7	I.I
I	A0	Door SW1	17			
I.I	E0	Coin-In Signal A	18	Button 12	C3	I.I
I.I	E2	Bill-In	19	Coin-In Signal B	E1	I.I
I.I	D1	OM Key-Lock	20	Setup Key-Lock (Hand Pay)	D2	I.I
I.I	C5	Button 14	21	Button 13	C4	I.I
GND		GND	22	Hopper Sensor	E3	I.I
O.C.	H7	Spare Meter1	23	Lamp13	G4	O.C.
O.C.	H0	Key-In Meter	24	Hand-Pay Meter1	H5	O.C.
O.C.	H1	Bill-In Meter	25	Hand-Pay	H6	O.C.

				Meter2		
O.C.	H2	Coin-In Meter	26	Lamp14	G5	O.C.
O.C.	H3	Pay-Out Meter	27	Lamp15	G6	O.C.
O.C.	H4	Key-Out Meter	28	Lamp16	G7	O.C.
O.C.	F0	Lamp1	29	Lamp7	F6	O.C.
O.C.	F1	Lamp2	30	Lamp8	F7	O.C.
O.C.	F2	Lamp3	31	Lamp9	G0	O.C.
O.C.	F3	Lamp4	32	Lamp10	G1	O.C.
O.C.	F4	Lamp5	33	Lamp11	G2	O.C.
O.C.	F5	Lamp6	34	Lamp12	G3	O.C.
		GND	35	GND		GND
GND		GND	36	GND		GND

7.3 20-pin Golden Finger

Component Side				Solder Side		
I/O	Port/Bit	Function	Pin	Function		I/O
		GND	1	GND		
		GND	2	GND		
		+5V	3	+5V		
		+5V	4	+5V		
		+12V	5	+12V		
		+12V	6	+12V		
O.C.	I2	Hopper SSR	7			
			8			
		GND	9	GND		
		GND	10	GND		

7.4 JAMMA Golden Finger

Component Side				Solder Side		
I/O	Port/ Bit	Function	Pin	Function	Port/Bit	I/O
P		GND	1	GND		P
P		GND	2	GND		P
P		Power +5V	3	Power +5V		P
P		Power +5V	4	Power +5V		P
P		+24V for ccTalk	5	+24V for ccTalk		P
P		Power +12V	6	Power +12V		P
I	A0	Intrusion 0	7	Intrusion 1 (Door Switch)	A1	I
			8	Output counter	H0	O.C.
			9	Input counter	H4	O.C.
A.O		Speaker Left+	10	Speaker Left-		A.P
A.O		Speaker Right+	11	Intrusion 2	A2	I
		Speaker Right -	12	Tower lamp 3	G2	O.C
			13			
			14	Test Switch	D1	I.I and I.P.H
I.I and I.P.H	B7	Button8	15	Lamp8	F7	O.C.
I.I and I.P.H	B6	Button7	16	Lamp7	F6	O.C.
I.I and I.P.H	B5	Button6	17	Lamp6	F5	O.C.
I.I and I.P.H	B2	Button3	18	Lamp3	F2	O.C.
I.I and I.P.H	B3	Button4	19	Lamp4	F3	O.C.
I.I and I.P.H	B1	Button2	20	Lamp2	F1	O.C.
I.I and I.P.H	B0	Button1	21	Lamp1	F0	O.C.
I.I and I.P.H	B4	Button5	22	Lamp5	F4	O.C.
I.I and I.P.H	D0	Button "Start"	23	Lamp "Start"	H1	O.C.
I.I and I.P.H	D2	REFILL key	24	REFILL counter	H2	O.C.
O.C	G0	Tower Lamp 1	25	CcTalk0 data	COM3	DIO
O.C	G1	Tower Lamp 2	26			
P		GND	27	GND		P
P		GND	28	GND		P

7.5 Port assignment

Port/Bit	I/O	Fruit	JAMMA	Remark
A0	I	Door SW1	Intrusion 0	AGA, Intrusion 0
A1	I	Door SW2	Intrusion 1 (Door Switch)	AGA, Intrusion 1
A2	I	Door SW3	Intrusion 2	AGA, Intrusion 2
A3	I	Door SW4		AGA, Intrusion 3
A4	I	Door SW5		AGA, Intrusion 4
A5	I			SRAM Battery Low
A6				(Optional on board Switch) AGA Intrusion 6,
A7				AGA Intrusion 7. On board Chassis Switch detector
B0	I.I	Button 1	Button 1	
B1	I.I	Button 2	Button 2	
B2	I.I	Button 3	Button 3	
B3	I.I	Button 4	Button 4	
B4	I.I	Button 5	Button 5	
B5	I.I	Button 6	Button 6	
B6	I.I	Button 7	Button 7	
B7	I.I	Button 8	Button 8	
C0	I.I	Button 9		
C1	I.I	Button 10		
C2	I.I	Button 11		
C3	I.I	Button 12		
C4	I.I	Button 13		

C5	I.I	Button 14		
C6	I.I	Button 15		
C7	I.I	Button 16		
D0	I.I	Dissolve Key-Lock	Button "Start"	
D1	I.I	OM Key-Lock	Test Switch	
D2	I.I	Setup Key-Lock (Hand Pay)	REFILL key	
D3	I.I	Touch-Cal Key-Lock		
D4	I.I	Spare Key-Lock		
D5	I			
D6	O			SPI-DI
D7	O			SPI-DO
E0	I.I	Coin-In Signal A		
E1	I.I	Coin-In Signal B		
E2	I.I	Bill-In		
E3	I.I	Hopper Sensor		
E4	O			SPI-CLK
E5				SPI-CS
E6	I	Coin enable feedback		
E7	I	Bill enable feedback		
F0	O.C.	Lamp1	Lamp1	
F1	O.C.	Lamp2	Lamp2	
F2	O.C.	Lamp3	Lamp3	
F3	O.C.	Lamp4	Lamp4	
F4	O.C.	Lamp5	Lamp5	
F5	O.C.	Lamp6	Lamp6	
F6	O.C.	Lamp7	Lamp7	
F7	O.C.	Lamp8	Lamp8	
G0	O.C.	Lamp9	Tower lamp 1	1000 mA
G1	O.C.	Lamp10	Tower lamp 2	1000 mA
G2	O.C.	Lamp11	Tower lamp 3	1000 mA
G3	O.C.	Lamp12		
G4	O.C.	Lamp13		
G5	O.C.	Lamp14		
G6	O.C.	Lamp15		
G7	O.C.	Lamp16		
H0	O.C.	Key-In Meter	Output counter	
H1	O.C.	Bill-In Meter	Lamp "Start"	

H2	O.C.	Coin-In Meter	REFILL counter	
H3	O.C.	Pay-Out Meter		
H4	O.C.	Key-Out Meter	Input counter	
H5	O.C.	Hand-Pay Meter1		
H6	O.C.	Hand-Pay Meter2		
H7	O.C.	Spare Meter1		
I0	O.C.	Coin-Enable		
I1	O.C.	Bill-Enable		
I2	O.C.	Hopper SSR		
I3				
I4				
I5	Bid			I2C:DIO
I6	O			I2C: CLK
I7	O			I2C: ENA
J0	I			DIP Switch2 S0
J1	I			DIP Switch2 S1
J2	I			DIP Switch2 S2
J3	I			DIP Switch2 S3
J4	I			DIP Switch2 S4
J5	I			DIP Switch2 S5
J6	I			DIP Switch2 S6
J7	I			DIP Switch2 S7
K0	I/O			Spare I/O
K1	I/O			Spare I/O
K2	I/O			Spare I/O
K3	I/O			Spare I/O
K4	I/O			Spare I/O
K5	I/O			Spare I/O
K6	I/O			Spare I/O

K7	I/O			Spare I/O
L0	I/O			Spare I/O
L1	I/O			Spare I/O
L2	I/O			Spare I/O
L3	I/O			Spare I/O
L4	I/O			Spare I/O
L5	I/O			Spare I/O
L6	I/O			Spare I/O
L7	I/O			Spare I/O
XA0				Spare IO
XA1				Spare IO
XA2				Spare IO
XA3				Spare IO
XA4				Spare IO
XA5				Spare IO
XA6				Spare IO
XA7				Spare IO
XB0				Spare IO
XB1				Spare IO
XB2				Spare IO
XB3				Spare IO

7.6 Spare GPIO connector (2x22Pin 2.0m.m Box header)

Pin	Pin Name	Function Describe	AGC Port
1	MDAP	Motor Driver to Phase A+	K0
2	MDAN	Motor Driver to Phase A-	K1
3	MDBP	Motor Driver to Phase B+	K2
4	MDBN	Motor Driver to Phase B-	K3
5	GND		
6	GND		
7	Lamp_Inn	Inner Common control	K4
8	Lamp_Out	Outer Common control	K5
9	Lamp0	Lamp 0 On / Off control	K6
10	Lamp1	Lamp 1 On / Off control	K7
11	Lamp2	Lamp 2 On / Off control	L0
12	Lamp3	Lamp 3 On / Off control	L1
13	Lamp4	Lamp 4 On / Off control	L2
14	LED	Module Power on /off switch	L3
15	LATCH_A	Latched Port A data to buffer	L4
16	LATCH_B	Latched Port B data to buffer	L5
17	GND		
18	GND		
19	LATCH_C	Latched Port C data to buffer	L6
20	LATCH_D	Latched Port D data to buffer	L7
21	LATCH_E	Latched Port E data to buffer	XA0
22	LATCH_F	Latched Port F data to buffer	XA1
23	LATCH_G	Latched Port G data to buffer	XA2
24	LATCH_H	Latched Port H data to buffer	XA3
25	FB_A	Motor A feedback signal	XA4
26	FB_B	Motor B feedback signal	XA5
27	+5V	Logical control power	
28	+5V	Logical control power	
29	FB_C	Motor C feedback signal	XA6
30	FB_D	Motor D feedback signal	XA7
31	FB_E	Motor E feedback signal	XB0

32	FB_F	Motor F feedback signal	XB1
33	+12V	Logical control power	
34	+12V	Logical control power	
35	FB_G	Motor G feedback signal	XB2
36	FB_H	Motor H feedback signal	XB3
37	+3.3V		
38	+3.3V		
39	GND		
40	GND		
41	GND		
42	GND		
43	LATCH_OE#	Latch output Enable	PORTK_DIR
44	N.C		