FleetPC-7 User Manual

Copyright

All Rights Reserved.

Manual's first edition:

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

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

This 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.

Table of Contents

Chapter 1 System Introduction 5

1.1 Specifications 5

1.2 Packing List 6

1.3 Features 6

1.4 System Dissection 7

Chapter 2 Procedures of Assembly/Disassembly 10

2.1 2.5”HDD Installation 10

2.2 CF Card Installation 14

2.3 SIM Card Installation 17

2.4 Antenna Installation 19

Board Guide 21

Chapter 1 Introduction 21

1.1 Specifications 21

1.2 Package Contents 22

1.3 Block Diagram 22

Chapter 2 H/W Information 23

2.1 Mainboard illustration(Top Side)…………………………………………….23

2.2 Locations of IO ports & Jumper settings definition...24

Chapter 3 BIOS Settings 33

3.1 Main Setup 33

3.2 Advanced Chipset Setup 35

3.3 Superio Setup 39

3.4 Security Setup 41

3.5 Boot setup 42

3.6 Exit Setup 43

Chapter 4 Function Description 47

4.1 DC Power input connection 47

4.2 Digital Inputs 47

4.3 Digital Outputs 48

4.4 Watchdog Timer 49

4.5 RS-232 Ports 49

4.6 Serial ATA (SATA) 51

4.7 USB 51

Chapter 5 Driver And Utility Installation 52

5.1 Introduction to Driver CD Interface 52

5.2 Windows 7 32 / 64 bit Driver Installation..64
Chapter 6 Software Installation and Programming Guide… …………………………....70
6.1 CAN bus..70
6.2 GPIO and Watchdog 72
6.3 Power Subsystem..73
6.4 I-Button Function..73
6.5 API List and Descriptions………………………………………………………74
6.5.1 CAN Bus...74
6.5.2 GPIO and Watchdog 81
6.5.3 Power Subsystem..82
6.5.4 I-Button 89
Appendix A……………………………………………………………………...90

Chapter 1 System Introduction

FleetPC-7 Series with Intel second generation Celeron and Core i5/ i7 processor is a multi-function In-Vehicle computer, which is suitable for using in all kind of applications. Besides basic I/O ports like VGA, LVDS, HDMI, DVI, Hybrid Multiple Display, USB, COM. LAN, and GPIO, FleetPC-7 has complete wireless solutions for GPS / 3.5G / WiFi / Bluetooth selection, Video capture, Swappable 2.5" HDD, DC output, Driver ID, and embedded CAN Bus function to allow micro-controllers and devices to communicate with each other in vehicle. In addition, FleetPC-7 has intelligent power management function with software utility to monitor power status and control power sequence, and also compliant with most industry standards for in-vehicle usage including CE, FCC, and E-Mark.

1.1 Specifications

- Support Intel Celeron and Core i5/i7 CPU + HM65 chipset
- DDR3 SO-DIMM * 2, up to 8GB memory
- Display --- VGA + HDMI + DVI
- Combo connector --- VGA + Audio + USB + DC power
- SATA x 2 & SATA power x 2
- Swappable Anti-Shock 2.5" HDD bay x 1
- Intel GbE chip LAN x 2
- COM x 3 (2 x connector & 1 x pin header)
- CF type II socket x 1 / SIM slot x 1
- Audio connector (MIC & Line-out)
- Mini PCIe socket x 2 (Capable for WiFi / 3.5G)
- Flexible GPIO ports (8) & CAN bus
- Driver ID (Use I-Button) can certified driver,
- 9 ~ 32V DC input & customer define power management mode for ODM
- 12V DC 20W output connector for monitor

1.2 Packing List

Check if the following items are included in the package.

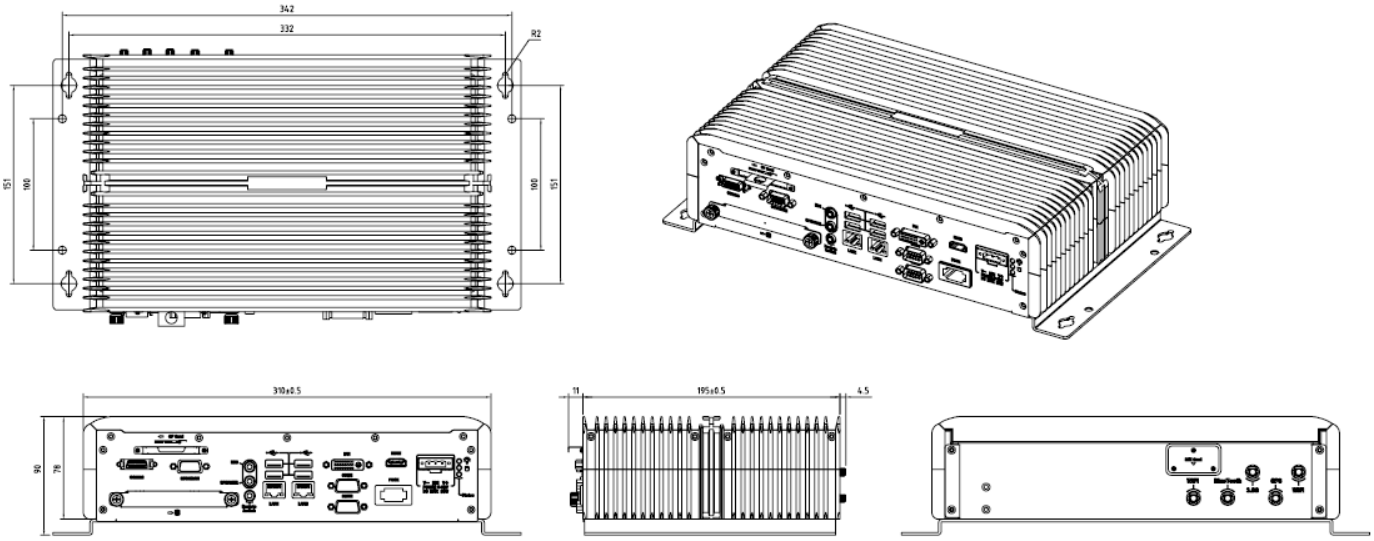
FleetPC-7	x 1
User Guide & System Driver CD	x 1
Screw pack(2.5”HDD bracket: 4pcs)	x 1
Terminal block female 3pin	x 1
Spare Fuse 10A	x 1
SATA & SATA power cable	x 1
Remote Switch Cable	x 1
GPIO/CAN/Driver ID DB15 Connector	x 1

1.3 Features

- Rugged fanless design
- Support Intel Celeron and Core i5/i7 CPU + HM65 chipset
- 2 * DDR3 SO-DIMM, up to 8GB
- Support CAN 2.0A/2.0B protocol and I-Button for driver ID
- VGA/HDMI/ DVI-I output
- Variety Wireless Communication
- Combo connector to simplify touch monitor installation

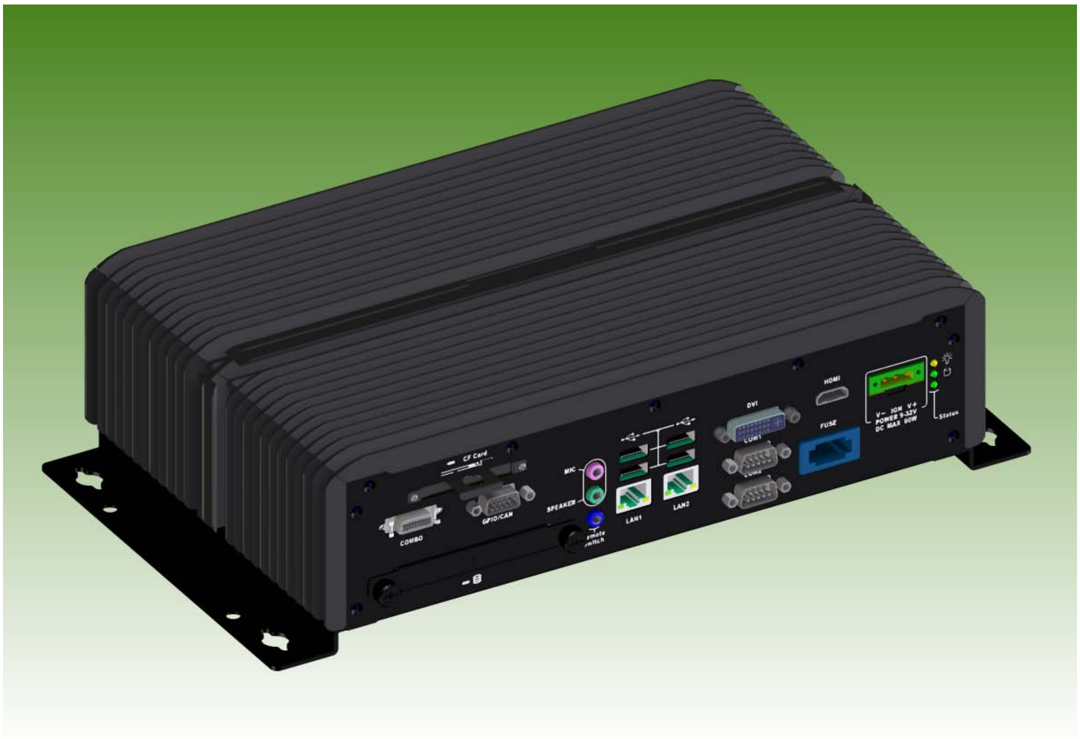
1.4 System Dissection

1.4.1 Dimensions

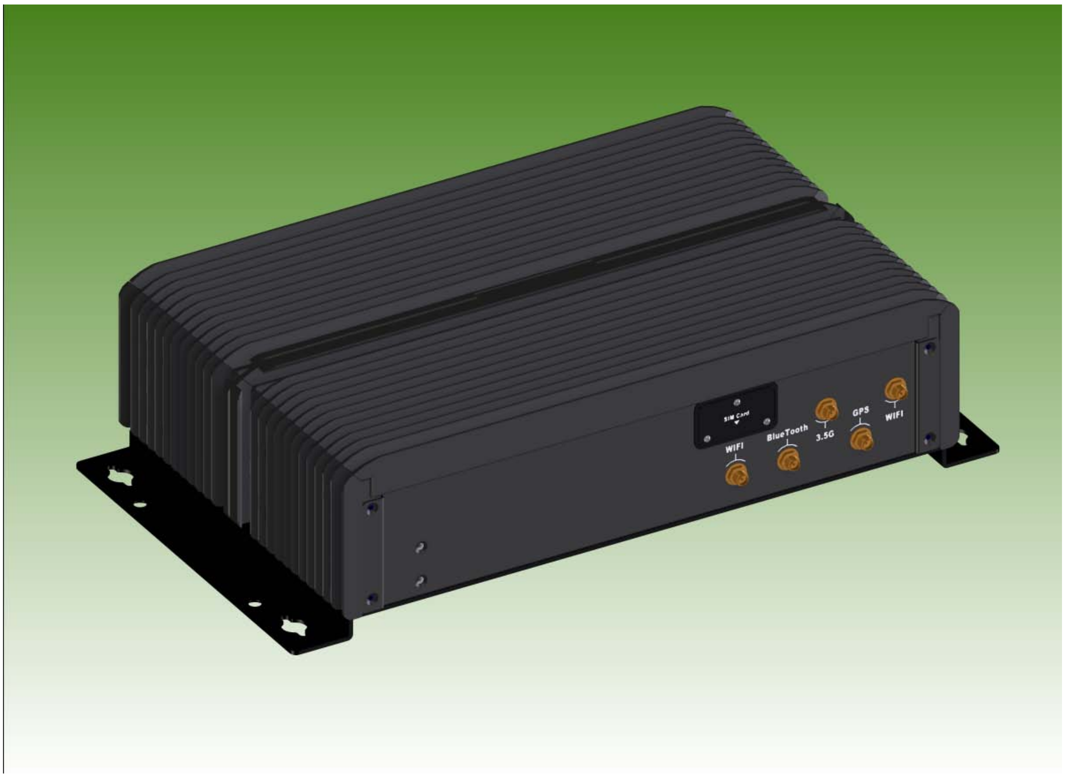


1.4.2 I/O Panel

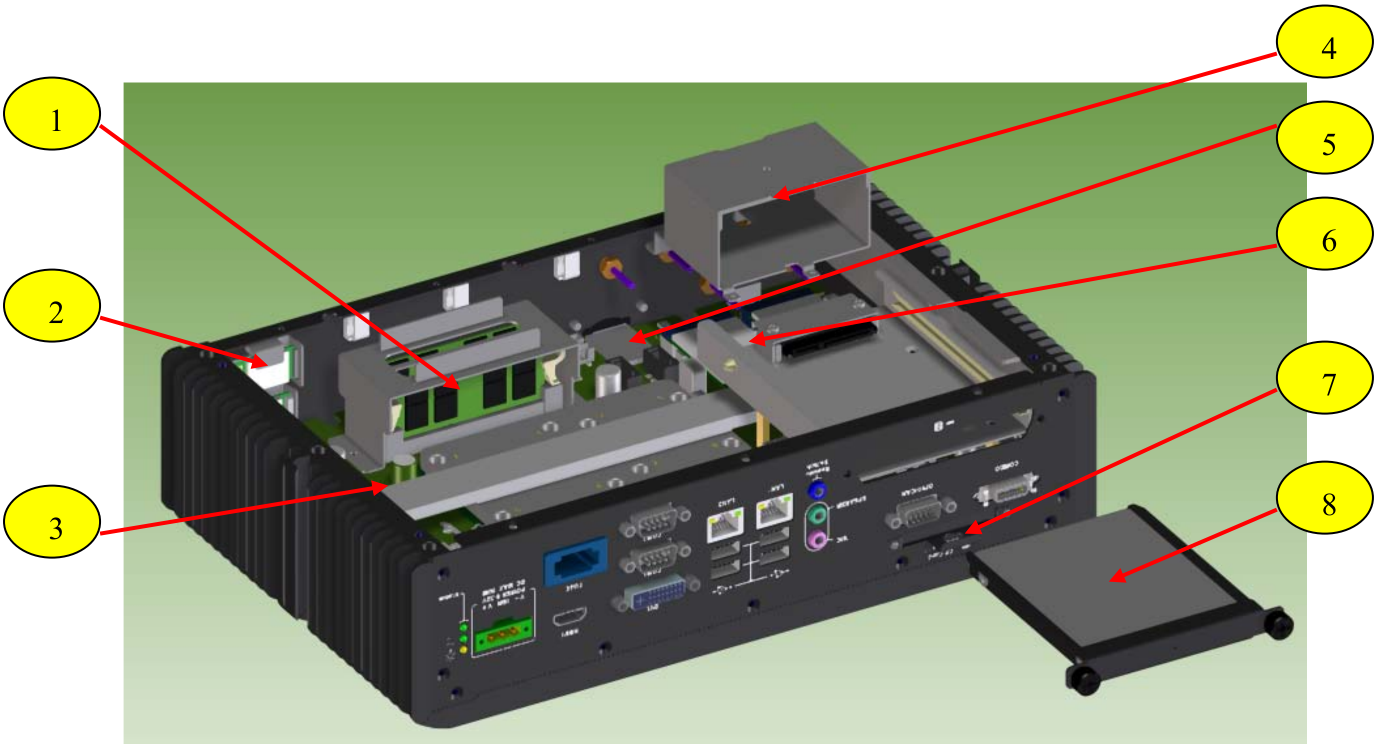
FRONT IO & PRINT



Rear I/O & PRINT



1.4.3 System Configuration



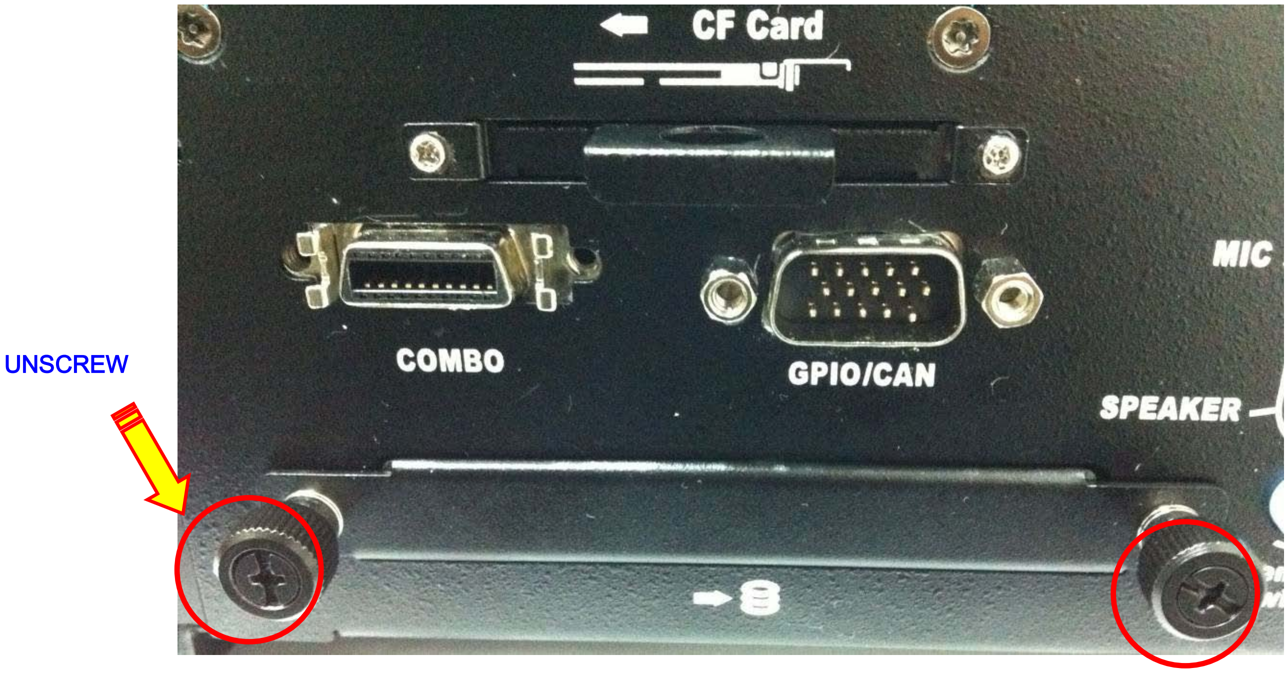
Item	Description	Quantity
1	DDR3 module	1
2	GPS & Bluetooth module	1
3	AR-V6100 main board	1
4	Module Heat-Spreader	1
5	Sim card connector	1
6	Wi-Fi & 3.5G module	1
7	CF Bracket	1
8	HDD Bracket	1

Chapter 2 Procedures of Assembly/Disassembly

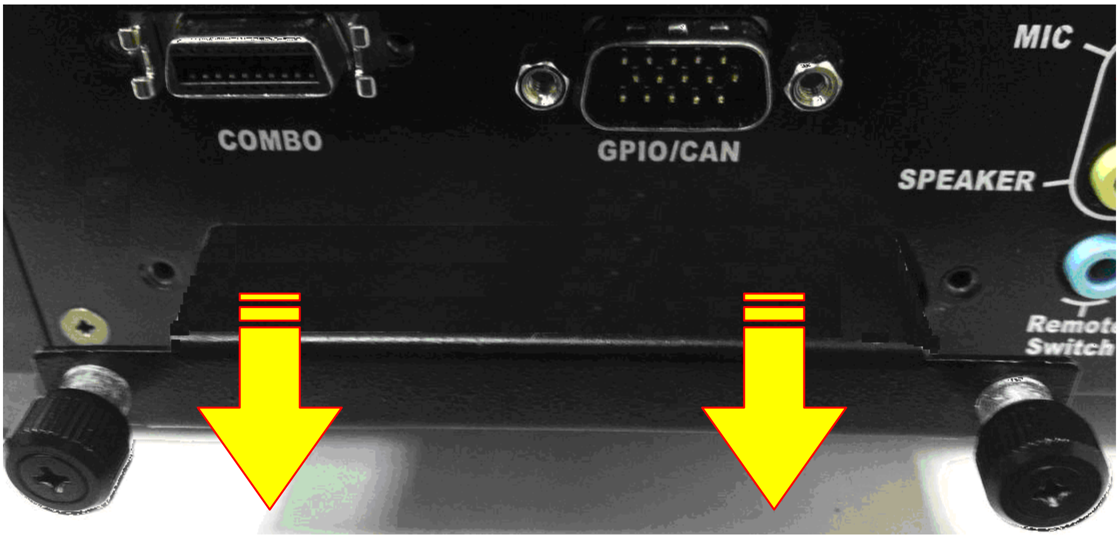
2.1 2.5"HDD Installation

The following instructions will guide you to install HDD step-by-step.

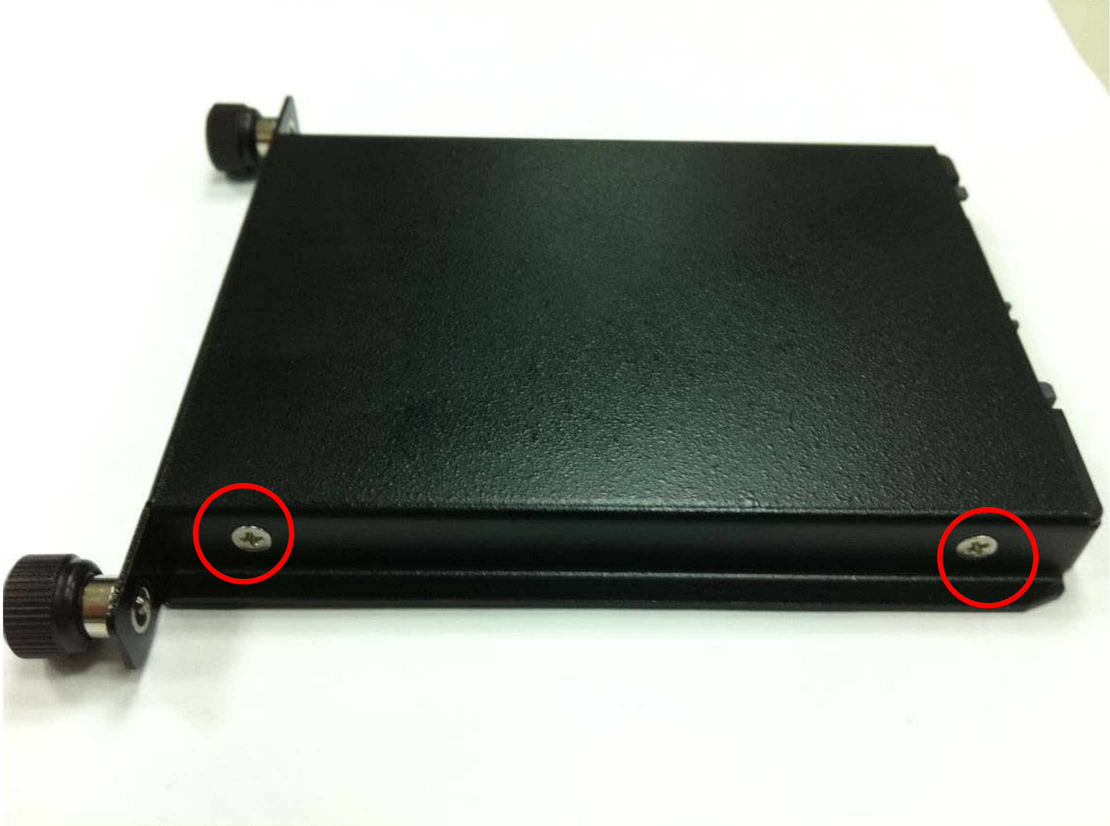
2.1.1 Unfasten the screw of chassis.

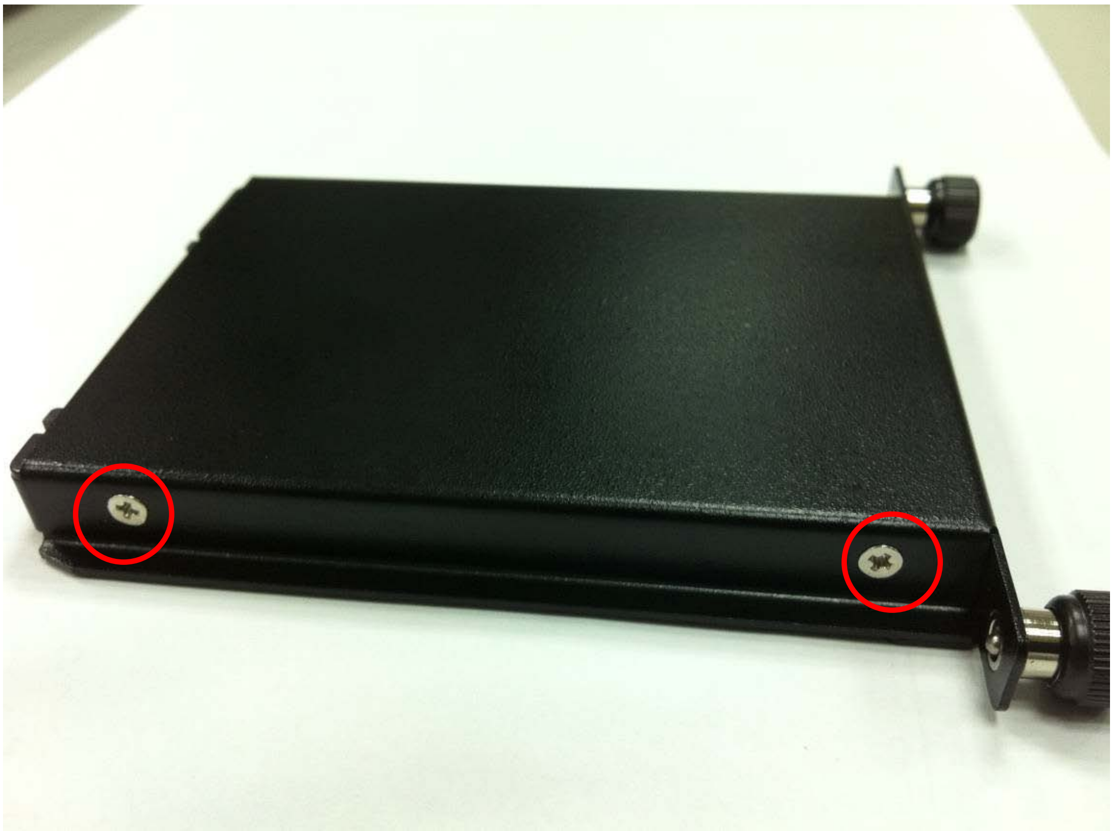


2.1.2 Open the bracket.

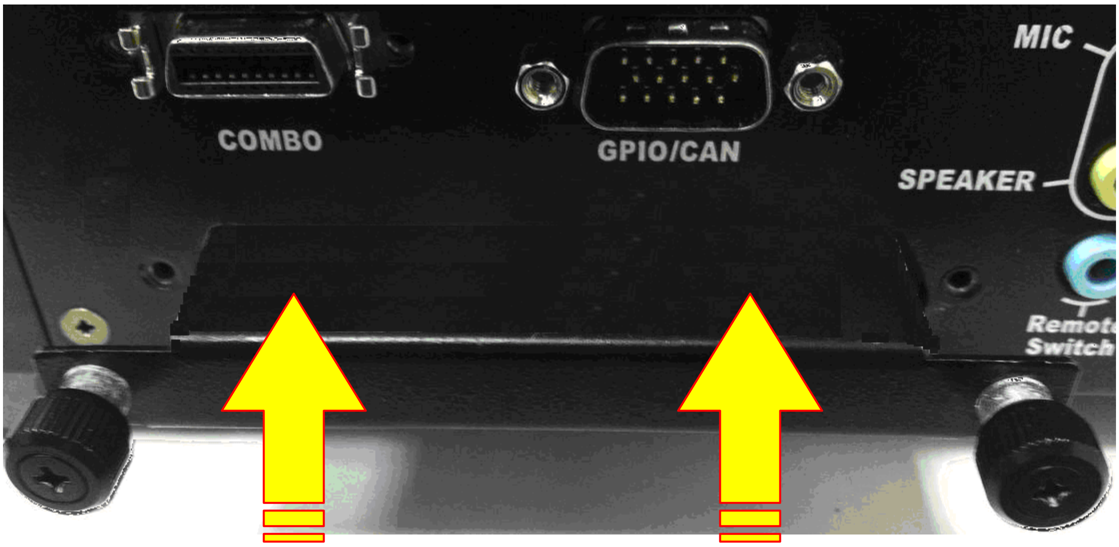


2.1.3 Assemble HDD into bracket by fastening 4 screws.

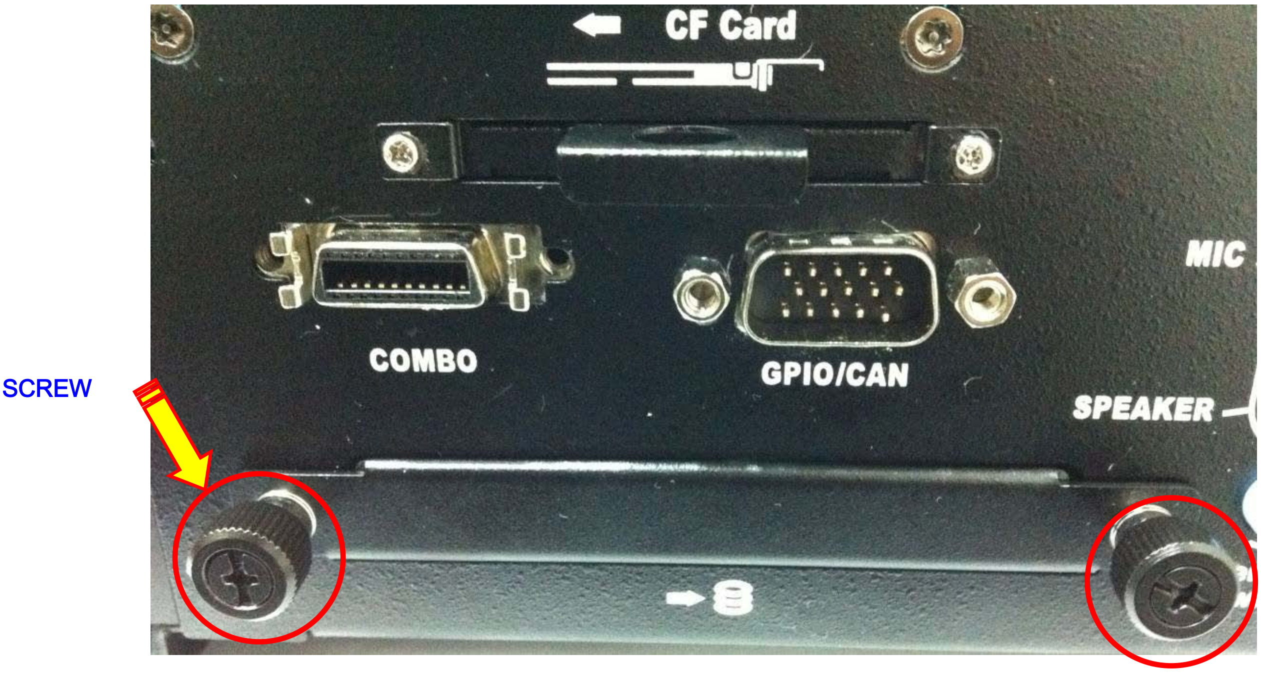




2.1.4 Assemble the HDD bracket back to system.

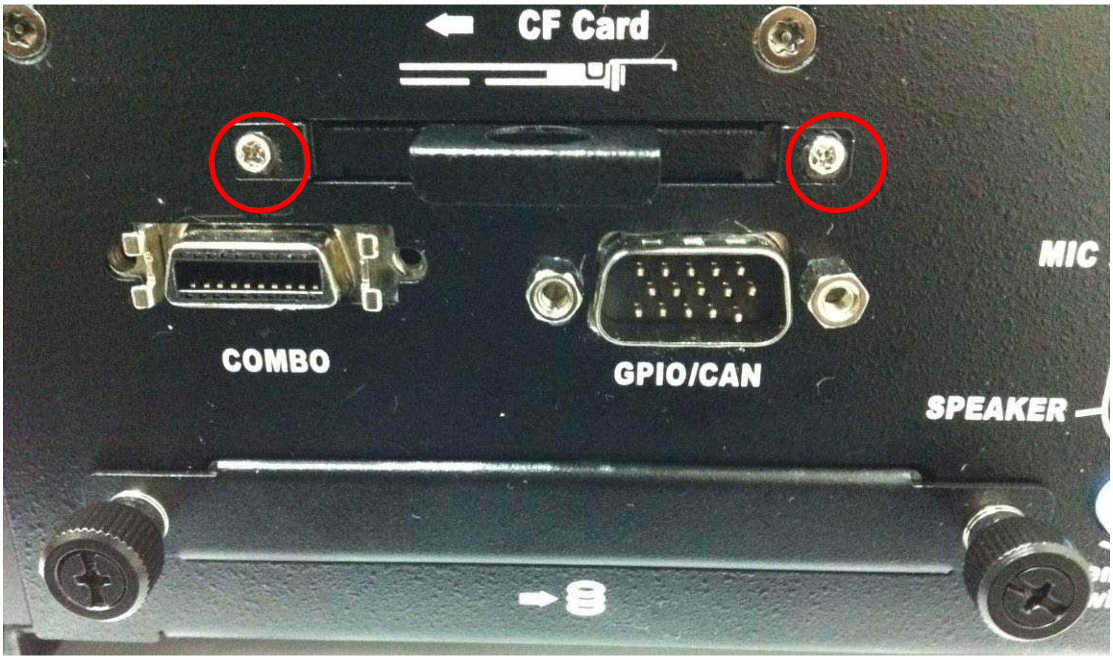


Finish.

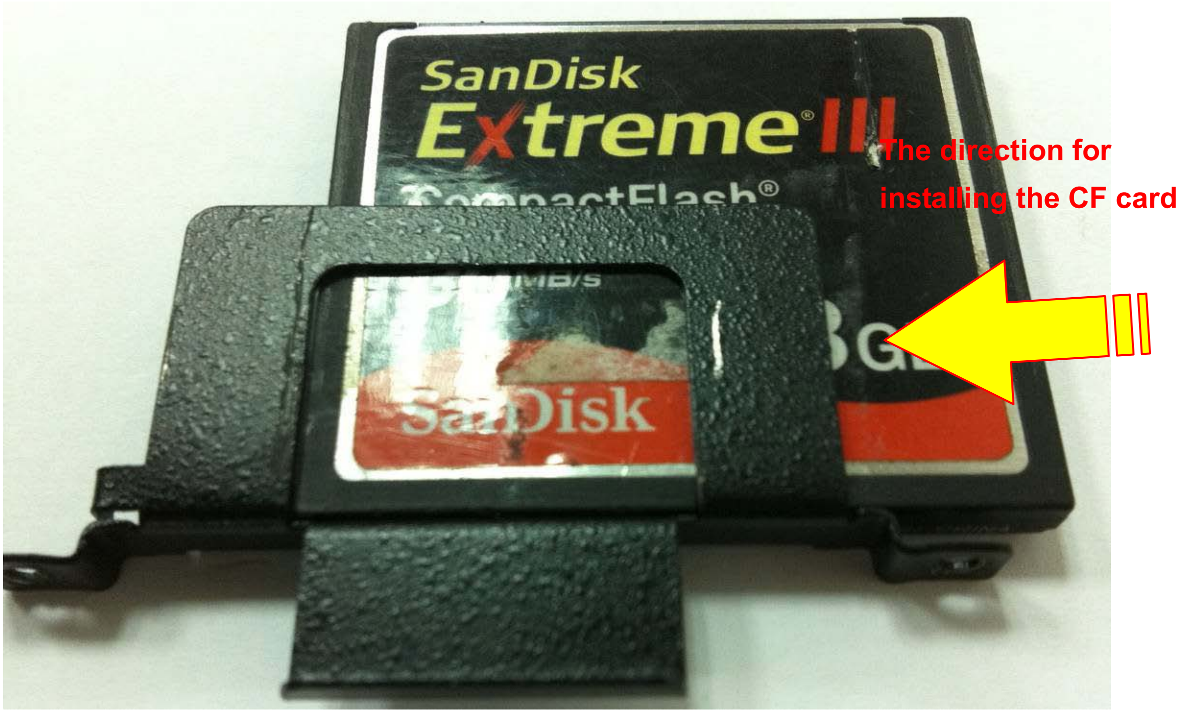


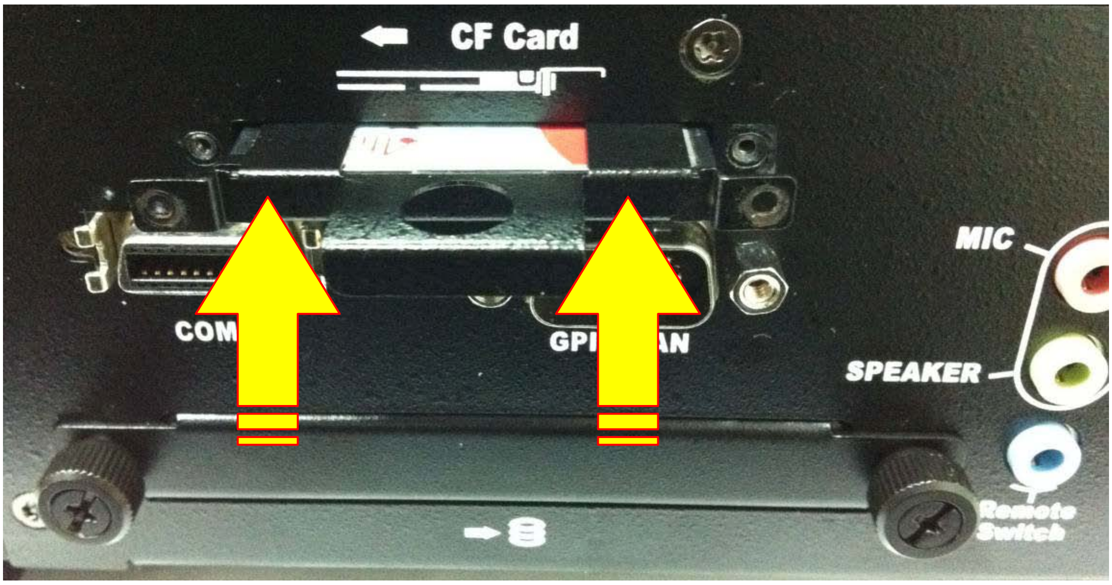
2.2 CF Card Installation

2.2.1 Unfasten the 2 s crews and pull the CF bracket from I/O panel.

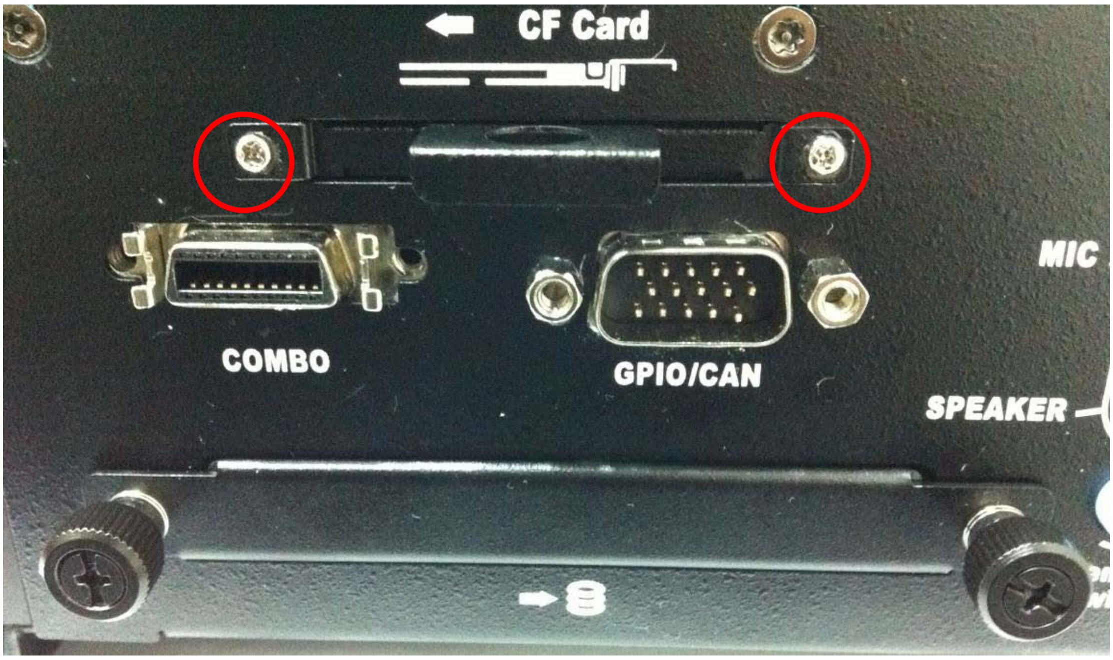


2.2.2 Assemble the CF card with CF bracket.



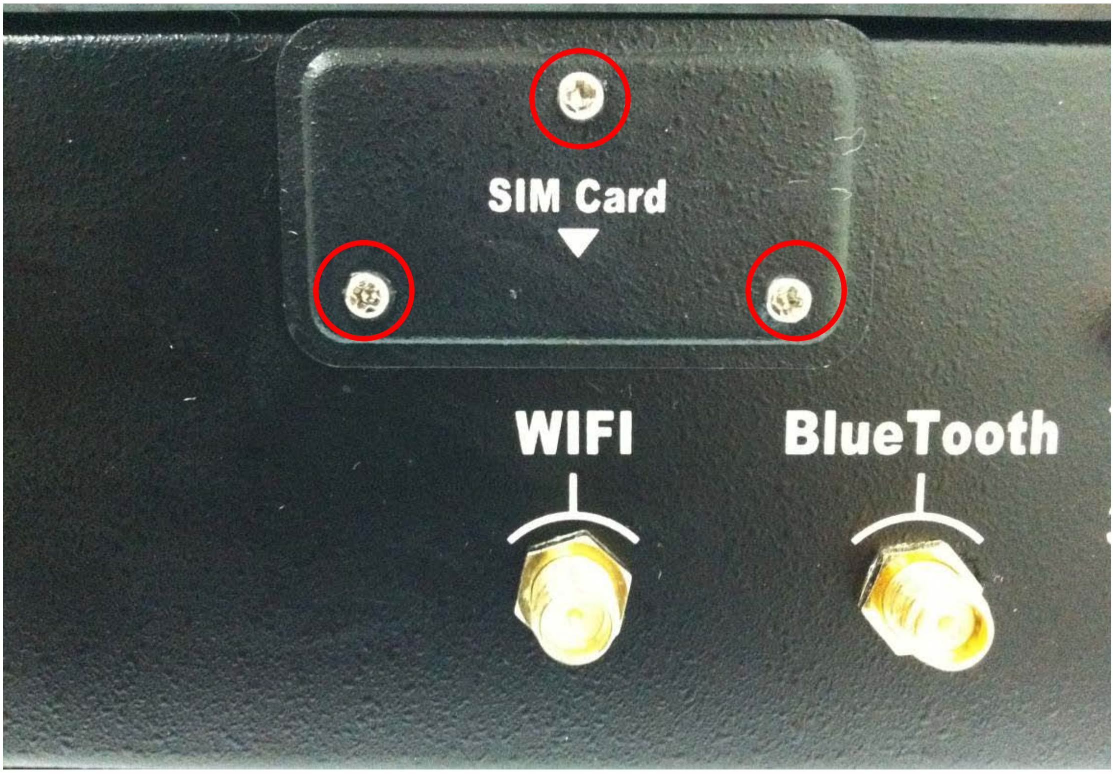


Finish.



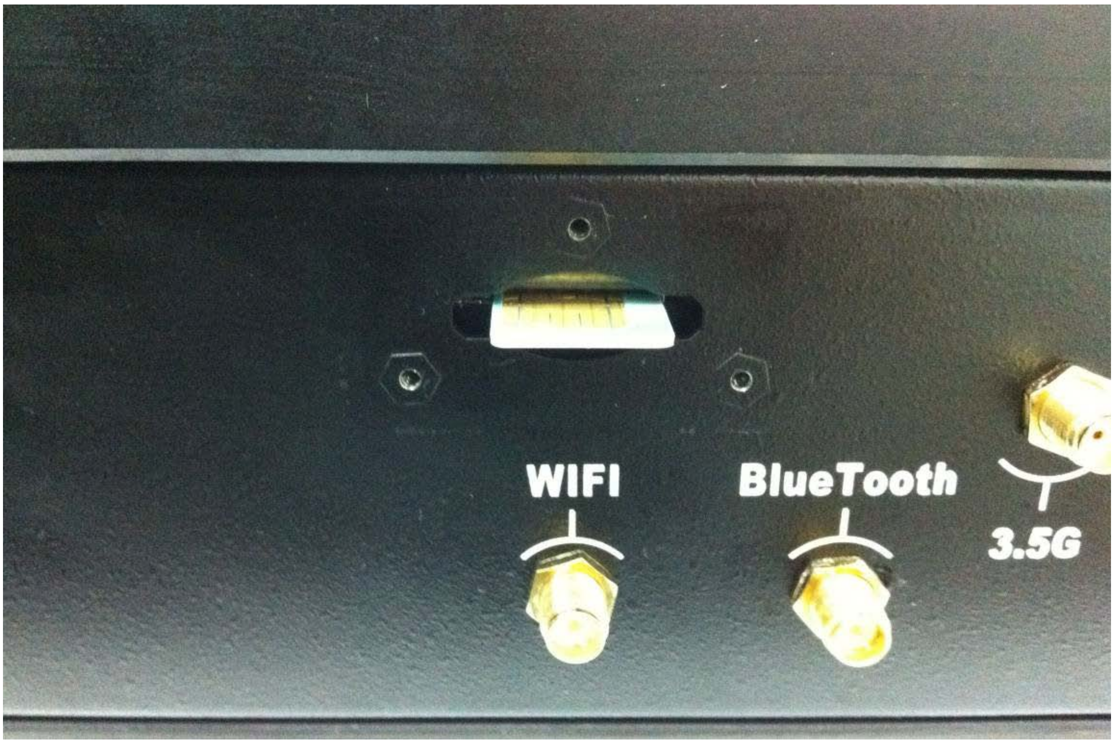
2.3 SIM Card Installation

2.3.1 Unfasten the 3 screws from Rear I/O panel.

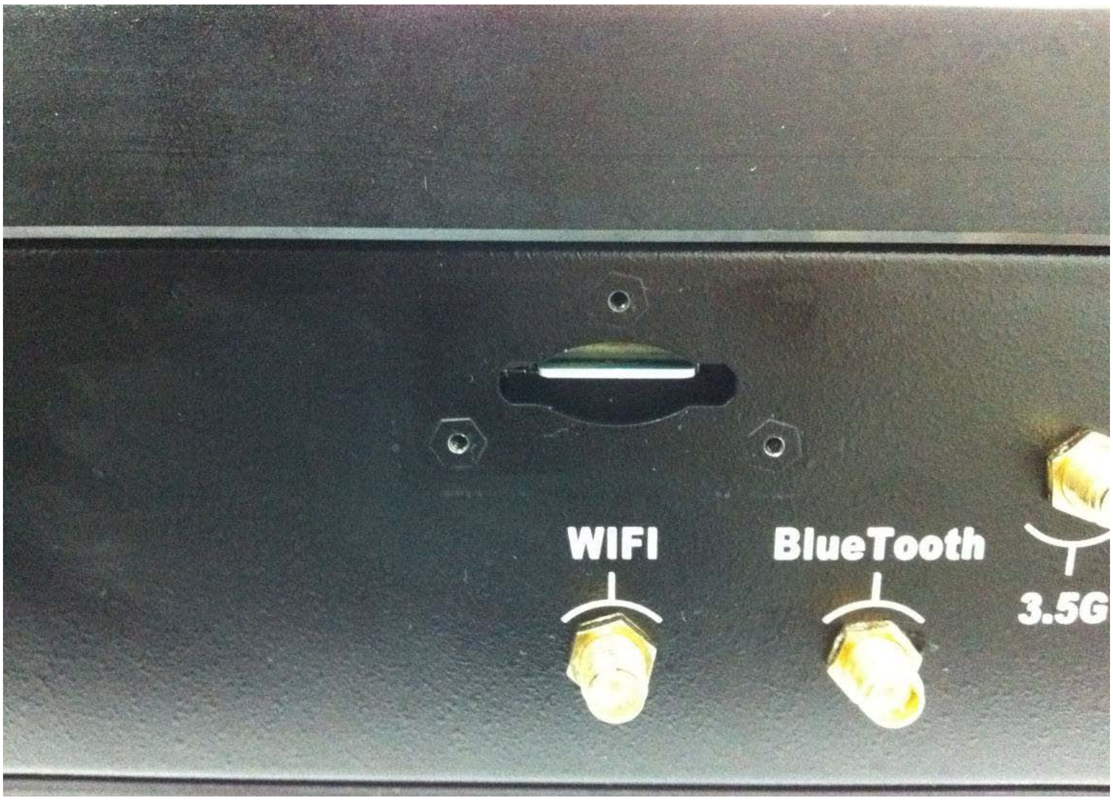


2.3.2 Insert sim card.

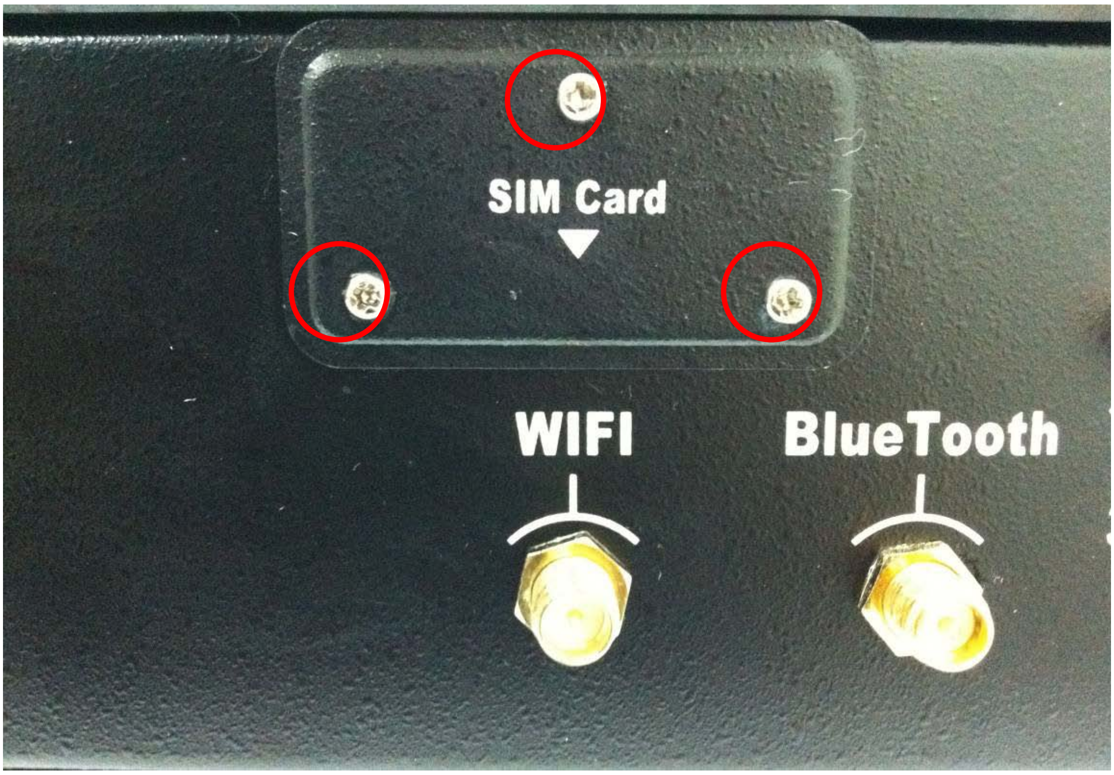
Step1.



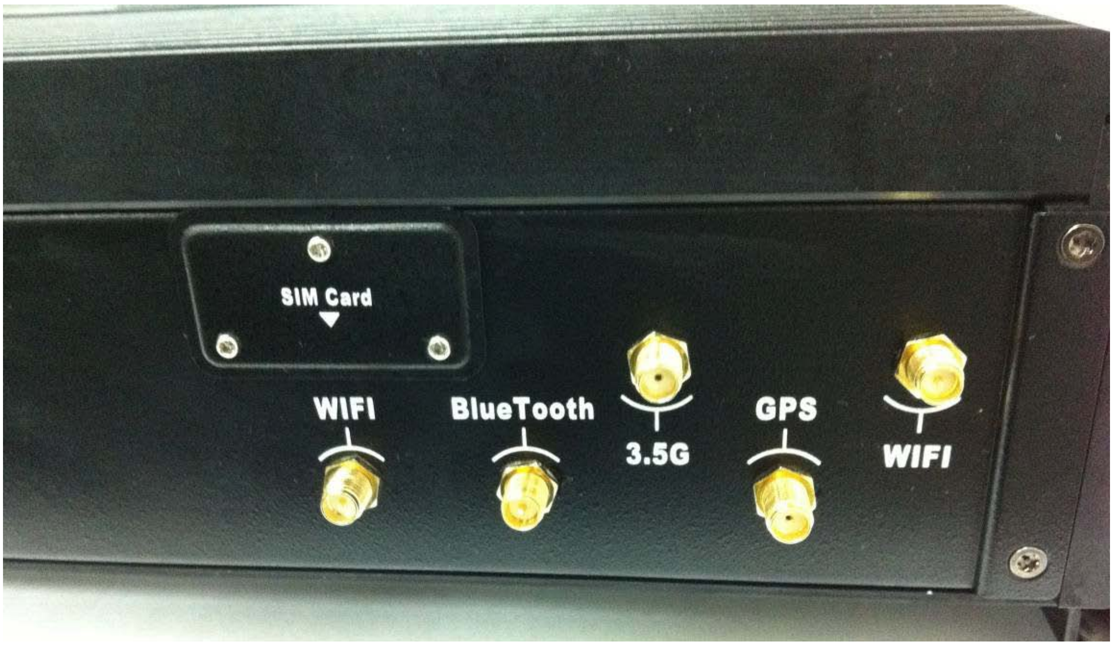
Step2.



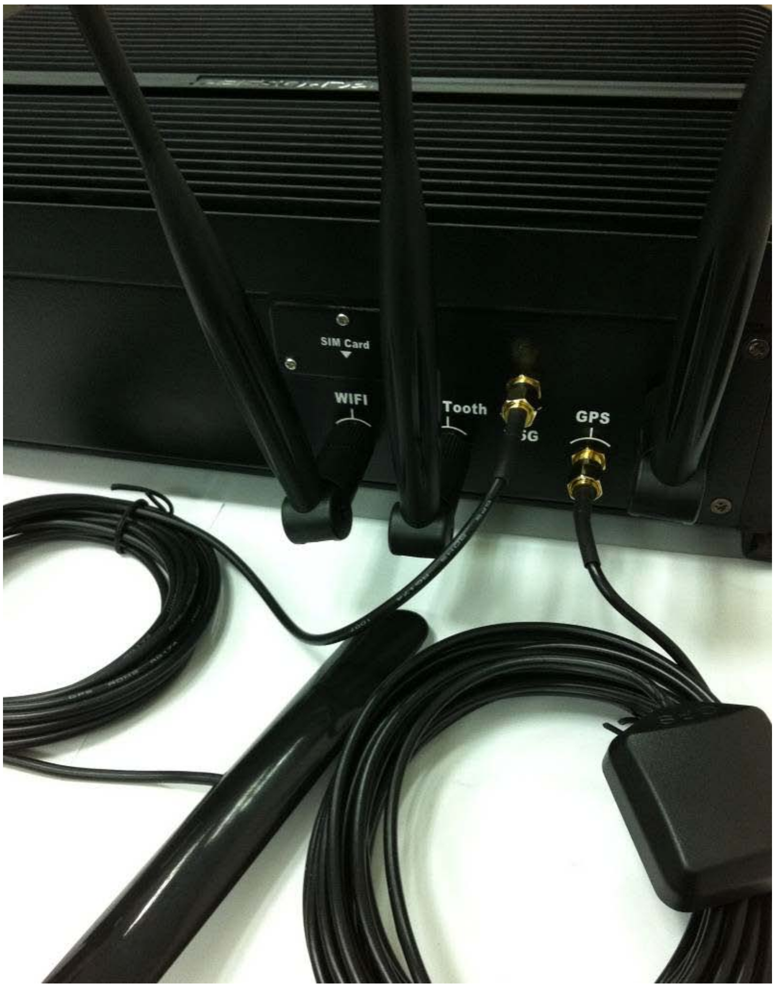
Finish.



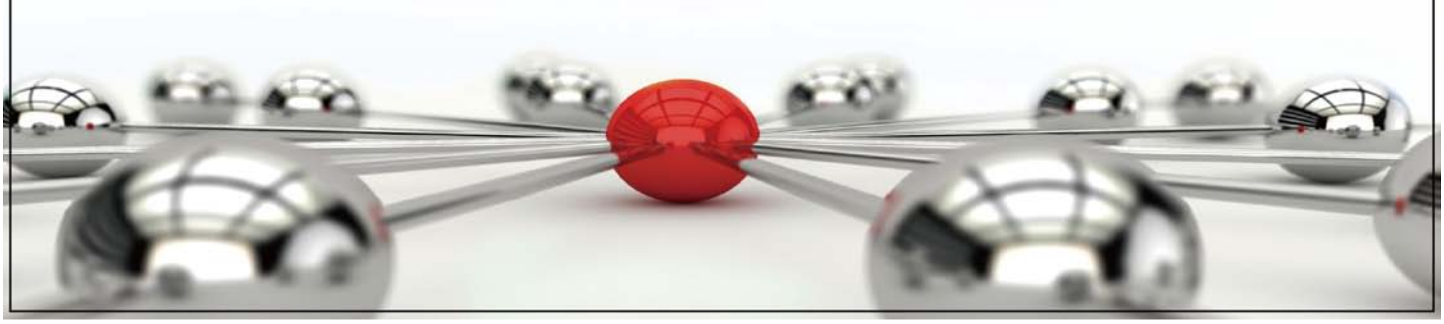
2.4 Antenna Installation



Tack out antenna from packing bag and install.



Board Guide



Chapter 1 Introduction

FleetPC-7 Series with Intel second generation Celeron and Core i5 / i7 processor is a multi-function In-Vehicle computer, which is suitable for using in all kind of applications. Besides basic I/O ports like VGA, LVDS, HDMI, DVI, Hybrid Multiple Display, USB, COM. LAN, and GPIO, FleetPC-7 has complete wireless solutions for GPS / 3.5G / WiFi / Bluetooth selection, Video capture, Swappable 2.5" HDD, DC output, Driver ID, and embedded CAN Bus function to allow micro-controllers and devices to communicate with each other in vehicle. In addition, AR-B6100 has intelligent power management function with software utility to monitor power status and control power sequence, and also compliant with most industry standards for in-vehicle usage including CE, FCC, and E-Mark.

1.1 Specifications

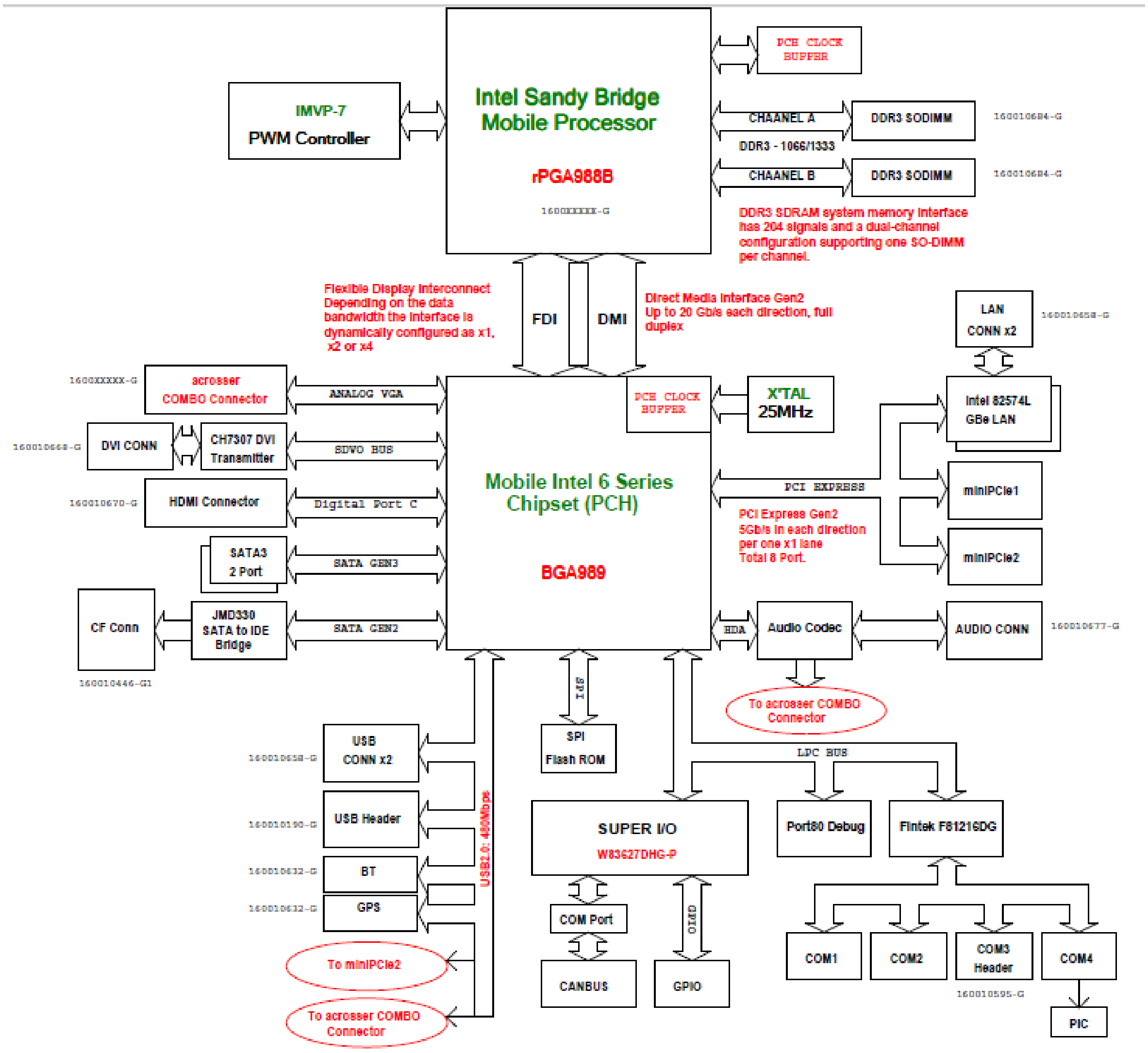
- Support Intel Celeron and Core i5/i7 CPU + HM65 chipset
- DDR3 SO-DIMM * 2, up to 8GB memory
- Display --- VGA + HDMI + DVI
- Combo connector --- VGA + Audio + USB + DC power
- SATA x 2 & SATA power x 2
- Swappable Anti-Shock 2.5" HDD bay x 1
- Intel GbE chip LAN x 2
- COM x 3 (2 x connector & 1 x pin header)
- CF type II socket x 1 / SIM slot x 1
- Audio connector (MIC & Line-out)
- Mini PCIe socket x 2 (Capable for WiFi / 3.5G)
- Flexible GPIO ports (8) & CAN bus
- Driver ID (Use I-Button) can certified driver,
- 9 ~ 32V DC input & customer define power management mode for ODM
- 12V DC 20W output connector for monitor

1.2 Package Contents

□ Check if the following items are included in the package.

- Quick Manual
- AR-B6100 board
- 1 x Software Utility CD

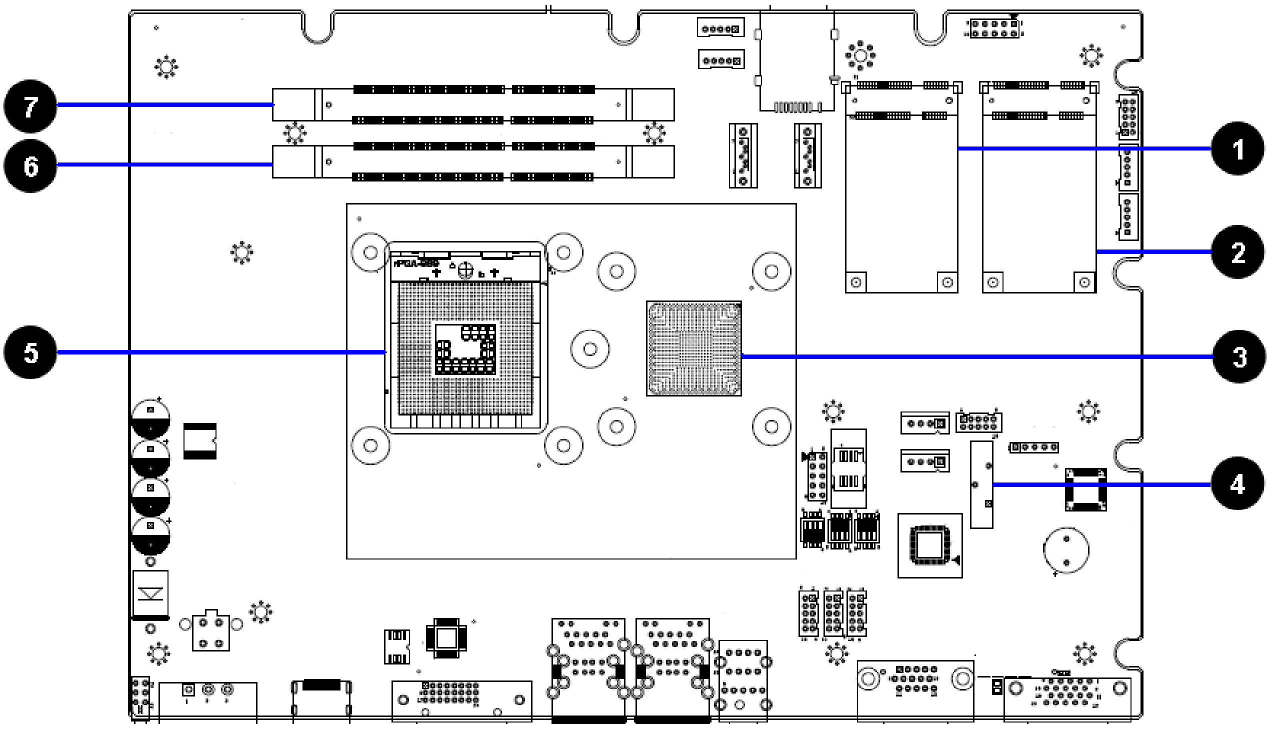
1.3 Block Diagram



Chapter 2 H/W Information

This chapter describes the installation of AR-B6100. At first, it shows the Function diagram and the layout of AR-B6100. It then describes the unpacking information which you should read carefully, as well as the jumper/switch settings for the AR-B6100 configuration.

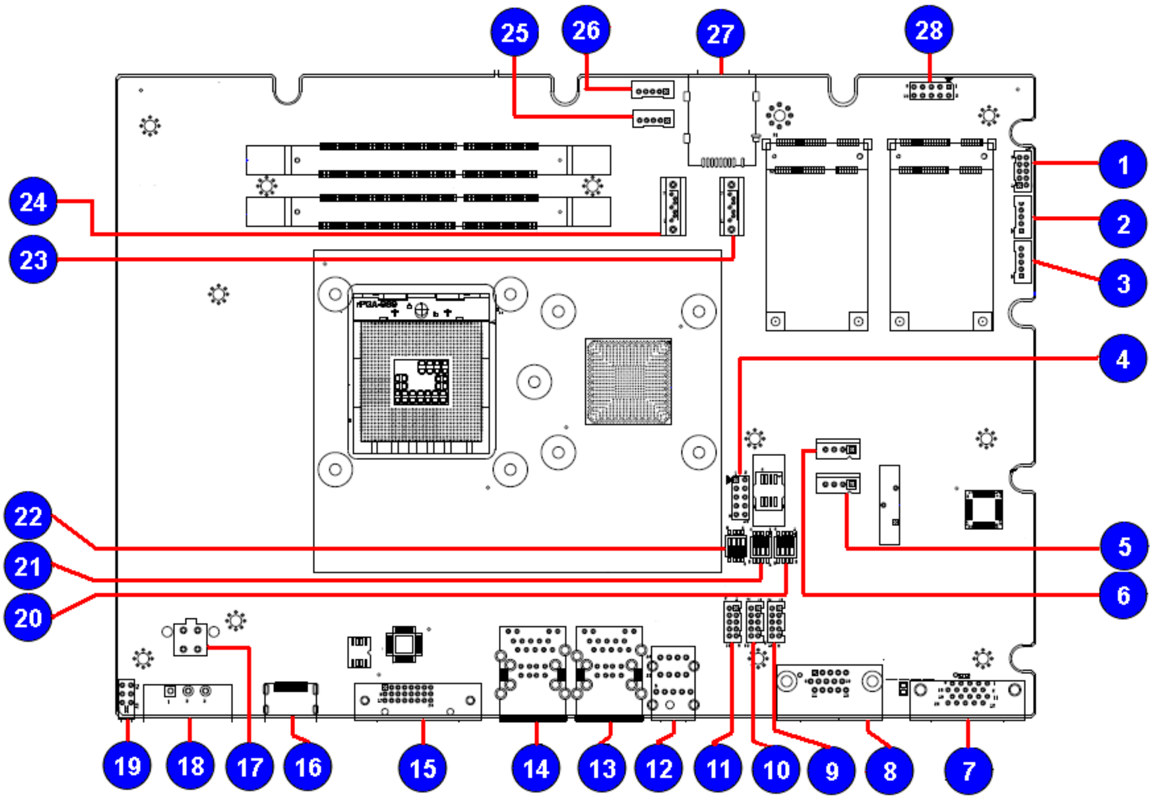
2.1 Mainboard illustration(Top Side)



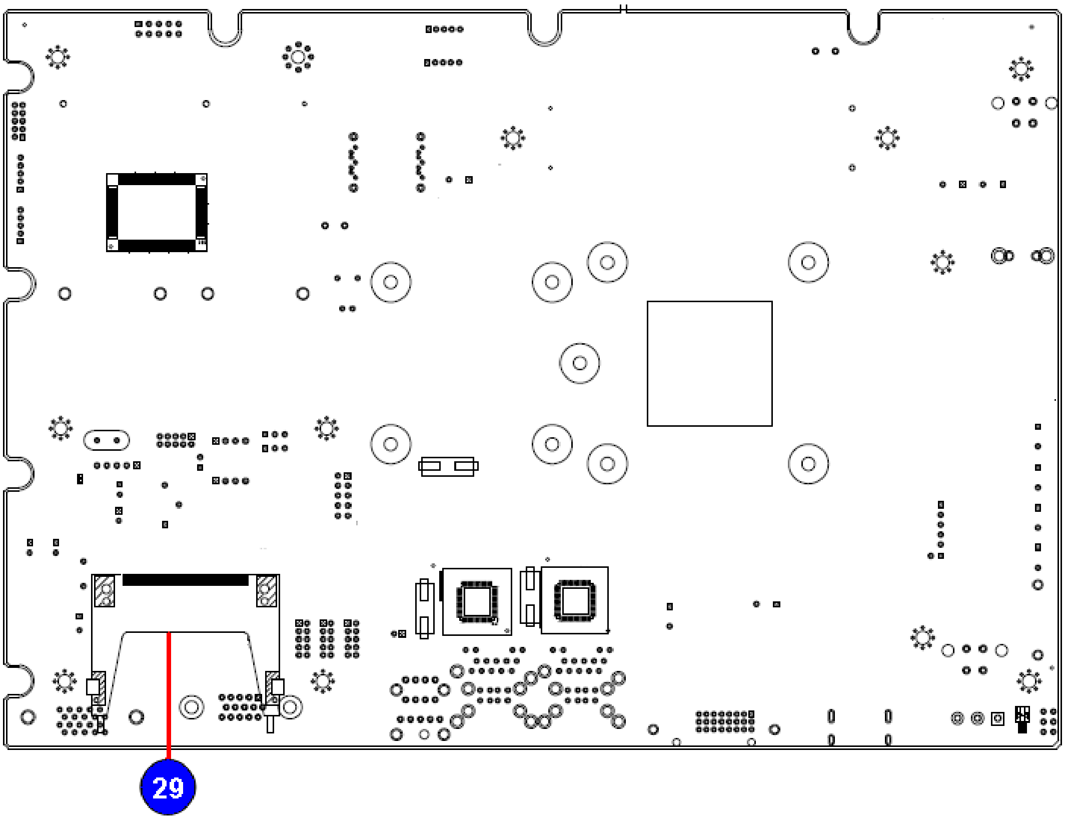
❶	*MINIPCIE1* Mini PCI-Express Slot1	❺	*rPGA988B CPU Socket*
❷	*MINIPCIE2* Mini PCI-Express Slot2	❻	*DIMM1* 204-Pin DDR3 Socket
❸	*Intel HM65 PCH*	❼	*DIMM2* 204-Pin DDR3 Socket
❹	*RTC1* System RTC battery socket		

2.2 Locations of IO ports & Jumper settings definition

TOP SIDE



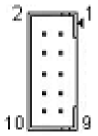
Bottom SIDE



1	**COM3** Pin Header for COM3 use RS-232 function	14	**CN2** RJ45 & USB 2 ports (USB2,USB3) Connector	27	**SIM1** For SIM Card Use.
2	**FP_USB1** Internal USB4 connector	15	**DVI1** DVI-D Connector.	28	**LPC1** LPC BUS Signal Header for Port-80 Debug Tools.
3	**FP_USB2** Internal USB5 connector	16	**HDMI1** HDMI Connector	29	**CF1** CF CARD SOCKET
4	**SPI1** For BIOS Firmware Update	17	**FUSE1** FUSE Connector.		
5	**SATA_PWR1** SATA Power Connector1.	18	**PWR1** 3 Pin External Power Input.		
6	**SATA_PWR2** SATA Power Connector2.	19	**LED2** 3 in 1 LED for Power ,HDD ,Status LED		
7	**COMBO1** COMBO Connector , Include Analog VGA , USB , Audio Signal.	20	**SW4** For RS-422,RS-485 function select		
8	**GPIO1** GPIO Connector , Include GPIO , CANBUS , I-Buttom Signal.	21	**SW3** For RS-422,RS-485 function select		
9	**COM2_485** Pin Header for COM2 use RS-422/485 function .	22	**SW2** For RS-422,RS-485 function select .		
10	**COM2** Pin Header for COM2 use RS-232 function	23	**SATA2** SATA device connector #2.		
11	**COM1** Pin Header for COM1 use RS-232 function .	24	**SATA1** SATA device connector #1.		
12	**AUDIO1** AUDIO connector.	25	**BT1** For Bluetooth Modular Connector.		
13	**CN1** RJ45 & USB 2 ports (USB0,USB1) Connector	26	**GPS1** For GPS Modular Connector		

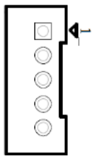
2.2.1 Connectors and Jumper Settings

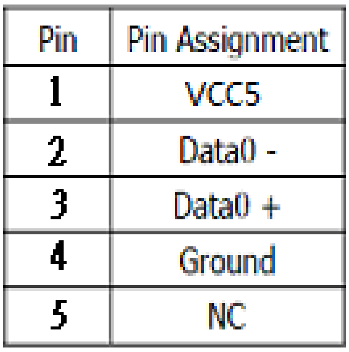
1. COM3 (For RS-232 Function)



Pin	SIGNAL
1	DSR
2	DCD
3	RTS
4	SIN
5	CTS
6	SOUT
7	RI
8	DTR
9	NC
10	GND

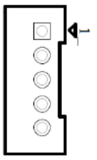
2. FP_USB1 Connector

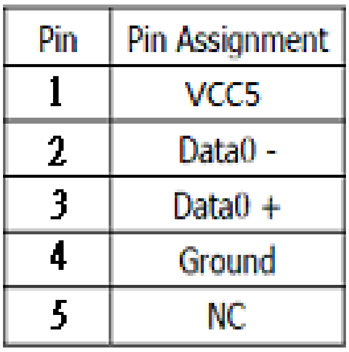




Pin	Pin Assignment
1	VCC5
2	Data0 -
3	Data0 +
4	Ground
5	NC

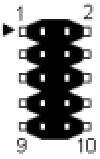
3. FP_USB2 Connector





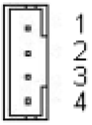
Pin	Pin Assignment
1	VCC5
2	Data0 -
3	Data0 +
4	Ground
5	NC

4. SPI1 (For BIOS FW Update)



Used BIOS Firmware Update Tools.

5. SATA_PWR1

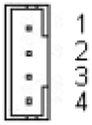


SATA_PWR1

SATA Device Power Connector

PIN	SIGNAL
1	+12V
2	GND
3	VCC3
4	VCC5

6. SATA_PWR2

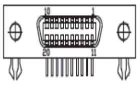


SATA_PWR2

SATA Device Power Connector

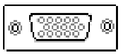
PIN	SIGNAL
1	+12V
2	GND
3	VCC3
4	VCC5

7. COMBO1 (COMBO Connector)



PIN	SIGNAL	PIN	SIGNAL
1	USB+	11	DDCCL
2	USB-	12	VCC12
3	GND	13	GND
4	VCC5	14	Audo_R
5	GND	15	GND
6	RED	16	NC
7	Green	17	Audo_L
8	Blue	18	NC
9	HSYNC	19	NC
10	VSYNC	20	DDCDA

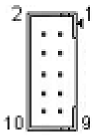
8. GPIO1 (GPIO Connector)



GPIO Piin Define:

PIN	SIGNAL	PIN	SIGNAL
1	GPO0	2	GPO1
3	GPO2	4	GPO3
5	GND	6	GND
7	CAN_H	8	CAN_L
9	GND	10	I-Button
11	GPI4	12	GPI5
13	GPI6	14	GPI7
15	VCC12A		

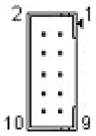
9. COM2_485 (For RS-422,RS-485 Function)



COM2_485: For RS-422,RS-485 Function

Pin	SIGNAL
1	NA
2	485_422_TX2+
3	NA
4	485_422_TX2-
5	422_RX2-
6	NA
7	422_RX2+
8	NA
9	422_485_SEL_L
10	GND

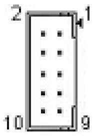
10. COM2 (For RS-232 Function)



COM2: For RS-232 Function

Pin	SIGNAL
1	DSR
2	DCD
3	RTS
4	SIN
5	CTS
6	SOUT
7	RI
8	DTR
9	NC
10	GND

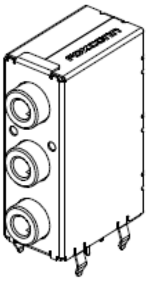
11. COM1(For RS-232 Function)



COM1: For RS-232 Function

Pin	SIGNAL
1	DSR
2	DCD
3	RTS
4	SIN
5	CTS
6	SOUT
7	RI
8	DTR
9	NC
10	GND

12. AUDIO1 (For Audio IN/Out & Remote Control)



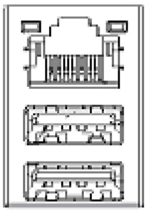
Audio Jack

BLUE：Remote

Green: Front Out

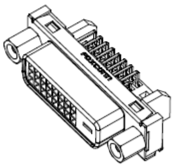
Pink: Mic in.

13, 14. CN1,CN2 (RJ45 x1& USB Port x2)



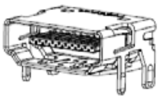
RJ45 Ethernet Connector with 2 ports of External USB Connector

15. DVI1 (DVI-D Connector)



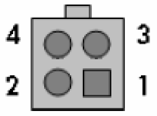
PIN	SIGNAL	PIN	SIGNAL
1	DATA2-	**2**	DATA2+
3	GND	**4**	NC
5	NC	**6**	DDC CLK
7	DDC Data	**8**	NC
9	DATA1-	**10**	DATA1+
11	GND	**12**	NC
13	NC	**14**	+5V
15	GND	**16**	HPD
17	DATA0-	**18**	DATA0+
19	GND	**20**	NC
21	NC	**22**	GND
23	CLK+	**24**	CLK-

16. HDMI1 (HDMI Connector)



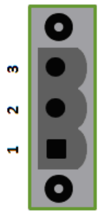
PIN	SIGNAL	PIN	SIGNAL
1	DATA2+	2	GND
3	DATA2-	4	DATA1+
5	GND	6	DATA1-
7	DATA0+	8	GND
9	DATA0-	10	CLK+
11	GND	12	CLK-
13	NC	14	NC
15	DDCCL	16	DDCDA
17	GND	18	+5V
19	HPD		

17. FUSE1 (FUSE connector)



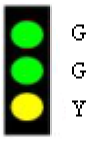
PIN	SIGNAL	PIN	SIGNAL
1	Power Out	2	Power Out
3	Power IN	4	Power IN

18. PWR1 (Power Input Terminal Block Connector)



PIN	DEFINE
1	Power IN
2	Ignition
3	GND

19. LED2

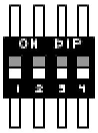


Green : Status LED

Green: HDD LED.

Yellow: Power ON LED

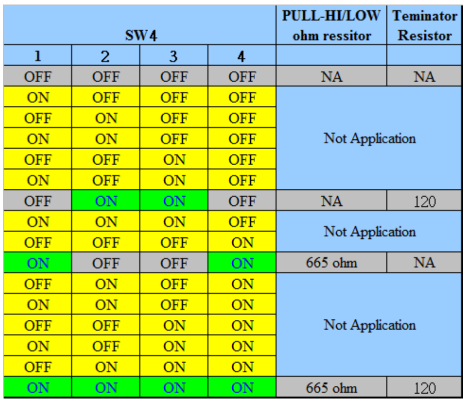
20. SW4 (RS-422 RX terminator resistor selection)



SW4 DIP Switch

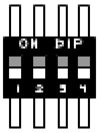
For RS-422 RX Terminator resistor selection)

(Default: all OFF)



SW4				PULL-HI/LOW ohm ressitor	Teminator Resistor
1	2	3	4		
OFF	OFF	OFF	OFF	NA	NA
ON	OFF	OFF	OFF	Not Application	
OFF	ON	OFF	OFF		
ON	ON	OFF	OFF		
OFF	OFF	ON	OFF		
ON	OFF	ON	OFF		
OFF	ON	ON	OFF	NA	120
ON	ON	ON	OFF	Not Application	
OFF	OFF	OFF	ON		
ON	OFF	OFF	ON	665 ohm	NA
OFF	ON	OFF	ON	Not Application	
ON	ON	OFF	ON		
OFF	OFF	ON	ON		
ON	OFF	ON	ON		
OFF	ON	ON	ON		
ON	ON	ON	ON	665 ohm	120

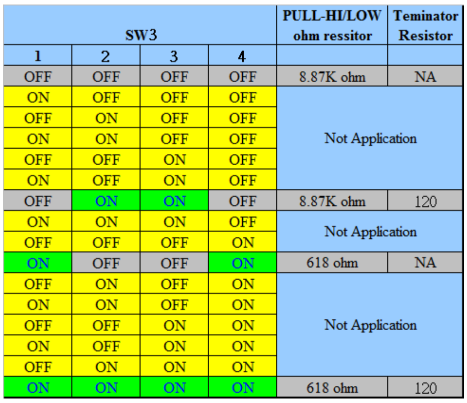
21. SW3 (RS-422/485 TX Terminator resistor selection)



SW3 DIP Switch

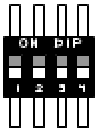
For RS-422/485 TX Terminator resistor selection)

(Default: all OFF)



SW3				PULL-HI/LOW ohm ressitor	Teminator Resistor
1	2	3	4		
OFF	OFF	OFF	OFF	8.87K ohm	NA
ON	OFF	OFF	OFF	Not Application	
OFF	ON	OFF	OFF		
ON	ON	OFF	OFF		
OFF	OFF	ON	OFF		
ON	OFF	ON	OFF		
OFF	ON	ON	OFF	8.87K ohm	120
ON	ON	ON	OFF	Not Application	
OFF	OFF	OFF	ON		
ON	OFF	OFF	ON	618 ohm	NA
OFF	ON	OFF	ON	Not Application	
ON	ON	OFF	ON		
OFF	OFF	ON	ON		
ON	OFF	ON	ON		
OFF	ON	ON	ON		
ON	ON	ON	ON	618 ohm	120

22. SW2 (RS-422,RS-485 function select)



SW2 DIP Switch

For RS-422,RS-485 Function select(Default: All OFF For RS-232)

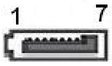
RS-422 setting:

1	OFF
2	ON
3	OFF
4	ON

RS-485 setting:

1	ON
2	ON
3	OFF
4	ON

23, 24. SATA2, SATA1 (SATA device connector #2 and #1)

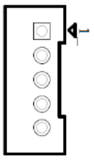


To connect SATA device:

1.Attach either end of the signal cable to the SATA connector on motherboard. Attach the other end to the SATA device.

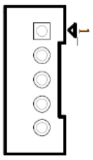
2. Attach the SATA power cable to the SATA device and connect the other end from the power supply.

25. BT1 (BLUETOOTH connector)



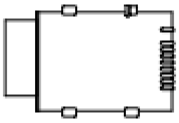
PIN	DEFINE
1	NC
2	USB_D-
3	USB_D+
4	GND
5	VCC3

26. GPS1 (GPS connector)



PIN	DEFINE
1	NC
2	USB_D-
3	USB_D+
4	GND
5	VCC3

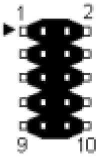
27. SIM1 (SIM CARD Socket)



SIM Card Holder

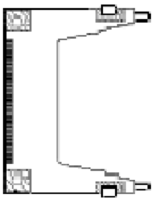
Connects to 3.5G Cell phone SIM Card.

28. LPC1 (LPC BUS Signal Header for Port-80 Debug Tools)



Pin	SIGNAL
1	33MHz Clock
2	LAD1
3	Reset#
4	LAD0
5	LFRAME#
6	VCC3
7	LAD3
8	GND
9	LAD2
10	GND

29. CF1 (CF CARD Socket)



Supports Compact Flash Card TYPE I/II

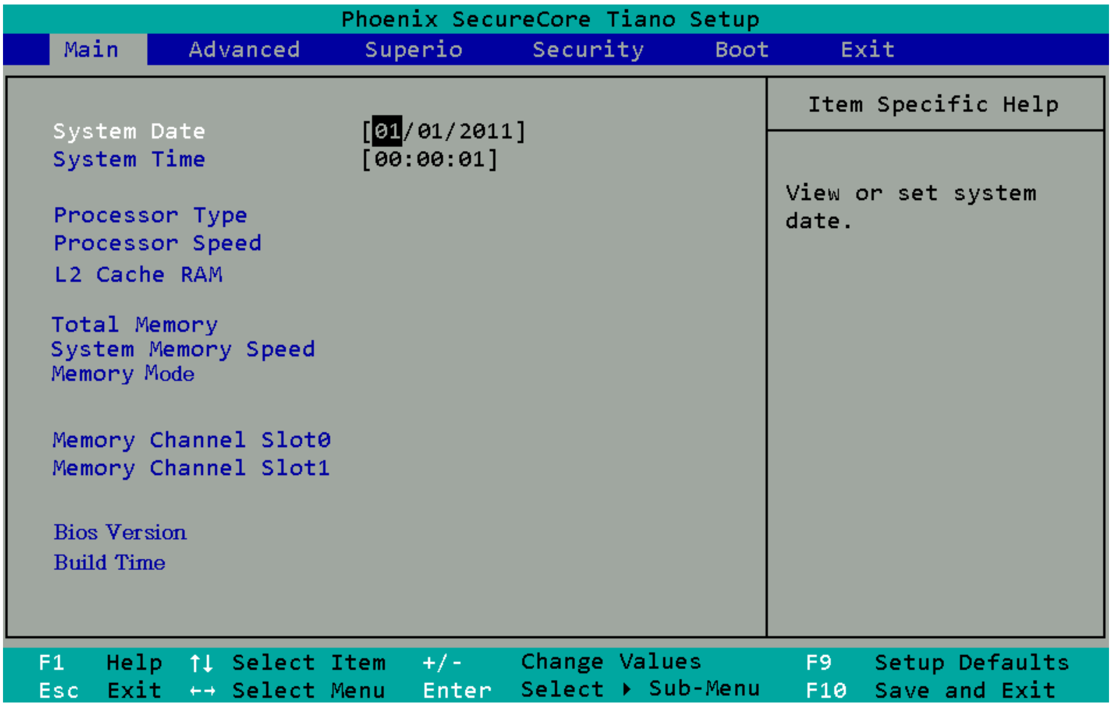
Chapter 3 BIOS Settings

This chapter describes the BIOS menu displays and explains how to perform common tasks needed to get the system up and running. It also gives detailed explanation of the elements found in each of the BIOS menus. The following topics are covered:

- Main Setup
- Advanced Chipset Setup
- SuperIO Setup
- Security Setup
- Boot Setup
- Exit Setup

3.1 Main Setup

Once you enter the Phoenix BIOS™ CMOS Setup Utility, the Main Menu will appear on the screen. Use the arrow keys to highlight the item and then use the <Pg Up> <Pg Dn> keys to select the value you want in each item.



```
System Date                 [01/01/2011]
System Time                 [00:00:01]

Processor Type
Processor Speed
L2 Cache RAM

Total Memory
System Memory Speed
Memory Mode

Memory Channel Slot0
Memory Channel Slot1

Bios Version
Build Time
```

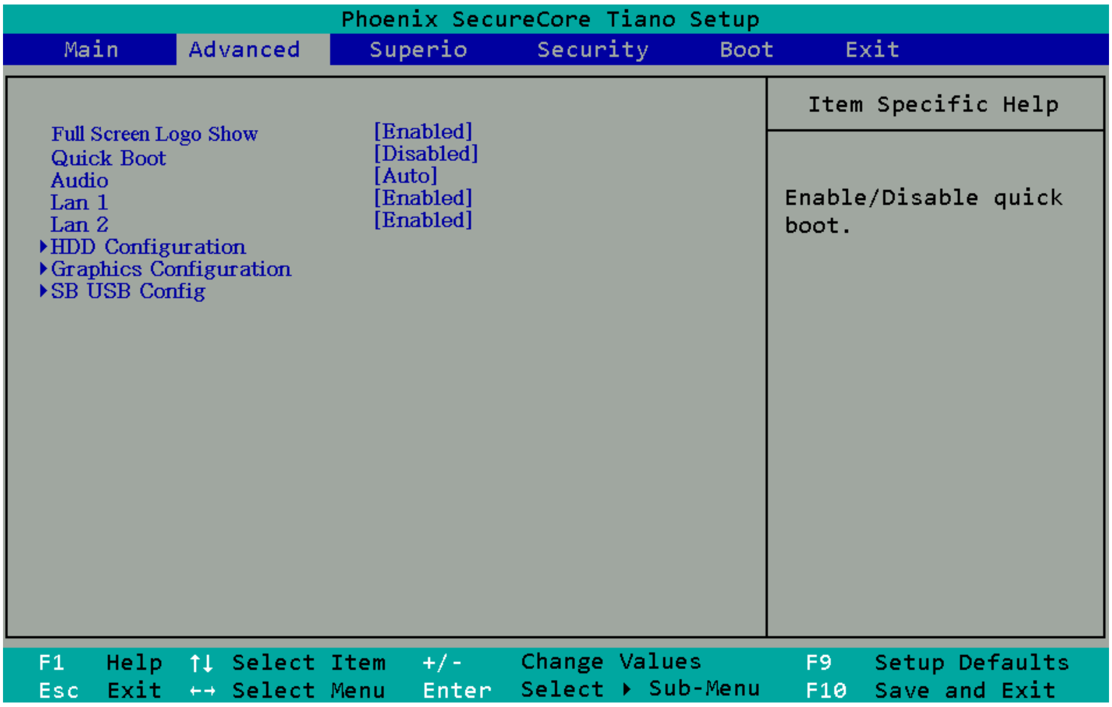
View or set system date.

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

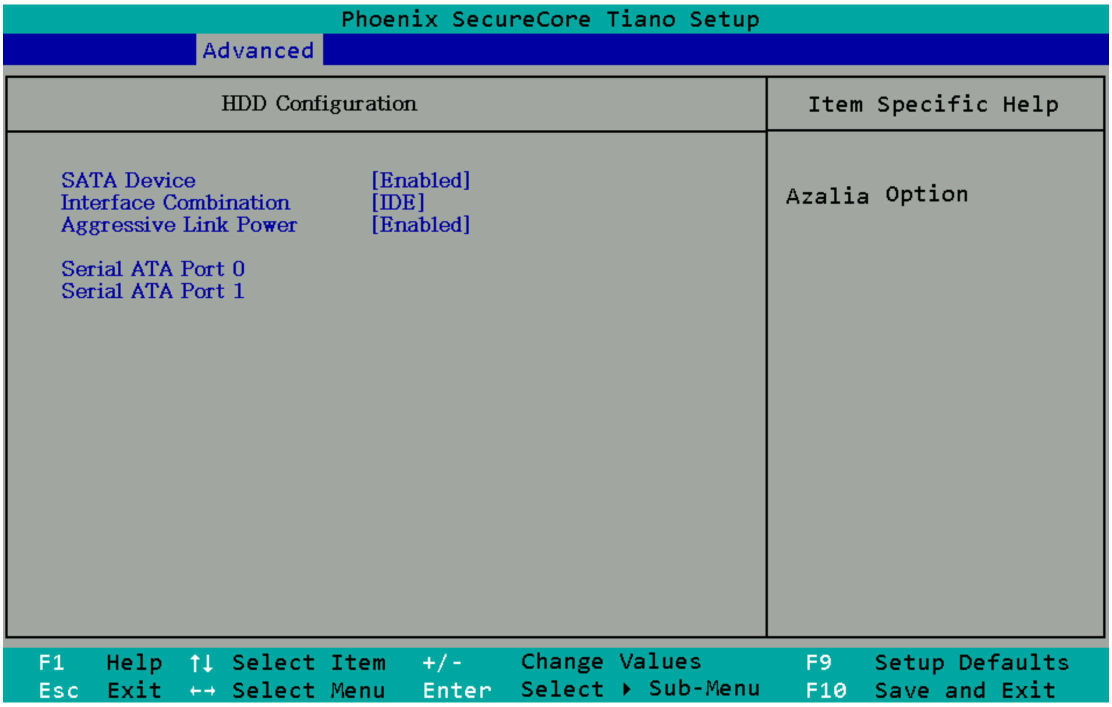
Option	Choice	Description
System Date	N/A	Set the system date. Note that the 'Day' automatically changes when you set the date
System Time	N/A	Set the system time.
Processor Type	N/A	This item displays the CPU Type
Processor Speed	N/A	This item displays the CPU Speed
L2 Cache Ram	N/A	This item displays the L2 ache memory size
Total Memory	N/A	This item displays the memory size that used.
System Memory Speed	N/A	This item displays the memory speed.

Memory Mode	N/A	This item displays the memory mode.
Memory Channel slot 0	N/A	This item displays the memory size that used On slot 0.
Memory Channel slot 1	N/A	This item displays the memory size that used On slot 0.
BIOS Version	N/A	This item displays BIOS's Version
Build Time	N/A	This item displays the building time of BIOS.

3.2 Advanced Chipset Setup



```
                    Phoenix SecureCore Tiano Setup
   Main      Advanced     Superio     Security     Boot     Exit

   Full Screen Logo Show      [Enabled]             Item Specific Help
   Quick Boot                 [Disabled]
   Audio                      [Auto]             Enable/Disable quick
   Lan 1                      [Enabled]          boot.
   Lan 2                      [Enabled]
  ▸HDD Configuration
  ▸Graphics Configuration
  ▸SB USB Config

 F1   Help  ↑↓ Select Item   +/-    Change Values       F9   Setup Defaults
 Esc  Exit  ↔  Select Menu   Enter  Select ▸ Sub-Menu   F10  Save and Exit
```


Phoenix SecureCore Tiano Setup

Advanced

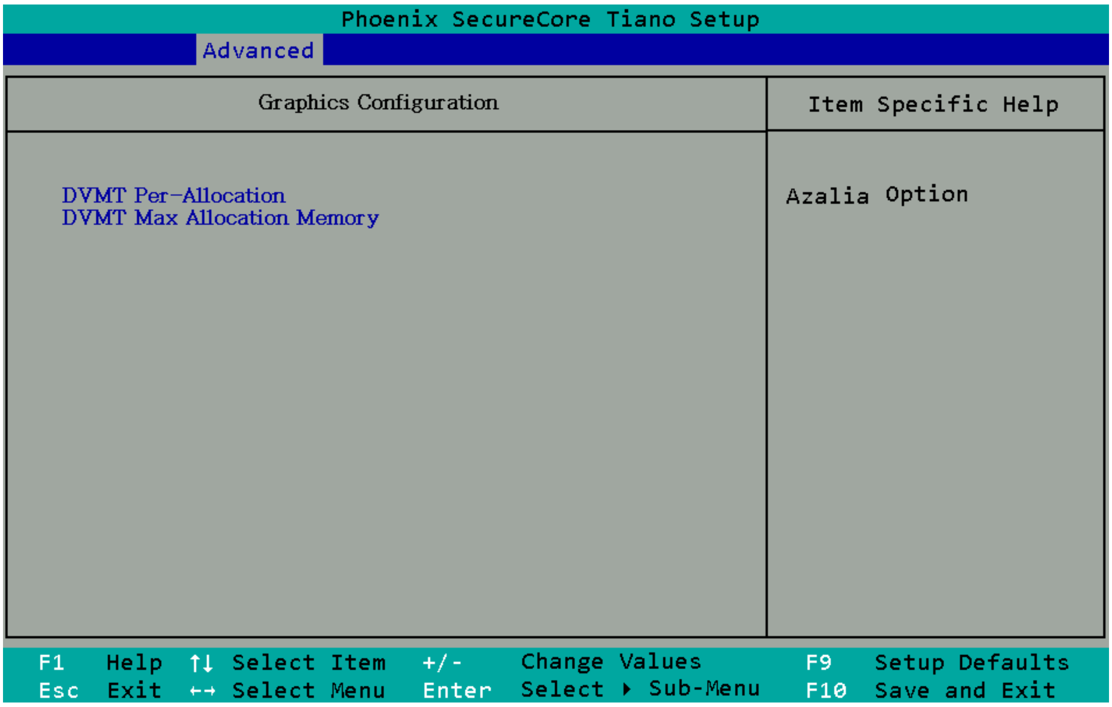
HDD Configuration

SATA Device	[Enabled]
Interface Combination	[IDE]
Aggressive Link Power	[Enabled]

Serial ATA Port 0
Serial ATA Port 1

Item Specific Help

Azalia Option


Phoenix SecureCore Tiano Setup

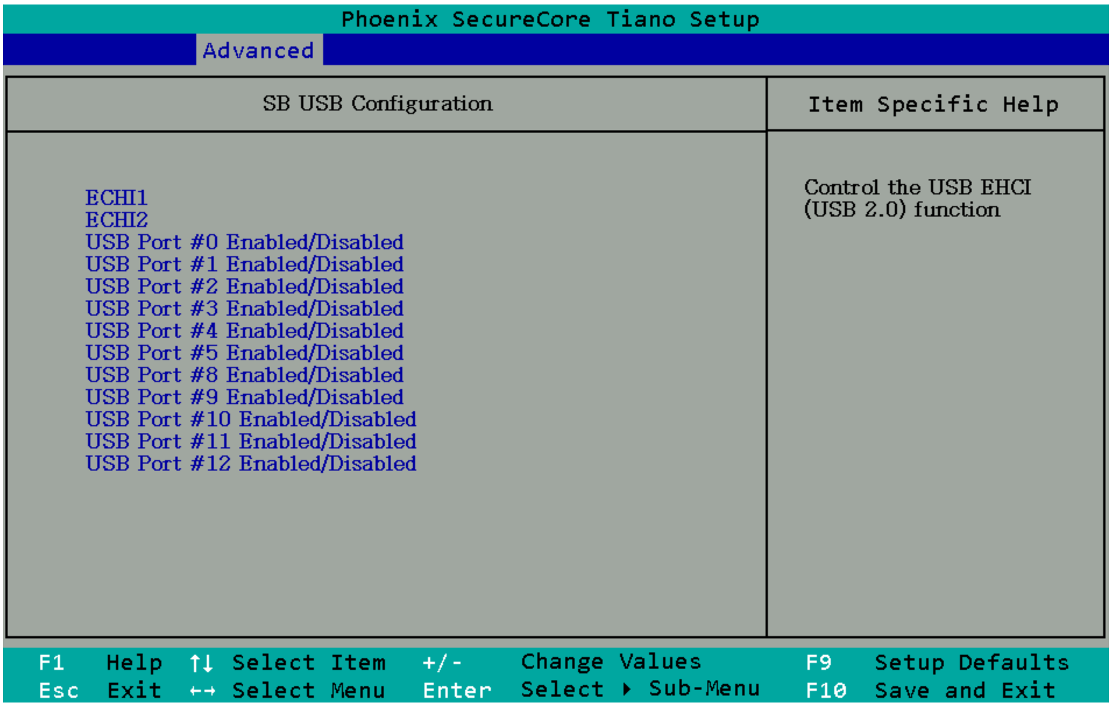
Advanced

Graphics Configuration

DVMT Per-Allocation
DVMT Max Allocation Memory

Item Specific Help

Azalia Option

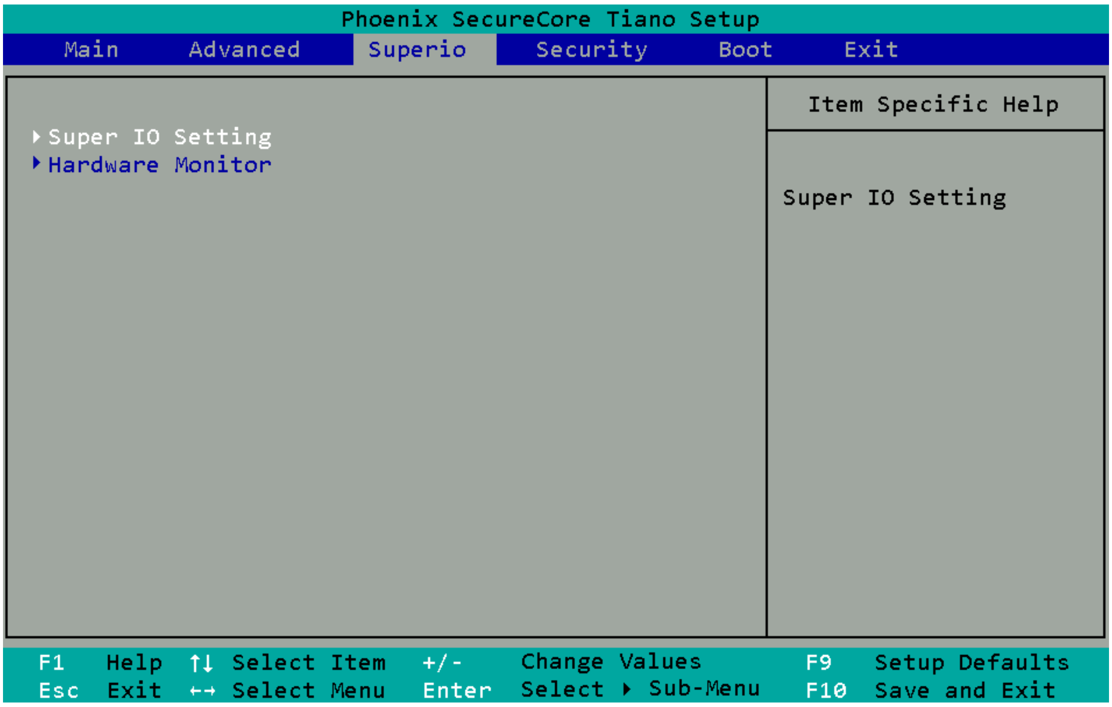


```
                    Phoenix SecureCore Tiano Setup
         Advanced
            SB USB Configuration                 Item Specific Help

     ECHI1                                     Control the USB EHCI
     ECHI2                                     (USB 2.0) function
     USB Port #0 Enabled/Disabled
     USB Port #1 Enabled/Disabled
     USB Port #2 Enabled/Disabled
     USB Port #3 Enabled/Disabled
     USB Port #4 Enabled/Disabled
     USB Port #5 Enabled/Disabled
     USB Port #8 Enabled/Disabled
     USB Port #9 Enabled/Disabled
     USB Port #10 Enabled/Disabled
     USB Port #11 Enabled/Disabled
     USB Port #12 Enabled/Disabled

 F1   Help  ↑↓ Select Item   +/-    Change Values      F9   Setup Defaults
 Esc  Exit  ←→ Select Menu   Enter  Select ▸ Sub-Menu  F10  Save and Exit
```

Option	Choice	Description
Full Screen Logo Show	Enabled Disabled	Displays the full screen logo upon BIOS booting
Quick Boot	Enabled Disabled	Allows the system to skip certain tests while booting. This will decrease the time needed to boot the system.
Audio	Auto Enable Disable	Control detection of the Azalia device.
Lan 1	Enabled Disabled	Control the Lan 1 port.
Lan 2	Enabled Disabled	Control the Lan2 port.
Sata Device	Enabled Disabled	Enabled E nables onboard SATA controller Disabled T urn off onboard SATA controller

Interface Combination	AHCI IDE	Select SATA mode.
Aggressive Link Power	Enabled Disabled	Enabled E nables onboard SATA power pin. Disabled T urn off onboard SATA power pin.
Serial ATA port 0	N/A	Show HDD information.
Serial ATA port 1	N/A	Show HDD information.
DVMT Pre-allocation	32MB 64MB 128MB	How much memory you want to point to the graphics card
DVMT Max allocation Memory	128MB 256MB MAX	Points up how much memory to the graphics card
EHCI 1,2	Enabled Disabled	Control the USB 2.0 functions.
USB Port #0~12 Enable/Disable	Enabled Disabled	Enable/Disable USB Ports.

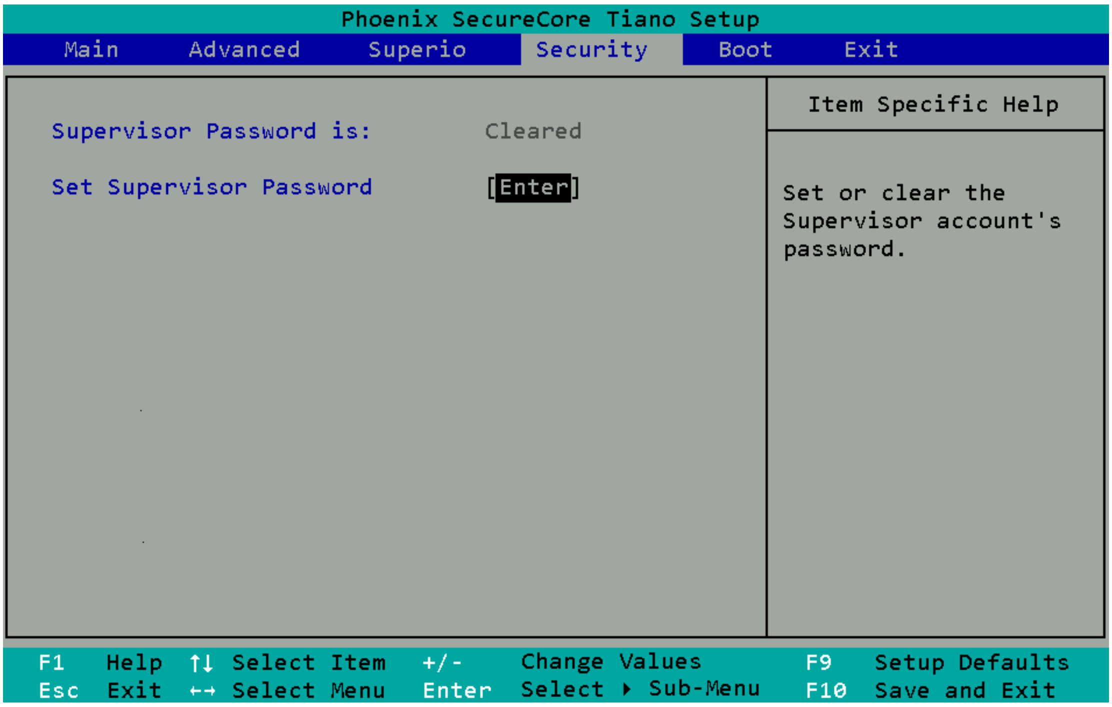
3.3 Superio Setup



Option	Choice	Description
Com_1 4F8/5	Enabled Disabled	Enable or Disable the com port function.
Com_2 4E8/7	Enabled Disabled	Enable or Disable the com port function.
CPU Temperature	N/A	These read-only fields show the functions of the hardware thermal sensor by CPU thermal diode that monitors the chip blocks to ensure a stable system.
System Temperature	N/A	Show you the current system temperature.
CPU VCore	N/A	Show you the voltage of Vcore.
+12V	N/A	Voltage of 12V on the mother board

+5V	N/A	Voltage of 5V on the mother board
+3.3V	N/A	Voltage of 3.3V on the mother board
VBAT	N/A	Voltage of Battery on the mother board

3.4 Security Setup

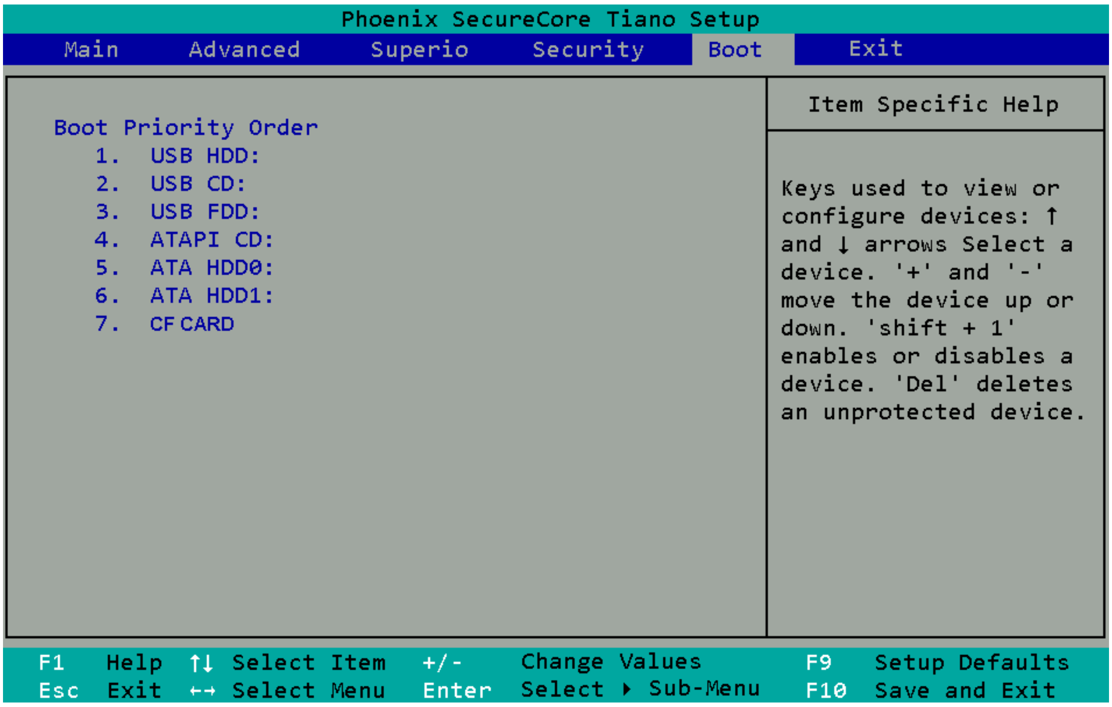


```
                         Phoenix SecureCore Tiano Setup
    Main      Advanced     Superio     Security     Boot      Exit

  Supervisor Password is:           Cleared          Item Specific Help

  Set Supervisor Password           [Enter]         Set or clear the
                                                    Supervisor account's
                                                    password.

 F1   Help  ↑↓ Select Item   +/-    Change Values        F9    Setup Defaults
 Esc  Exit  ↔  Select Menu   Enter  Select ▸ Sub-Menu    F10   Save and Exit
```

Option	Choice	Description
Supervisor Password is	N/A	The BIOS attempts to load the operating system from the devices in the sequence selected in these items.
Set Supervisor Password	Pressing <Enter> on this item for confirmation: **ENTER PASSWORD:**	When a password has been enabled, you will be prompted to enter your password every time you try to enter Setup. This prevents unauthorized persons from changing any part of your system configuration. 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. 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.

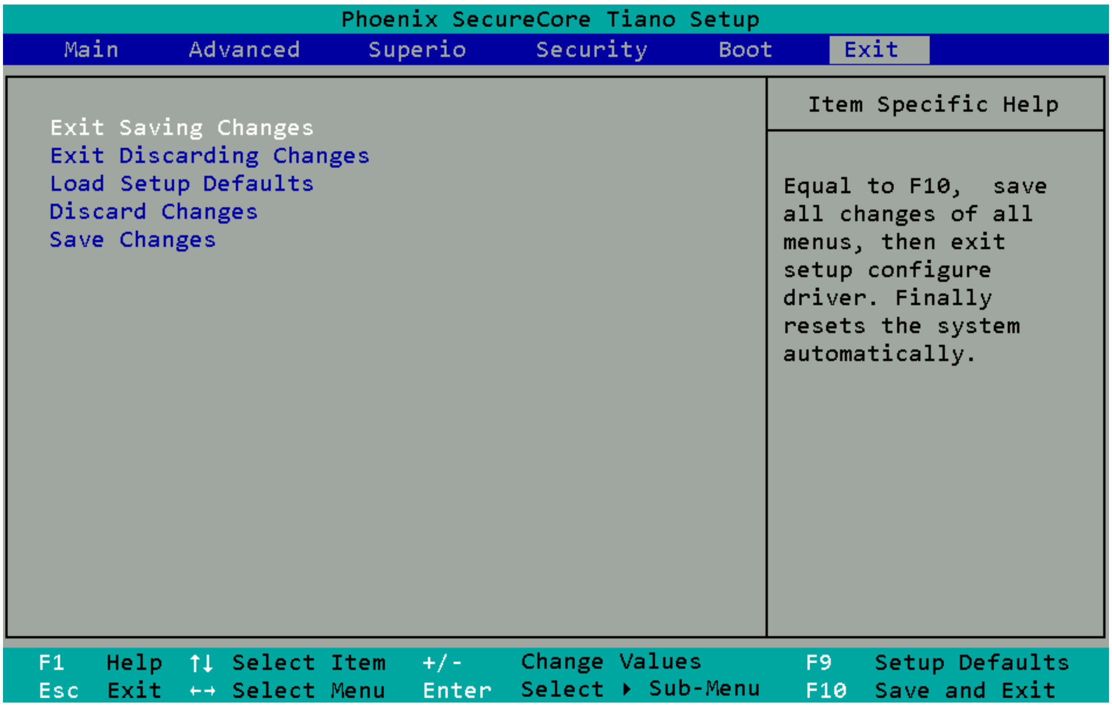
3.5 Boot setup

Choice boot priority.



```
                 Phoenix SecureCore Tiano Setup
   Main     Advanced     Superio     Security     Boot      Exit

   Boot Priority Order                          Item Specific Help
      1.  USB HDD:
      2.  USB CD:                             Keys used to view or
      3.  USB FDD:                            configure devices: ↑
      4.  ATAPI CD:                           and ↓ arrows Select a
      5.  ATA HDD0:                           device. '+' and '-'
      6.  ATA HDD1:                           move the device up or
      7.  CF CARD                             down. 'shift + 1'
                                              enables or disables a
                                              device. 'Del' deletes
                                              an unprotected device.

 F1   Help  ↑↓ Select Item   +/-     Change Values          F9    Setup Defaults
 Esc  Exit  ↔ Select Menu    Enter   Select ▸ Sub-Menu      F10   Save and Exit
```

3.6 Exit Setup

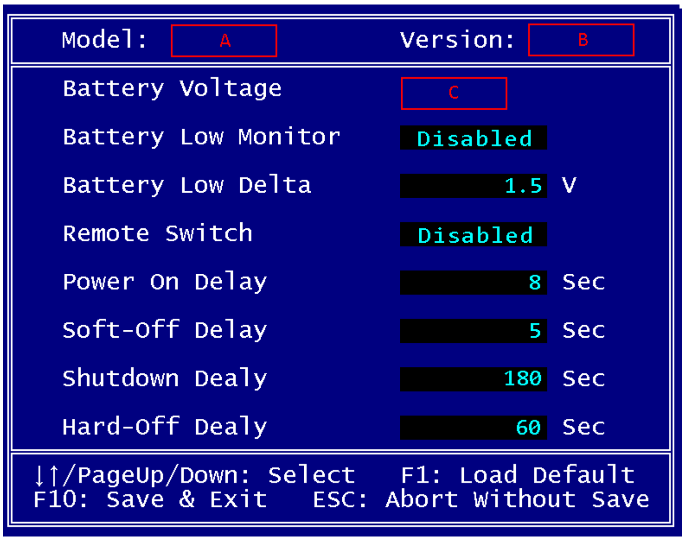


```
                     Phoenix SecureCore Tiano Setup
   Main     Advanced     Superio     Security     Boot     Exit

     Exit Saving Changes                          Item Specific Help
     Exit Discarding Changes
     Load Setup Defaults                        Equal to F10,  save
     Discard Changes                            all changes of all
     Save Changes                               menus, then exit
                                                setup configure
                                                driver. Finally
                                                resets the system
                                                automatically.

 F1   Help  ↑↓ Select Item  +/-    Change Values       F9   Setup Defaults
 Esc  Exit  ←→ Select Menu  Enter  Select ▸ Sub-Menu   F10  Save and Exit
```

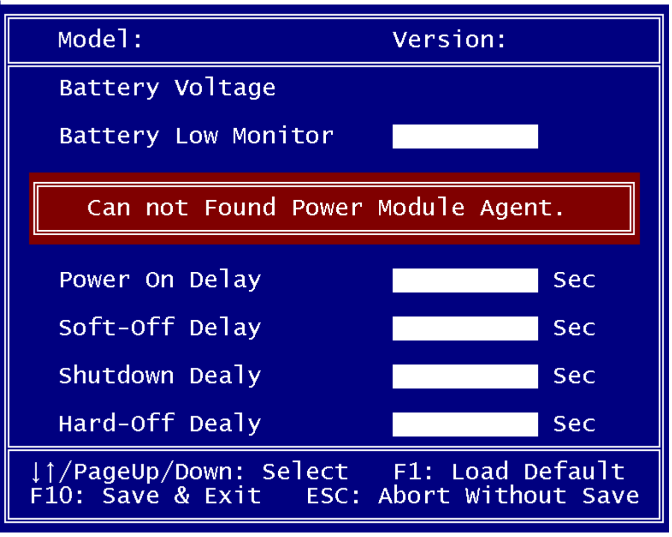
option	Choice	Description
Exit Saving Changes	Pressing <Enter> on this item for confirmation:	Exit BIOS Setup and Save Changes BIOS Setting.
Exit Discarding Changes	Pressing <Enter> on this item for confirmation:	Exit BIOS Setup and Without Save Changes BIOS Setting.

Load Setup Defaults	When you press <Enter> on this item you get a confirmation dialog box with a message like this:	Press 'Y' to load the default values that are factory-set for optimal-performance system operations.
Discard Changes	Pressing <Enter> on this item for confirmation:	N/A
Save Changes	Pressing <Enter> on this item for confirmation:	Save Changes BIOS Setting but without exit BIOS Setup.

Appendix



```
   Model:  A               Version:  B

   Battery Voltage          C
   Battery Low Monitor     Disabled
   Battery Low Delta           1.5  V
   Remote Switch           Disabled
   Power On Delay                8  Sec
   Soft-Off Delay                5  Sec
   Shutdown Dealy              180  Sec
   Hard-Off Dealy               60  Sec

 ↓↑/PageUp/Down: Select    F1: Load Default
 F10: Save & Exit    ESC: Abort Without Save
```



```
   Model:                      Version:

   Battery Voltage
   Battery Low Monitor

     Can not Found Power Module Agent.

   Power On Delay                      Sec
   Soft-Off Delay                      Sec
   Shutdown Dealy                      Sec
   Hard-Off Dealy                      Sec

 ↓↑/PageUp/Down: Select    F1: Load Default
 F10: Save & Exit    ESC: Abort Without Save
```

a.Power Sub-System Parameter Setting

Power subsystem parameters can be set by BIOS or Power Management Utility or Application Program through API. All parameters shall be able to read through the serial port of platform.

1. Remote Switch:
   - A. Remote Switch Disabled (Ignition only)
   - B. Remote Switch Enabled (Ignition + Remote Switch)
   - C. Default setting: ***Disable***

2. Power On Delay:
   - A. Range: 8 second to 60 seconds with 1 second increment
   - B. Default Setting: ***8 seconds***

3. Soft Off Delay:
   - A. Range: 0 second to 3600 seconds with 1 second increment
   - B. Default Setting: ***5 seconds***

4. Shutdown Delay:
   - A. Range: 120 seconds to 3600 seconds with 1 second increment
   - B. Default Setting: **180 seconds**

5. Hard Off delay:
   - A. Range: 0 second to 3600 seconds with 1 second increment
   - B. Default Setting: ***60 seconds***
6. Battery Low monitor
   - A. Enable or disable: If it is disable, the battery low monitor will not prohibit power on or shut down platform due to battery low. Customers need to confirm their power supply can support sufficient power for our system.
   - B. Default: Disable

7. Battery low delta :
   - A. Battery low delta is a number in unit of Volt to determine the Battery low voltage.
   - B. Battery low voltage = Standard battery voltage (12V or 24V) – Delta. For example, if delta is 2 Volts for a 12V vehicle, the Battery low voltage is 10 Volts.
   - C. Range: 0.5V to 3.0V with 0.5V increment.
     Default Setting: ***1.5V***
8. Area “A” for Power Sub-System Model Name
9. Area “B” for Power Sub-System Firmware Revision
10. Area “C” for Power Sub-System Current Battery Voltage

b. Power Sub-System Setup Manual:

Setup manual can be activate during BIOS POST by pressing a hot key “F4” on the keyboard. The setup manual is used for power subsystem parameter setting. The changes will be stored into power subsystem PIC controller after pressing “F10” or remain unchanged by pressing “ESC” key.

c. Reset Power Sub-System Parameters

When PIC detected the parameter reset pins are shorted or Setup manual pressing a hot key “F1”, all following parameters will be reset to all their default setting

1. Remote Switch: Disable
2. Power On Delay: 8 Sec
3. Soft Off Delay: 5 Sec
4. Shutdown delay: 180 Sec
5. Hard Off delay: 60 Sec
6. Battery Low monitor : Disable
7. Battery low delta : 1.5 V

Chapter 4 Function Description

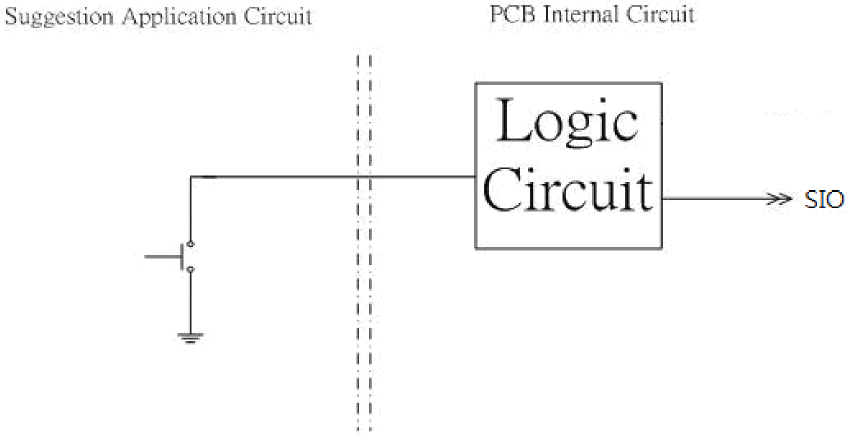
4.1 DC Power input connection

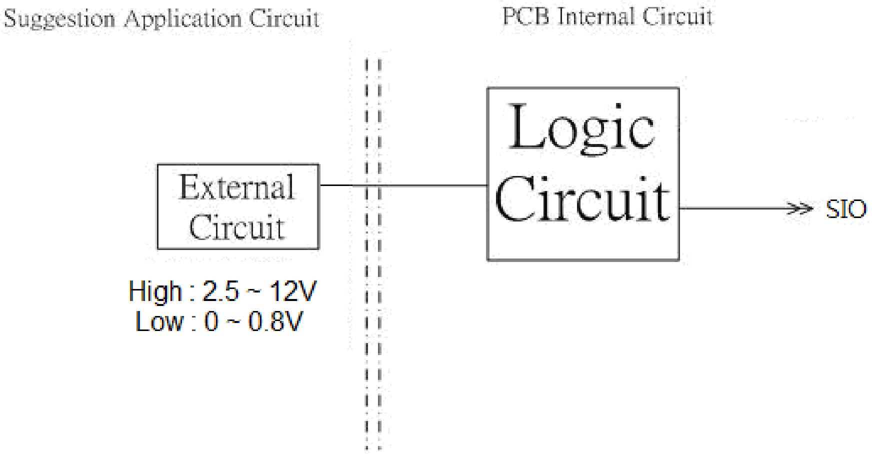
AR-B6100 needs +9~32V to power the board.

4.2 Digital Inputs

There are 4 clamped diode protection digital inputs on GPIO1 connector. You can read the status of any input through the software API. These digital inputs are general purpose input. You can define their purpose for any digital input function. The detailed information please refers to Software Programming Guide for how to use the API.

Following diagrams state how to connect the digital inputs to devices on the embedded system.

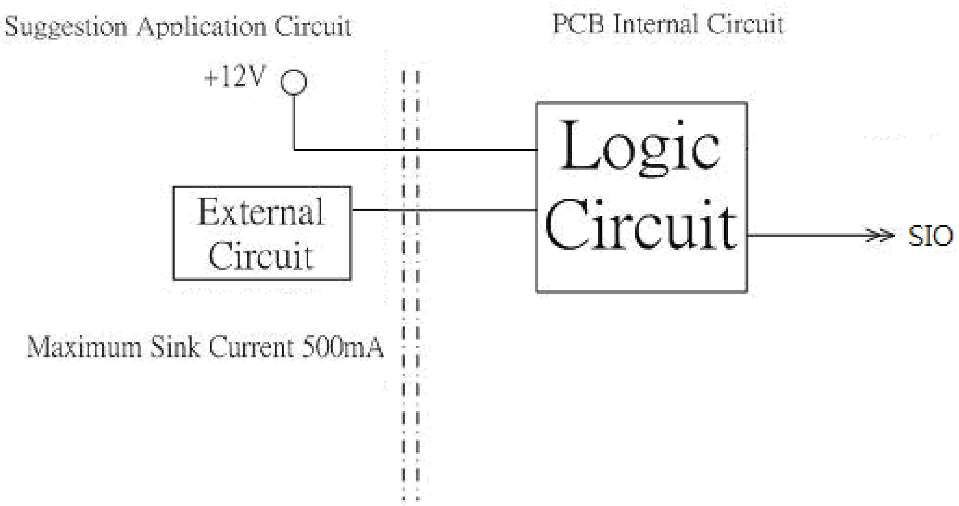




4.3 Digital Outputs

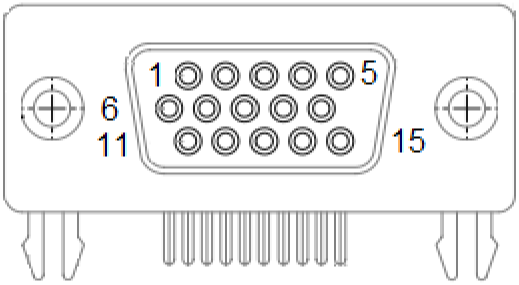
There are 4 clamped diode protection digital outputs on GPIO1 connector. You can control the output status of these digital outputs through the software API. The four digital outputs are capable sink maximum 500 mA current for each channel and maximum output voltage is 12V. The output reference voltage of device, please connect to GPIO #VCC12V(Pin15). These digital outputs are general purpose outputs. The detailed information please refers to Software Programming Guide for how to use the API.

Following diagrams state how to connect the digital outputs to devices on the embedded system.



GPIO Pin Define:

PIN	SIGNAL	PIN	SIGNAL
1	GPO0	2	GPO1
3	GPO2	4	GPO3
5	GND	6	GND
7	CAN_H	8	CAN_L
9	GND	10	I-Button
11	GPI4	12	GPI5
13	GPI6	14	GPI7
15	VCC12A		

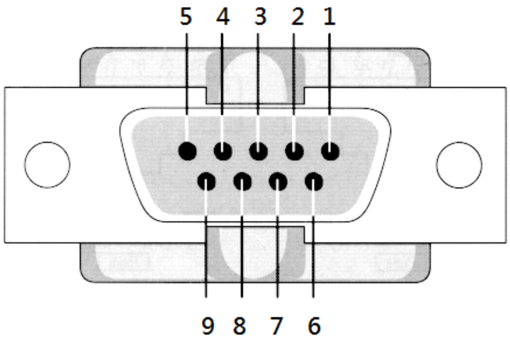


4.4 Watchdog Timer

If you set a watchdog timer, you can use it to reset the system when system hangs up due to hardware issue. After you set the watchdog timer, the software shall re-set the timer to re-start a new cycle before it time-out. Please refer to Chapter 6 Software Installation and Programming Guide for how to set the watchdog timer.

4.5 RS-232 Ports

The COM1\COM2\COM3 is connected through a cable (Pin Header). Users need to plug into RS-232 or RS-422/485 connector. Please refer to SW2, SW3 and SW4 setting. The following diagram is their pin definition and signal.



Pin number	RS-232 male
1	DCD
2	TXD
3	RXD
4	DSR
5	GND
6	DTR
7	CTS
8	RTS
9	RI

COM1, COM2, COM3: For RS-232 Function

Pin	SIGNAL
1	DSR
2	DCD
3	RTS
4	SIN
5	CTS
6	SOUT
7	RI
8	DTR
9	NC
10	GND

COM2_485: For RS-422, RS-485

Pin	SIGNAL
1	NA
2	485_422_TX2+
3	NA
4	485_422_TX2-
5	422_RX2-
6	NA
7	422_RX2+
8	NA
9	422_485_SEL_L
10	GND

4.6 Serial ATA (SATA)

There are 2 SATA 2.5 ports on the AR-B6100. There are also two SATA power connectors for the SATA hard disks. The SATA power cable is an optional accessory. If you need a SATA power connector, please contact CarTFT.com
.

4.7 USB

There are six USB 2.0 interfaces on the AR-B6100. Four USB connectors are located on the edge of the board. The other two USB ports are supported by two 5 pin internal connector. You need a special cable for using these two USB ports and they are optional accessories.

Note:

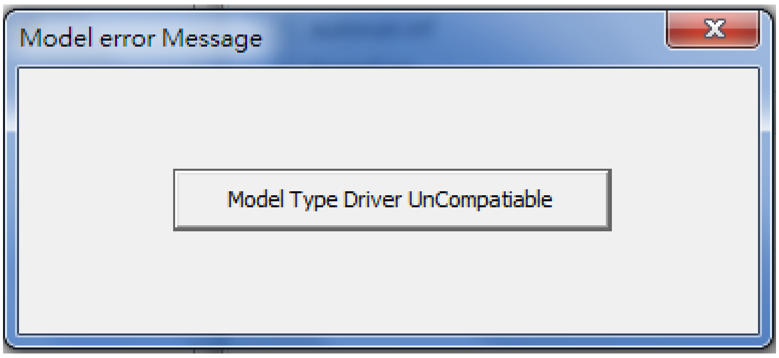
1. If remote switch is not connected or loosed, the status LED will be flashing.
2. Please use Intel Graphics AP to adjust resolution clone.

Chapter 5 Driver And Utility Installation

5.1 Introduction to Driver CD Interface

CarTFT.com provides the a driver CD, which includes the drivers, utilities, applications and documents. For Windows environment, it can be guided by the setup program; for Linux environment, the related files can be found at folder “ARB6100\Linux”.

Once putting the CD into the optical disk drive, it will run automatically. The driver CD will also detect the MB information to see if they are matched. The following error messages appear if you get an incorrect driver CD.



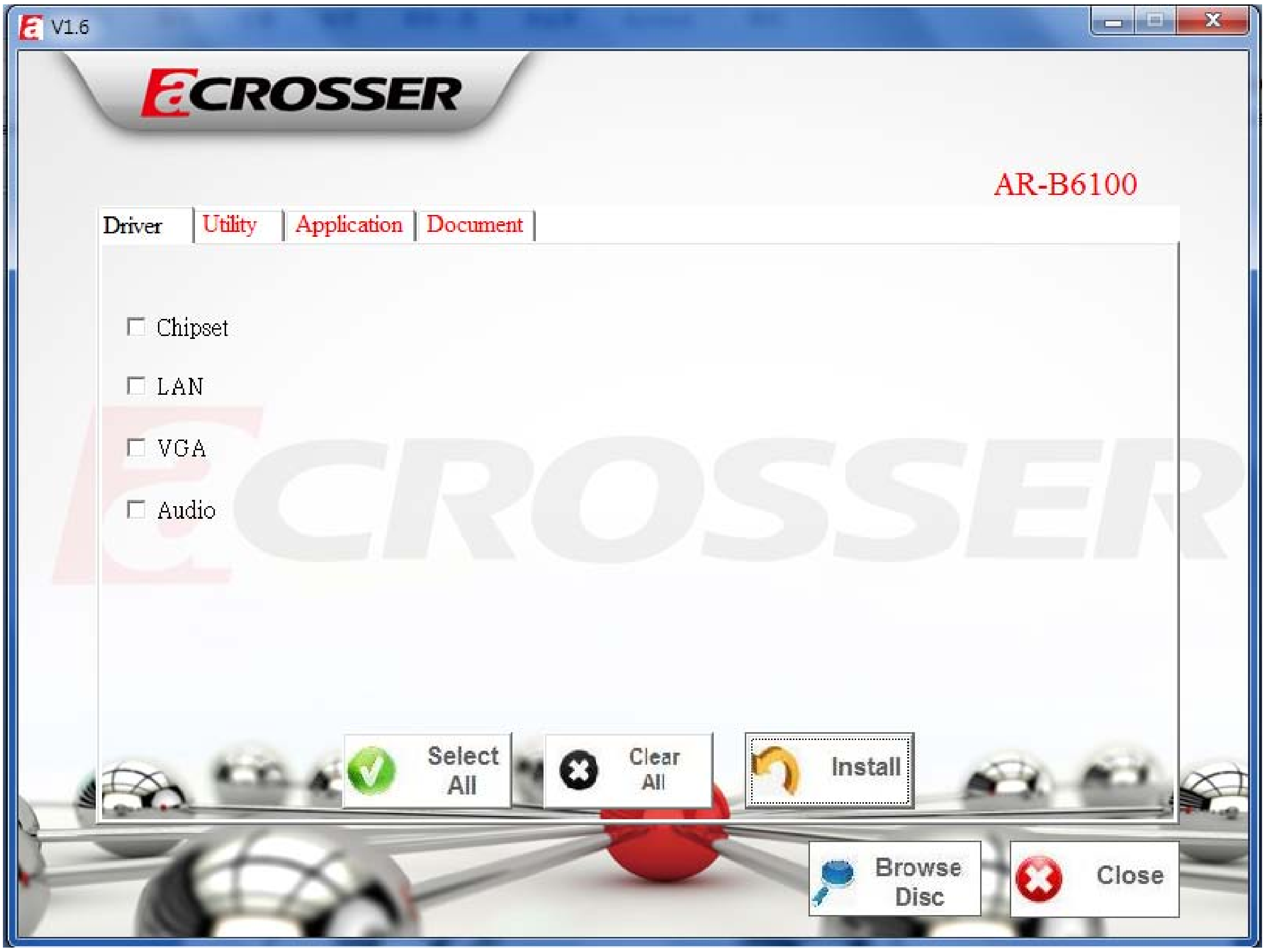
Model error Message

Model Type Driver UnCompatiable

It indicates that the board information is not available and the program gets wrong boid information.

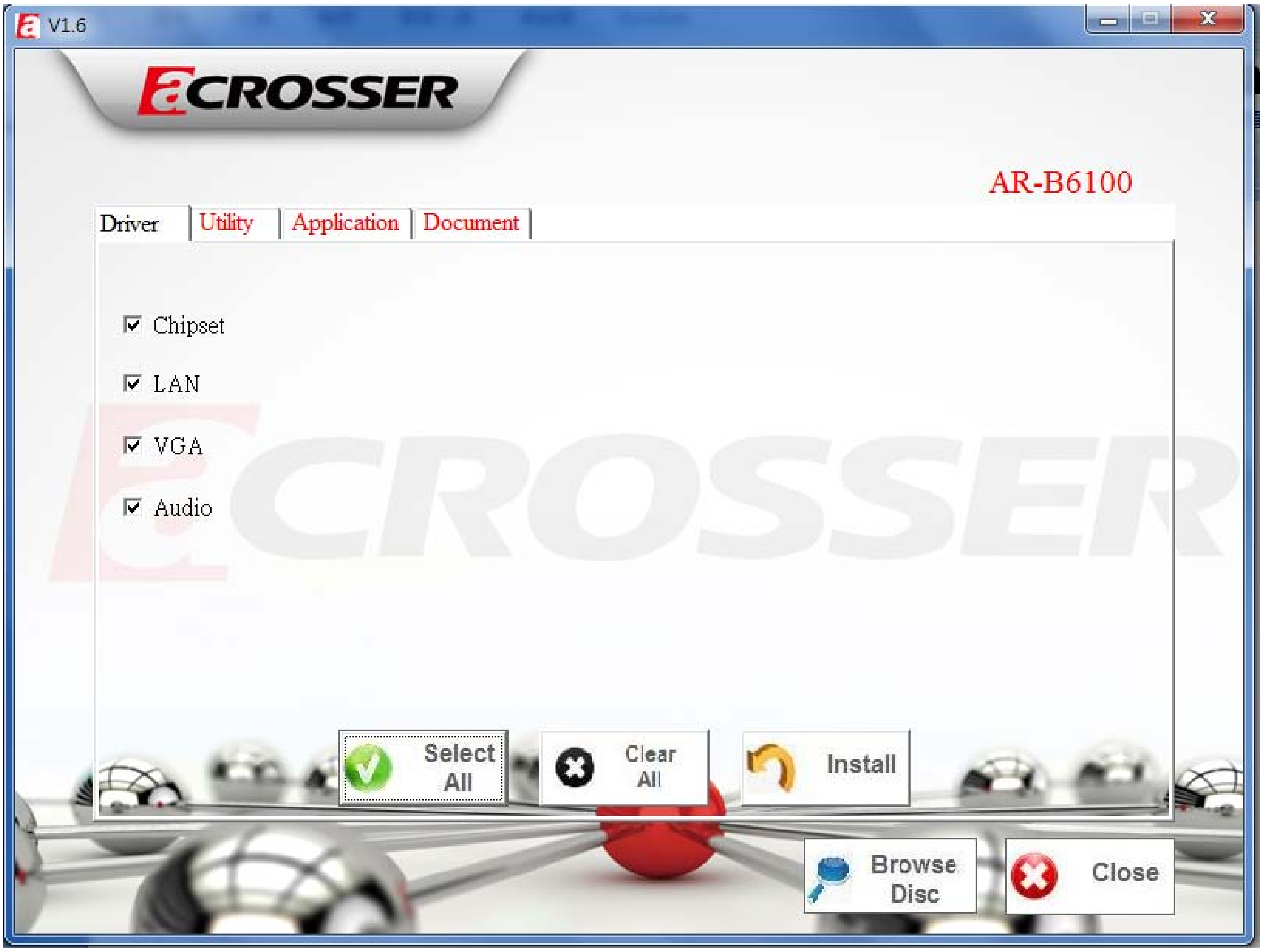
Driver Page

This is the Driver Installation Page.



✓ Select All

Click the icon, all the drivers will be selected.



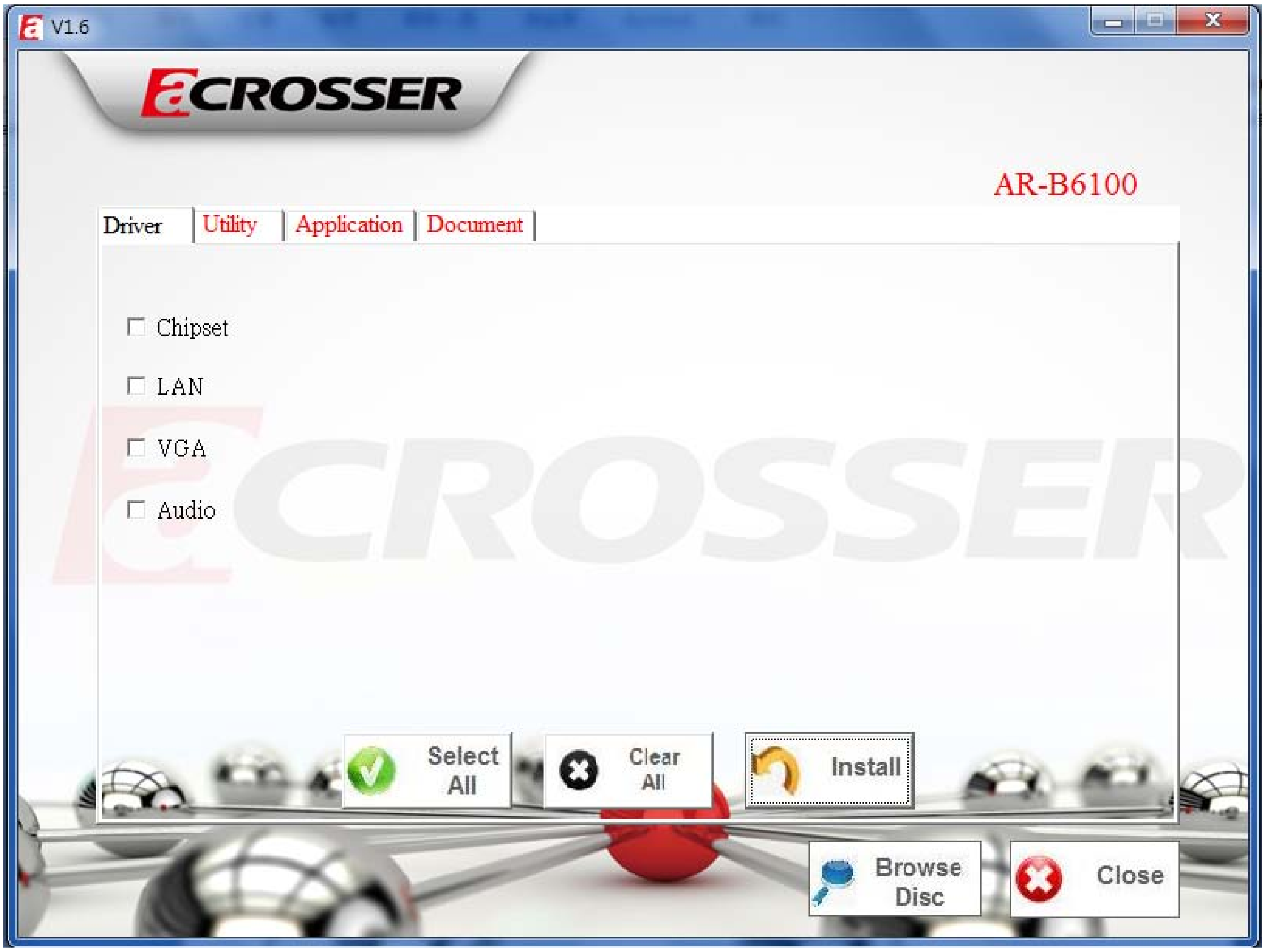
☑ Chipset

☑ LAN

☑ VGA

☑ Audio

Clear All Click the icon, all selected items will be cleared.



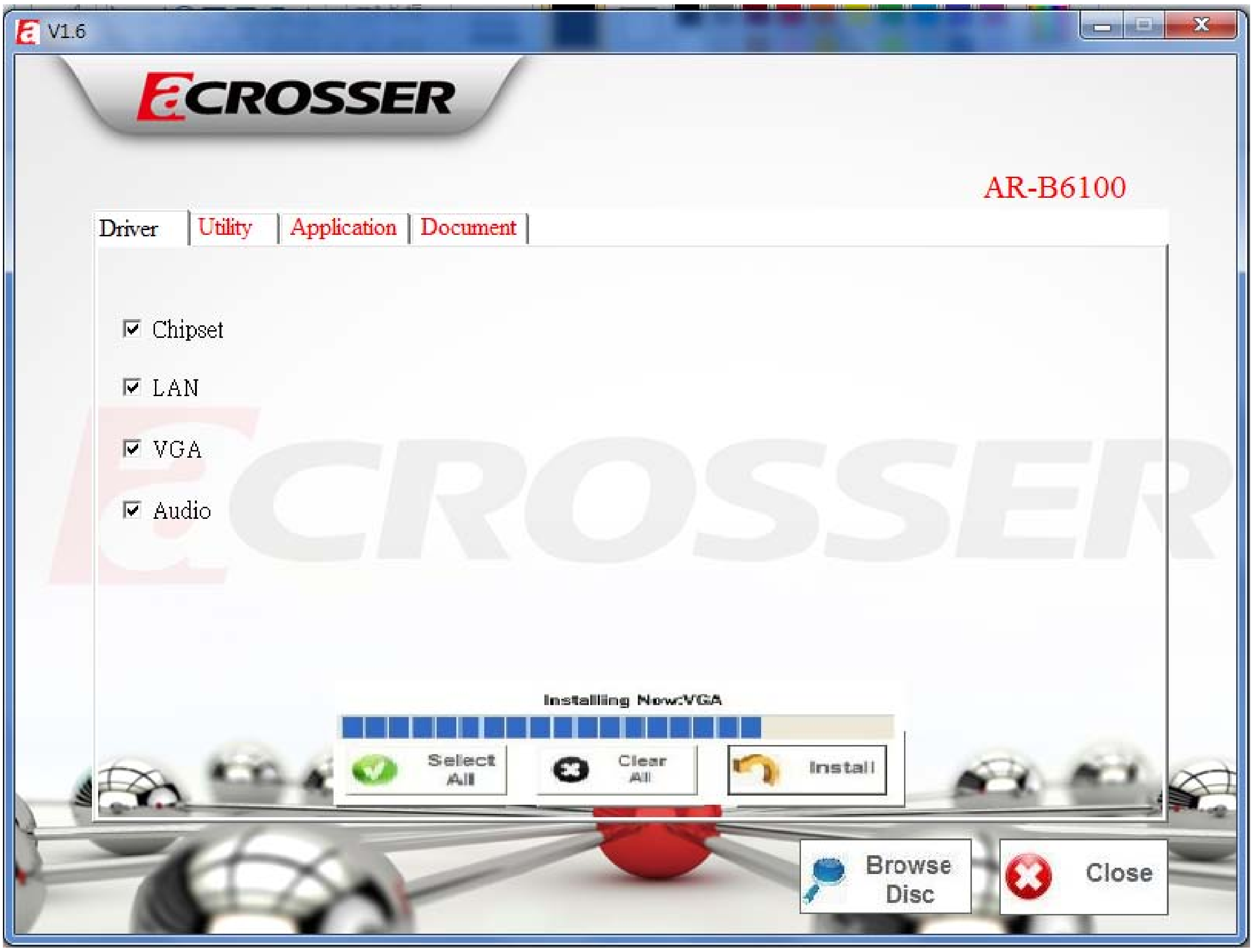
□ Chipset

□ LAN

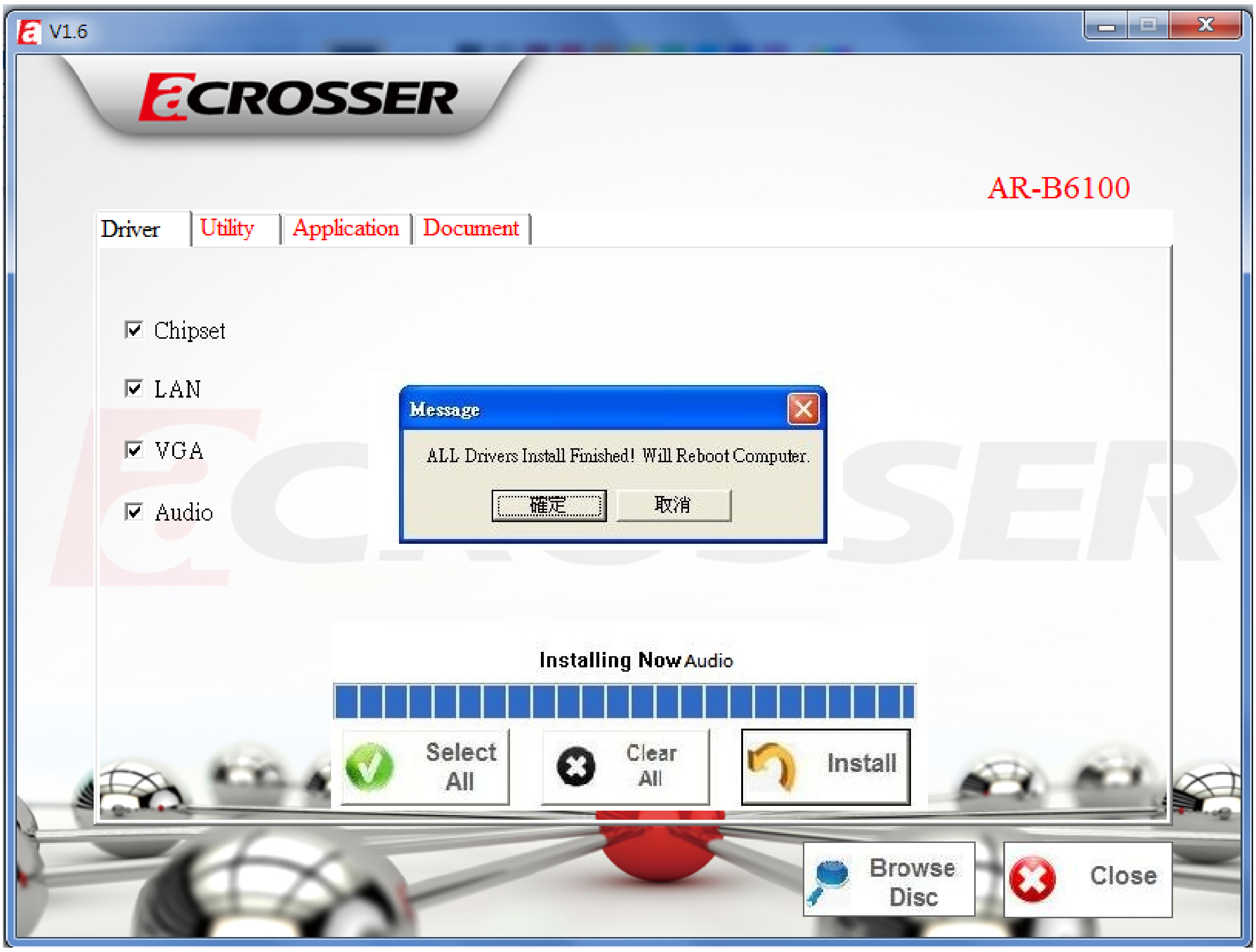
□ VGA

□ Audio

Install Click the icon to install the selected drivers.(**Windows XP 32bit Driver Installation)**



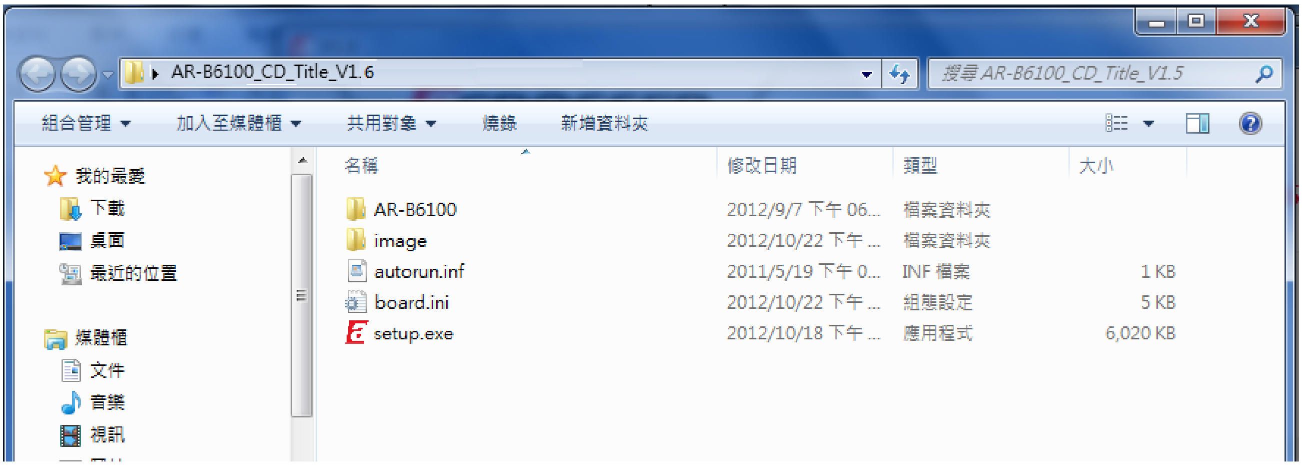
Please click 'Yes' to restart the system.



ALL Drivers Install Finished! Will Reboot Computer.

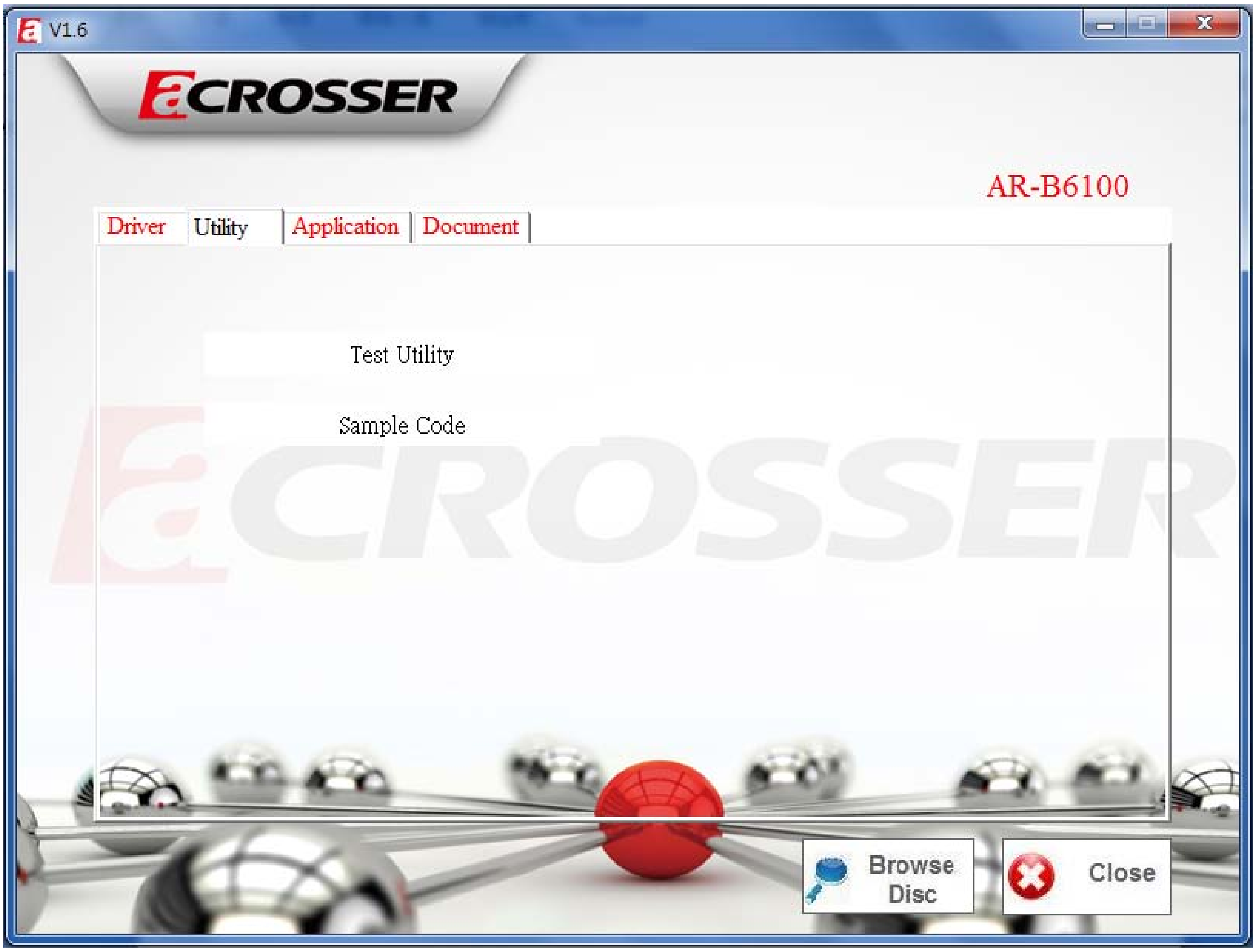
Installing Now Audio

Browse Disc Click this icon to browse this CD content.



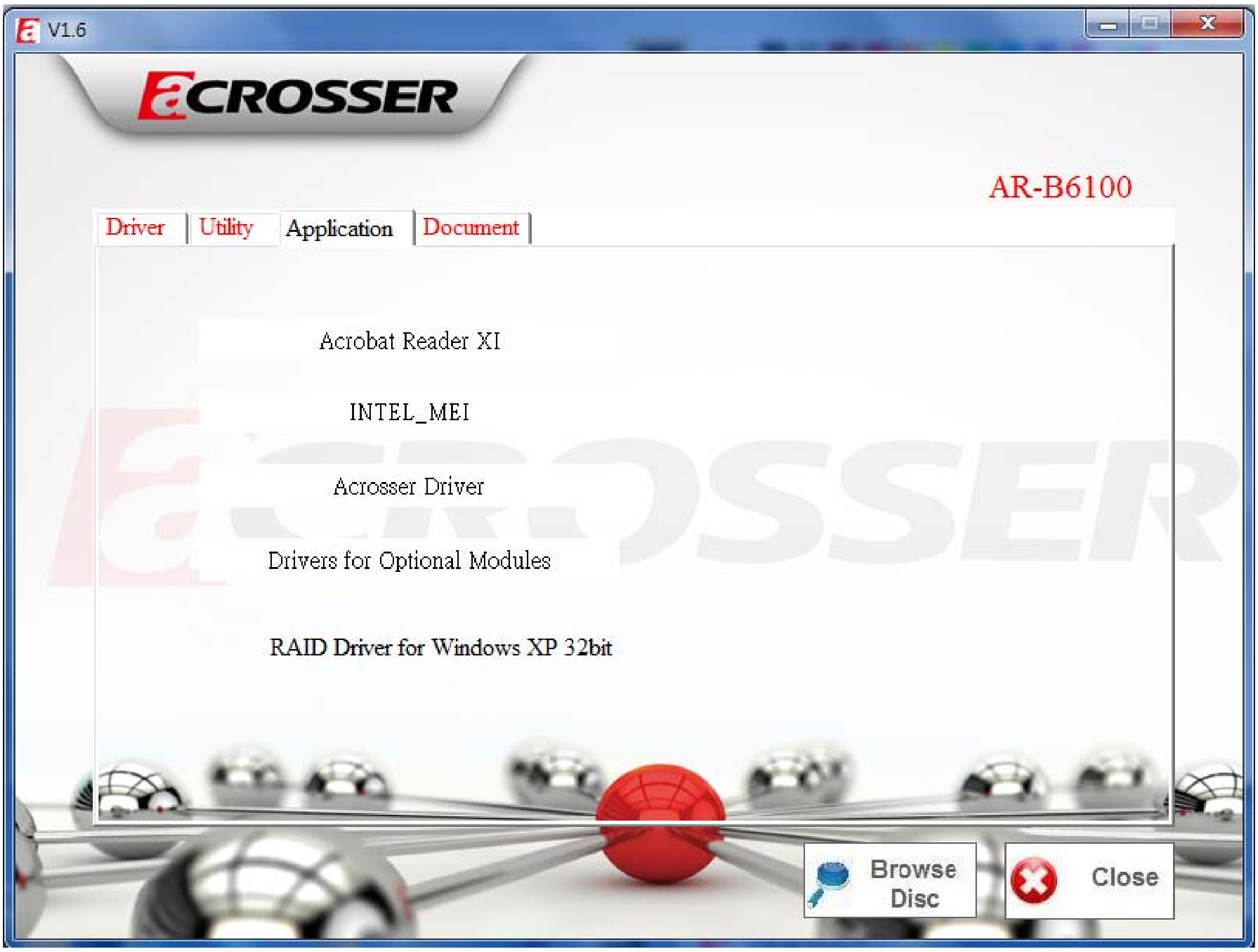
名稱	修改日期	類型	大小
AR-B6100	2012/9/7 下午 06...	檔案資料夾	
image	2012/10/22 下午 ...	檔案資料夾	
autorun.inf	2011/5/19 下午 0...	INF 檔案	1 KB
board.ini	2012/10/22 下午 ...	組態設定	5 KB
setup.exe	2012/10/18 下午 ...	應用程式	6,020 KB

Utility Page

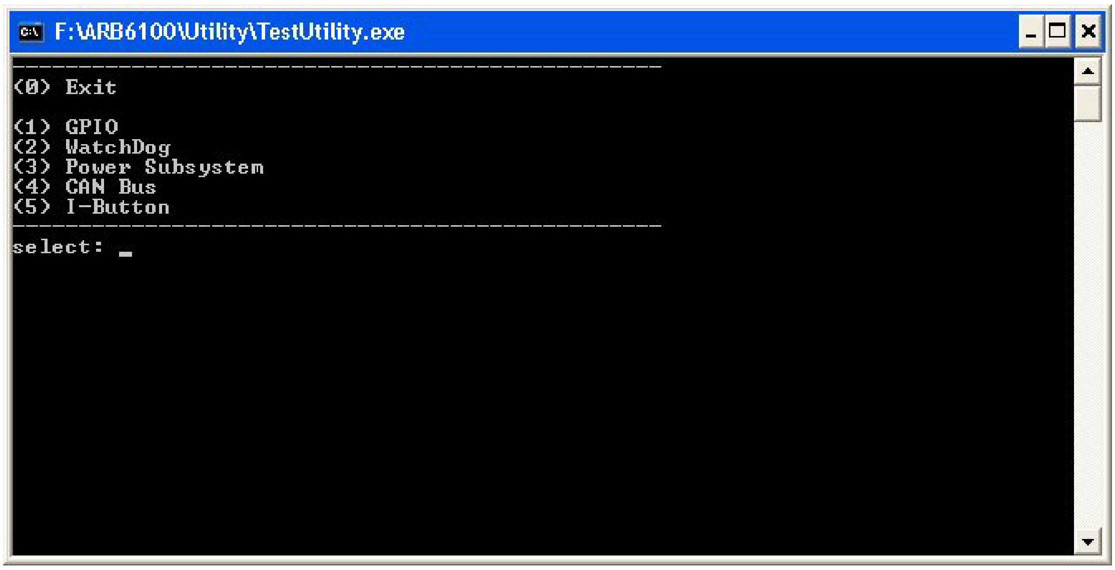


CarTFT.com provides a test utility. Users can double click the item ‘Test Utility’ on the ‘Utility’ page to launch this utility.

Before launching this utility, users have to install the ‘Acrosser Driver’ in advance. You can find this driver on the ‘Application’ page. The system may ask for installing other libraries. You can find the libraries on the ‘Application’ page also.

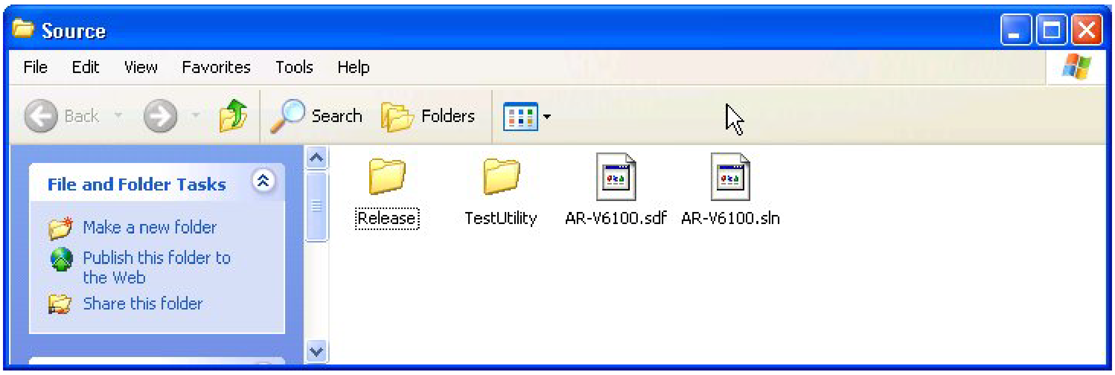


This is the test utility.

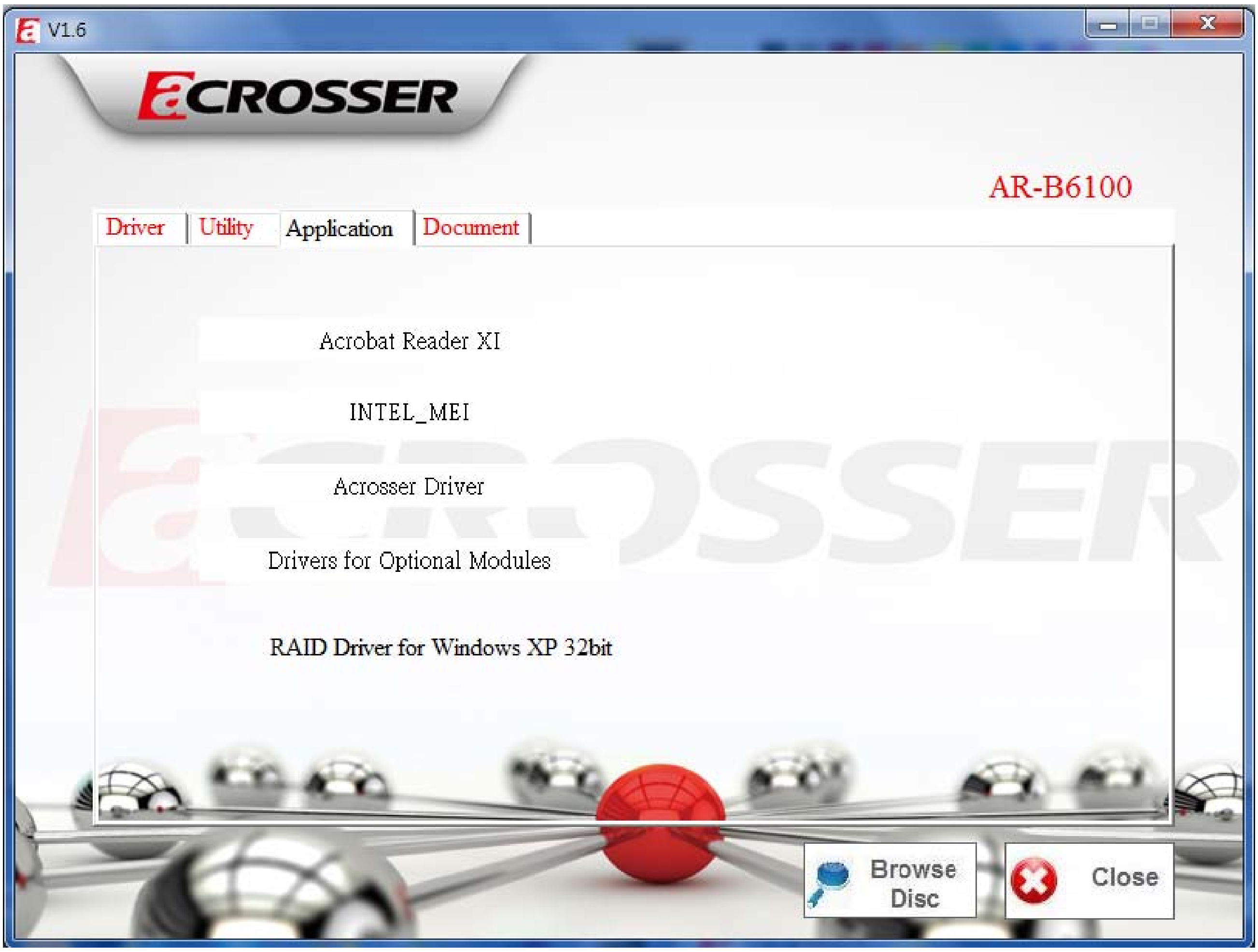


```
--------------------------------------------------
<0> Exit

<1> GPIO
<2> WatchDog
<3> Power Subsystem
<4> CAN Bus
<5> I-Button
--------------------------------------------------
select: _
```

Users can double click the ‘Sample Code’ to open the sample code folder. The source code of the test utility is in this folder.



Application Page

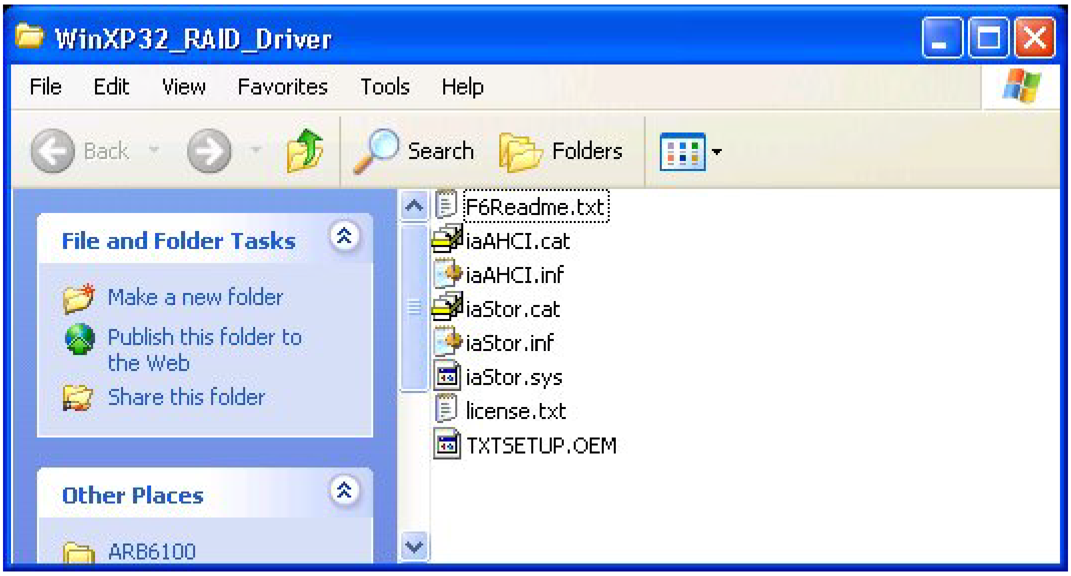


- Acrobat Reader 9.2

  Double click this item to install the Acrobat Reader program.

- RAID Driver for Windows XP 32bit

  Double click this item to open the folder of the Windows XP RAID Driver. Users need this driver package if they install the Windows XP in the AHCI mode.(You can reference "Note" in the end of this chapter.)

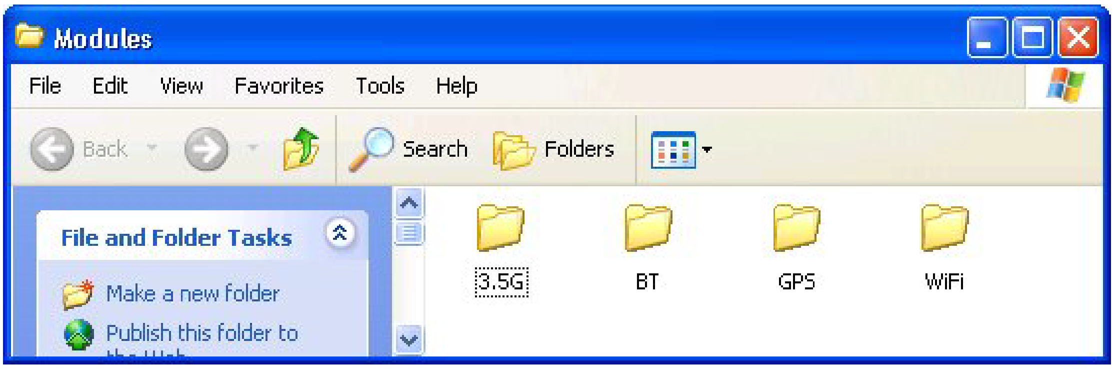


- Acrosser Driver

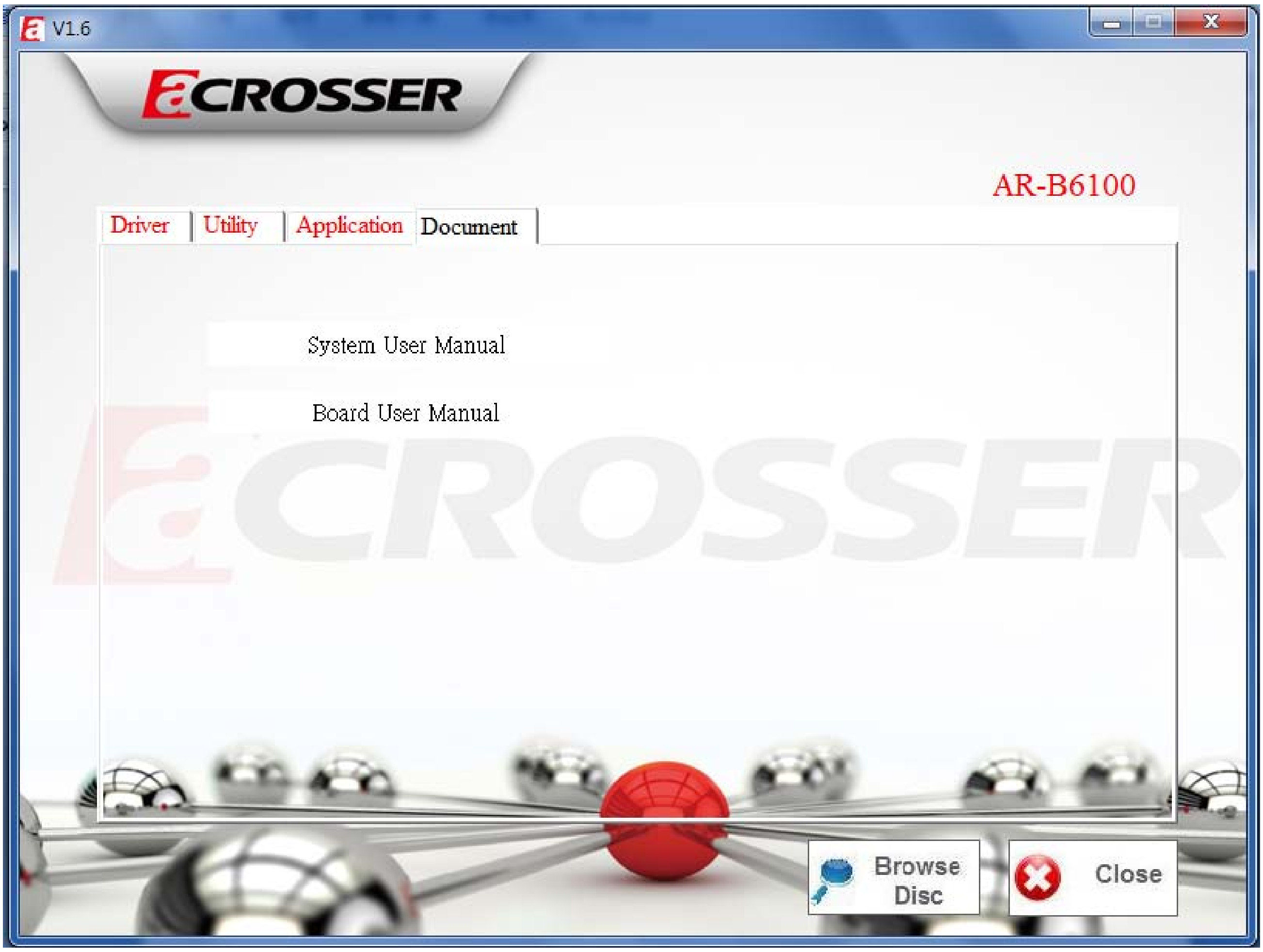
  Install this driver before launching the Test Utility for the first time.

- Driver for Optional Modules

  Double click this item to open the folder. There are drivers for optional modules in this folder.



Documents Page



System User Manual

Board User Manual

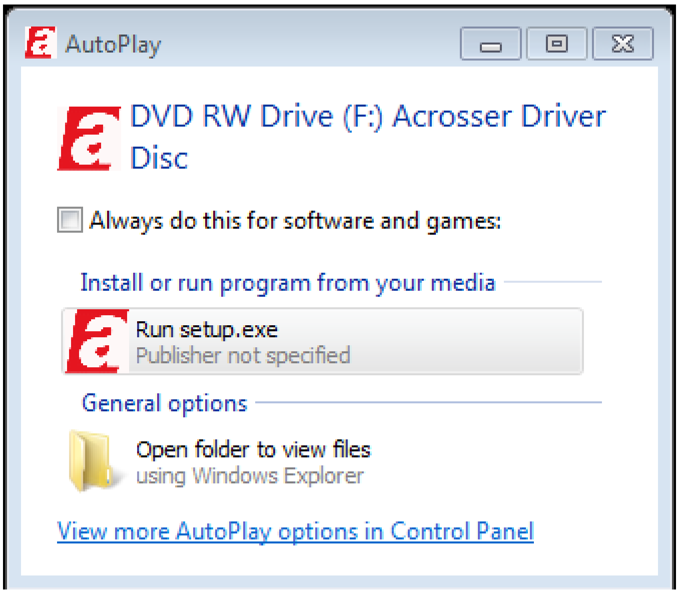
Double click on one of the items to open the manual.

5.2 Windows 7 32 / 64 bit Driver Installation

Please be noted. Since Windows 7 64 bit edition needs certified digital signing to load hardware drivers, in order to run our product correctly, the installation program will automatically enable the test signing feature if it runs under Windows 7 64 bit environment.

Installing Drivers

- Put the Driver Disk into the optical disk drive. Then click the 'Run setup.exe' to run the install program.



AutoPlay

DVD RW Drive (F:) Acrosser Driver Disc

□ Always do this for software and games:

Install or run program from your media

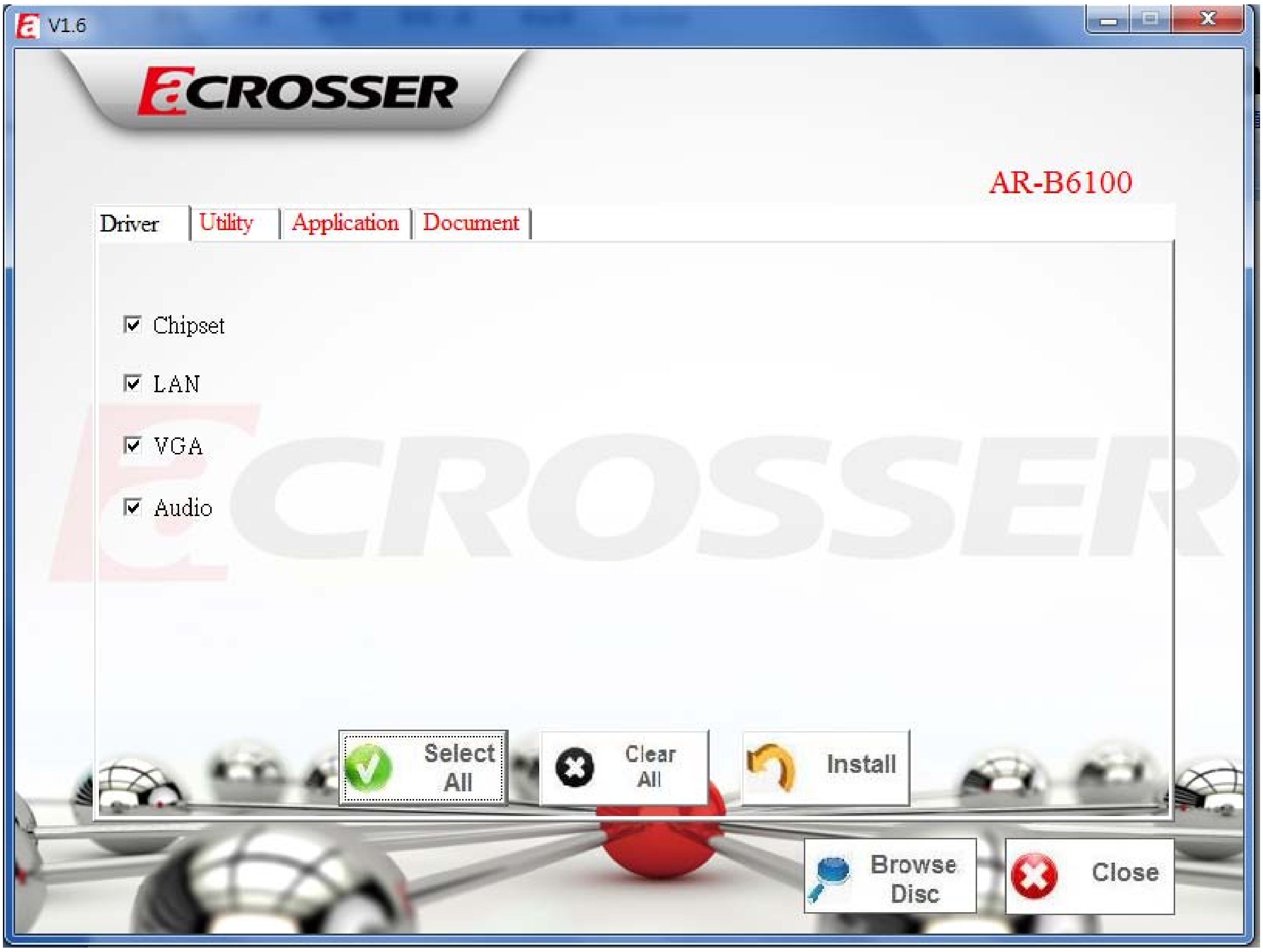
Run setup.exe
Publisher not specified

General options

Open folder to view files
using Windows Explorer

View more AutoPlay options in Control Panel

- The program will appear on the screen. Please click the ‘Select All’ icon.



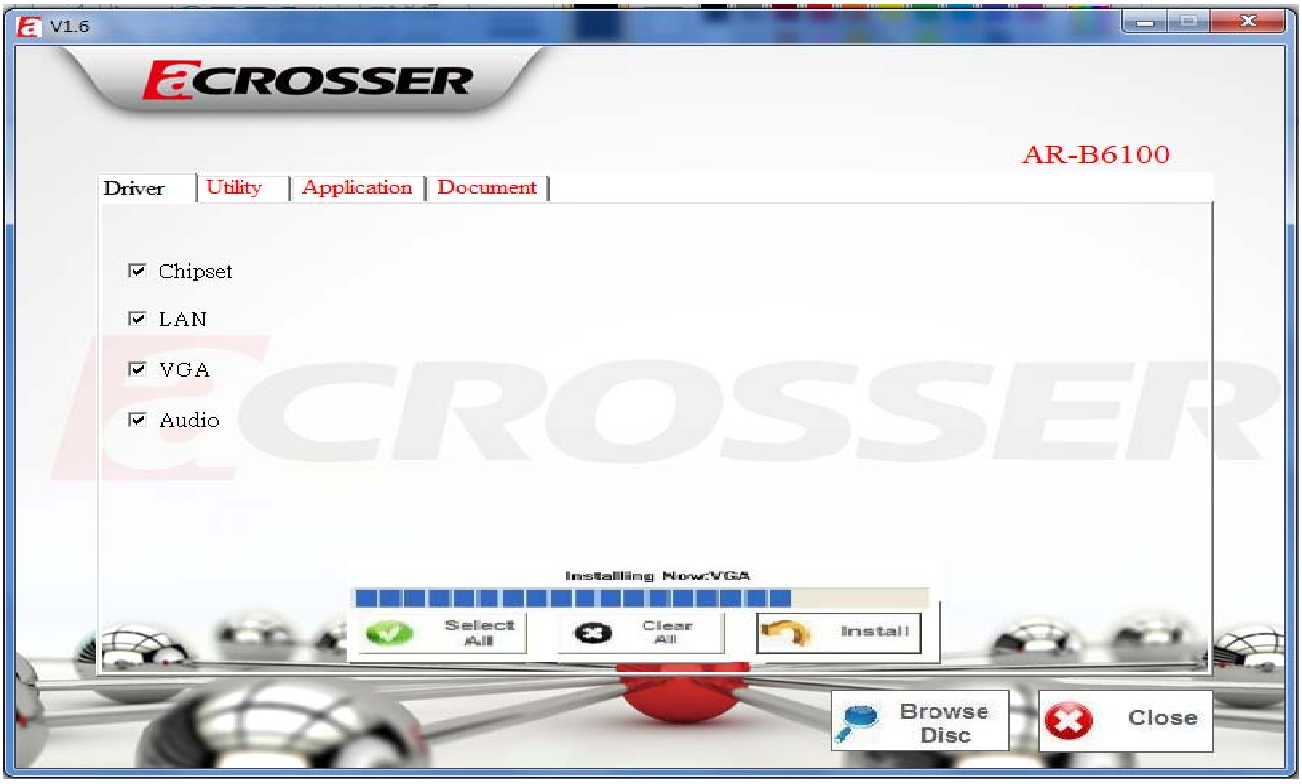
☑ Chipset

☑ LAN

☑ VGA

☑ Audio

- Click the ‘Install’ icon to install the drivers.



☑ Chipset

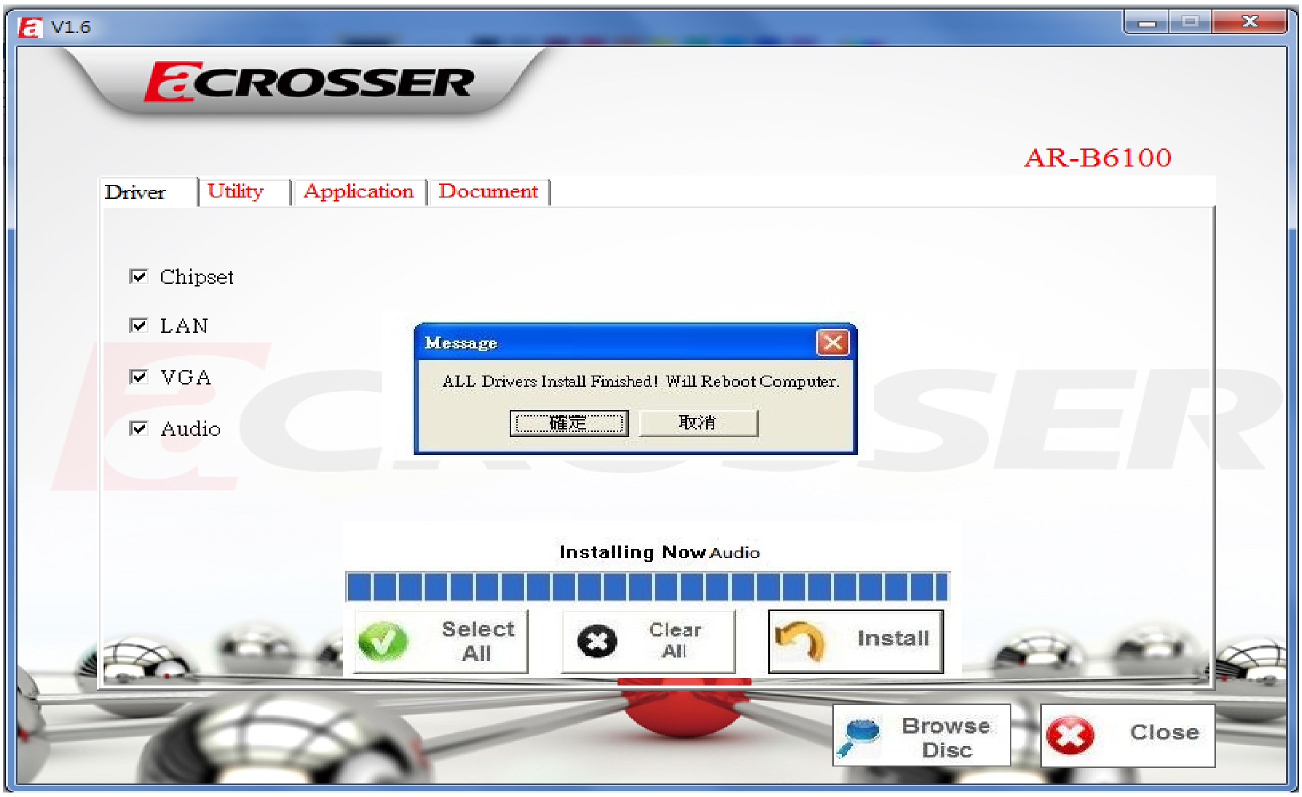
☑ LAN

☑ VGA

☑ Audio

Installing Now:VGA

- Finish the driver installation. Please click ‘Yes’ to restart the system.



☑ Chipset

☑ LAN

☑ VGA

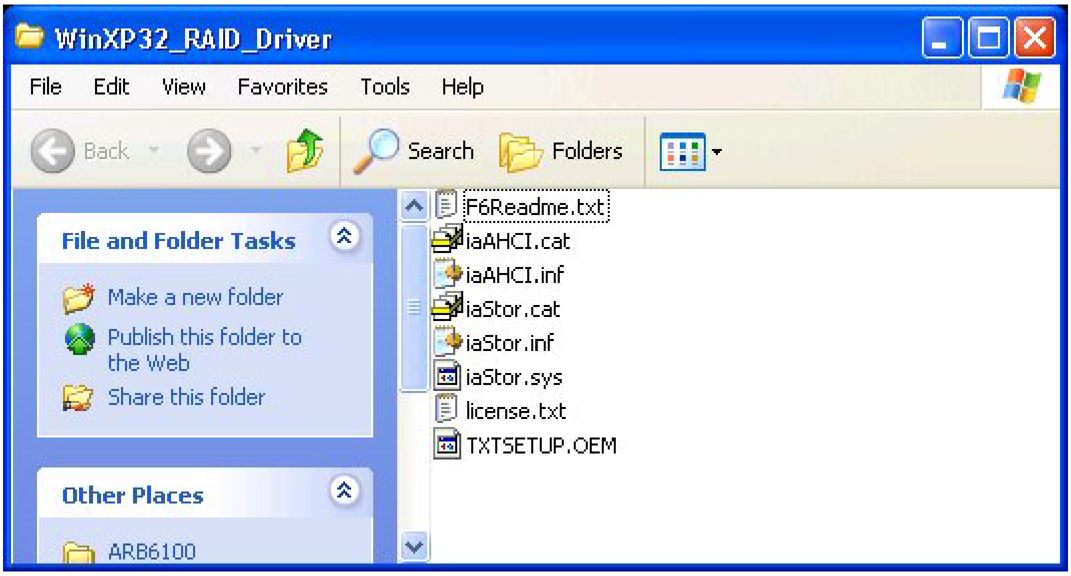
☑ Audio

ALL Drivers Install Finished! Will Reboot Computer.

Installing Now Audio

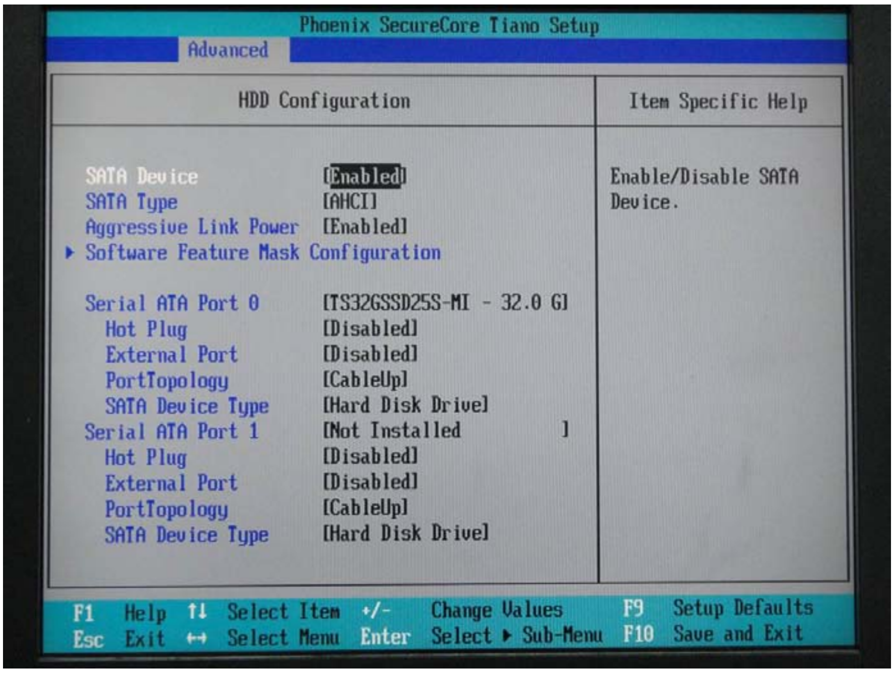
Note: Installing Windows XP in the AHCI mode

Due to Windows XP is older operating system, it don’t include AHCI driver.
If you want to install Windows XP operating system in the AHCI mode, please follow the steps listed below. (reference ‘F6Readme.txt’ from folder of the WinXP32_RAID Driver)
Double click this item to open the folder of the WinXP32_RAID Driver. Users need this driver package if they install the Windows XP in the AHCI mode.



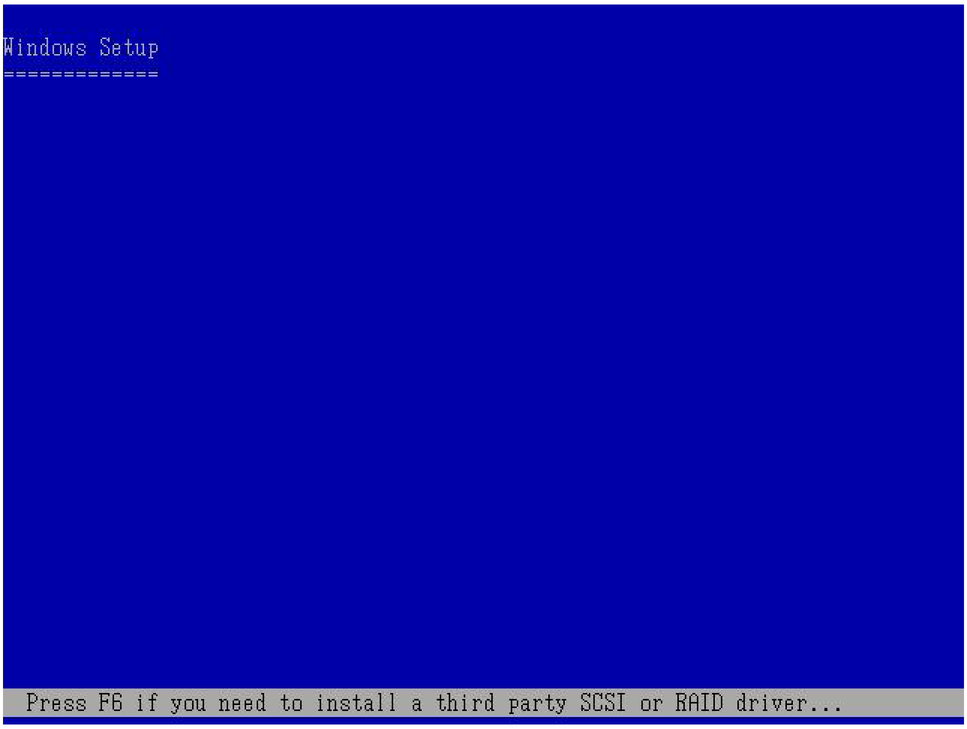
F6Readme.txt
iaAHCI.cat
iaAHCI.inf
iaStor.cat
iaStor.inf
iaStor.sys
license.txt
TXTSETUP.OEM

- Prepare a USB floppy drive and a floppy disk. Copy all the files in this folder to the floppy disk.
- In the BIOS setup, enable the AHCI mode of the hard drive.



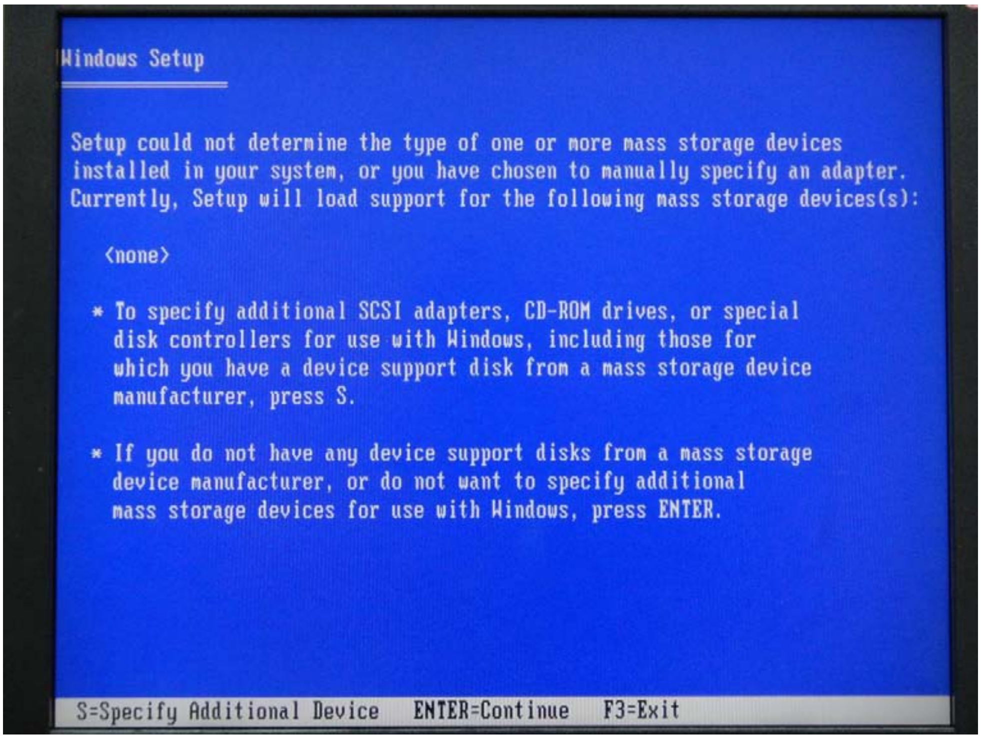
```
                     Phoenix SecureCore Tiano Setup
          Advanced

              HDD Configuration                      Item Specific Help

   SATA Device                  [Enabled]            Enable/Disable SATA
   SATA Type                    [AHCI]               Device.
   Aggressive Link Power        [Enabled]
 ▸ Software Feature Mask Configuration

   Serial ATA Port 0            [TS32GSSD25S-MI - 32.0 G]
     Hot Plug                   [Disabled]
     External Port              [Disabled]
     PortTopology               [CableUp]
     SATA Device Type           [Hard Disk Drive]
   Serial ATA Port 1            [Not Installed            ]
     Hot Plug                   [Disabled]
     External Port              [Disabled]
     PortTopology               [CableUp]
     SATA Device Type           [Hard Disk Drive]

 F1   Help  ↑↓  Select Item  +/-    Change Values     F9   Setup Defaults
 Esc  Exit  ↔   Select Menu  Enter  Select ▸ Sub-Menu  F10  Save and Exit
```

- Connect the floppy drive to the system before installing the Windows XP operating system. Make sure the floppy disk is inserted.
- Boot the system with the installation CD. Follow the instructions on the screen. As soon as the screen shows this information, press 'F6'.



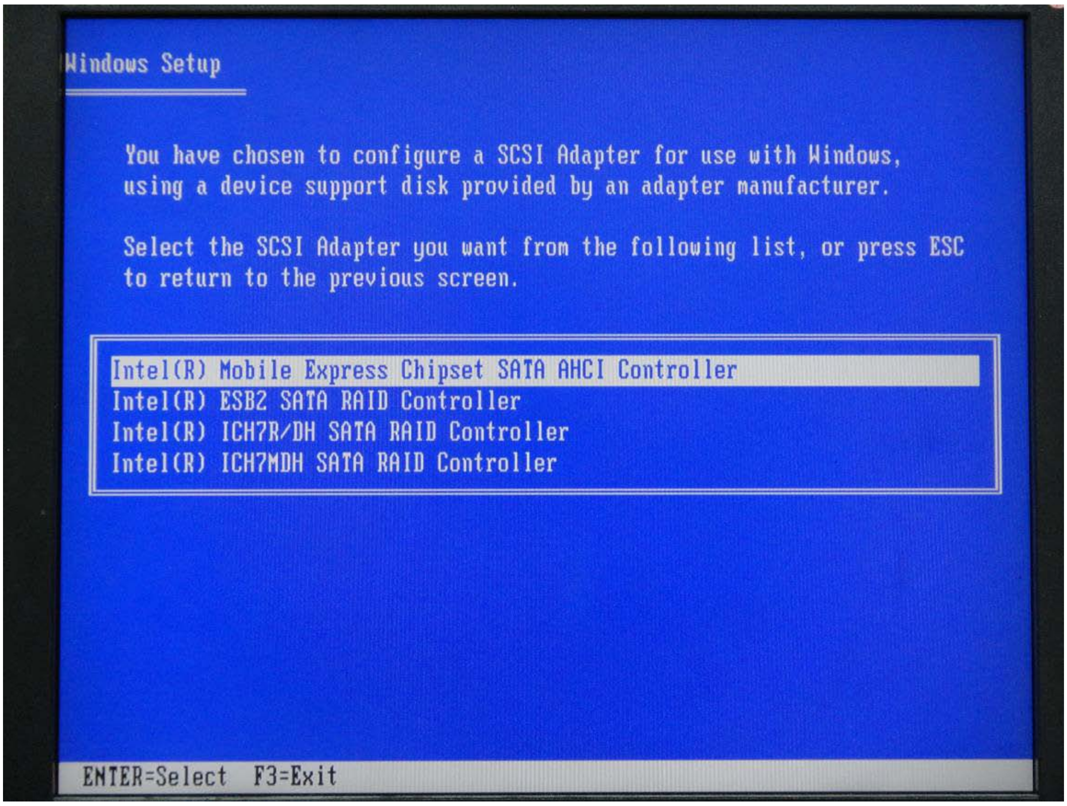
```
Windows Setup
=============

 Press F6 if you need to install a third party SCSI or RAID driver...
```

- When the screen shows this information, press 'S'.



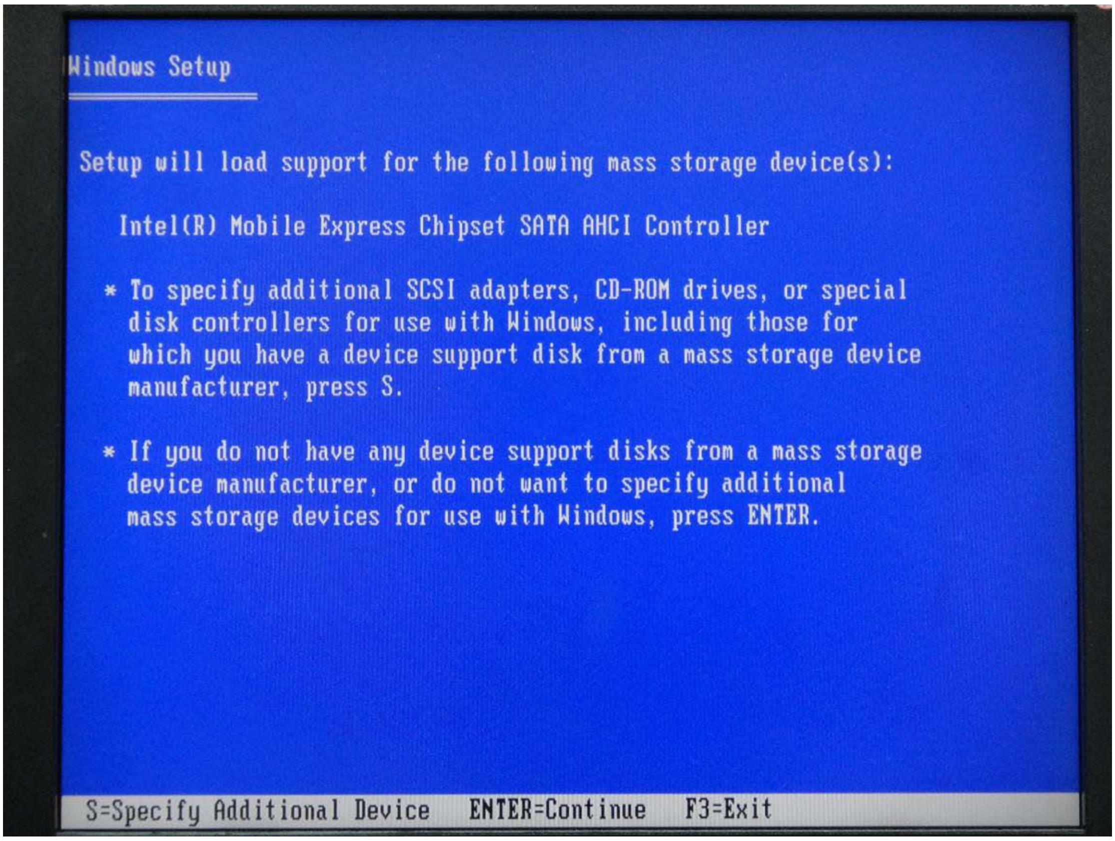
```
Windows Setup

 Setup could not determine the type of one or more mass storage devices
 installed in your system, or you have chosen to manually specify an adapter.
 Currently, Setup will load support for the following mass storage devices(s):

    <none>

  * To specify additional SCSI adapters, CD-ROM drives, or special
    disk controllers for use with Windows, including those for
    which you have a device support disk from a mass storage device
    manufacturer, press S.

  * If you do not have any device support disks from a mass storage
    device manufacturer, or do not want to specify additional
    mass storage devices for use with Windows, press ENTER.

 S=Specify Additional Device   ENTER=Continue   F3=Exit
```

- When the screen shows a list of available drivers, choose the ‘Mobile Express Chipset SATA AHCI Controller’.



```
Windows Setup

   You have chosen to configure a SCSI Adapter for use with Windows,
   using a device support disk provided by an adapter manufacturer.

   Select the SCSI Adapter you want from the following list, or press ESC
   to return to the previous screen.

    Intel(R) Mobile Express Chipset SATA AHCI Controller
    Intel(R) ESB2 SATA RAID Controller
    Intel(R) ICH7R/DH SATA RAID Controller
    Intel(R) ICH7MDH SATA RAID Controller

 ENTER=Select  F3=Exit
```

- When the screen shows this information, press ‘Enter’ to continue installing the operating system.



```
Windows Setup

 Setup will load support for the following mass storage device(s):

    Intel(R) Mobile Express Chipset SATA AHCI Controller

   * To specify additional SCSI adapters, CD-ROM drives, or special
     disk controllers for use with Windows, including those for
     which you have a device support disk from a mass storage device
     manufacturer, press S.

   * If you do not have any device support disks from a mass storage
     device manufacturer, or do not want to specify additional
     mass storage devices for use with Windows, press ENTER.

 S=Specify Additional Device   ENTER=Continue   F3=Exit
```

Chapter 6 Software Installation and Programming Guide

6.1 CAN bus

6.1.1 Overview

The CAN bus APIs provide interfaces to CAN bus subsystem. By invoking these APIs, programmers can implement the applications which have the functions listed below:

1. Set the BAUD rate.
2. Send the CAN packages over the CAN bus.
3. Receive the CAN packages via the CAN bus hardware interface.
4. Set the CAN package filter to selectively receive CAN packages with specific ID.
5. Set the mask bits to selectively make some filter bits take effect.
6. Full Mode Enable.
7. Full Mode Disable.

In folder 'ARB6100\Utility\AR-V6100_Source' on the CD, we provides:

1. API header file.
2. API library in static library format and shared library format.
3. Test utility and its source code.

6.1.2 CAN Message Format

```
// TYPE DEFINITION
typedef  char            i8;
typedef  unsigned char   u8;
typedef  short           i16;
typedef  unsigned short  u16;
typedef  unsigned long   u32;
typedef  int             i32;


struct CanMsg {
        u32   id;
        u8    id_type;
        u8    length;
```

```
    u8    da  ta[8];
}
```

To transmit a CAN packet, the programmer has to fill in the fields in the variable of type CanMsg and pass this CanMsg variable as an argument to invoke the APIs. The fields in CAN message are described below:

id:

This field holds the ID information of the CAN packet. In a ‘Standard Data Frame’ CAN packet, the ID field consists of 11 bits of binary digitals. In an ‘Extended Data Frame’ CAN packet, the ID field consists of 29 bits of binary digitals. That the CAN packet is a ‘Standard Data Frame’ packet or an ‘Extended Data Frame’ packet is determined by the ‘id_type’ field in the CanMsg variable.

The ‘id’ field in the CanMsg variable is a 32-bit long space. If a CanMsg variable is configured as a ‘Standard Data Frame’ CAN packet, the bit[0] ~ bit[10] in the ‘id’ field is the ID of the CAN packet. The bit[11] ~ bit[31] are ignored when the APIs in the library processing the CanMsg variable.

'id' field in the CanMsg variable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	1	0	0	1	1	1	0	1	0	1	1

If a CanMsg variable is configured as an ‘Extended Data Frame’ CAN packet, the bit[0] ~ bit[28] in the ‘id’ field is the ID of the CAN packet. The bit[29] ~ bit[31] are ignored when the APIs in the library processing the CanMsg variable.

'id' field in the CanMsg variable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
☒	☒	☒	1	0	0	1	0	0	1	0	1	1	1	0	0	1	0	1	1	0	1	0	0	1	1	1	0	1	0	1	1

id_type:

This field identifies that the CAN packet is a ‘Standard Data Frame’ CAN packet or a ‘Extended Data Frame’ CAN packet:

```
struct CanMsg canMsg;
canMsg.id_type = EXT_ID;    /  / A ‘Extended Data Frame’ packet
canMsg.id_type = STD_ID;      // A ‘Standard Data Frame’ packet
```

length:

This field identifies the number of data bytes in the next field 'data[8]' which are

filled with effective data. Because the 'data' field is an 8-byte long array, the range of this field 'length' is 0 ~ 8.

data[8]:

This array of data will be filled with effective data.

For example:

```
struct CanMsg msg;

msg.data[0] = 0xa1;
msg.data[1] = 0xb2;
msg.data[2] = 0xc3;

msg.length = 3;
```

6.2 GPIO and Watchdog

6.2.1 Overview

This model provides both a GPIO interface and a Watchdog timer. Users can use the GPIO and Watchdog APIs to configure and to access the GPIO interface and the Watchdog timer. The GPIO has four input pins and four output pins. The Watchdog timer can be set to 1~255 seconds. Setting the timer to zero disables the timer. The remaining seconds of the timer to reboot can be read from the timer.

6.2.2 Installing Device Driver

Before executing the applications which invoke the GPIO or Watchdog APIs, users should make sure that the Linux device driver or the Windows device driver has been installed.

On Linux platform, after successfully installing the device driver, a character device node named "/dev/AcroDev" will be created automatically. The APIs open the device node "/dev/AcroDev" implicitly so acquiring a file descriptor of "/dev/AcroDev" is not ncecssary.

On Windows platform, after successfully installing the device driver, there is a device which shows 'Acrosser Device' in the 'Device Manager'. The APIs on Windows platform open this device implicitly.

6.3 Power Subsystem

6.3.1 Overview

The Power Subsystem APIs can be used to get and set the configuration of power subsystem. By invoking the Power Subsystem APIs, the users can:

1. Get the firmware version number of the Power Subsystem.
2. Set all the settings of the Power Subsystem to the default values.
3. Get/Set the status of the remote switch(ENABLE or DISABLE).
4. Get the battery voltage.
5. Get/set the status of the battery monitor (ON or OFF).
6. Get/set the delta value which identifies how much the battery voltage can be lower than the nominal voltage. When the voltage is lower than the tolerable voltage, the power subsystem turns off the system.
7. Get/set the Soft Off deley.
8. Get/set the Hard Off delay.
9. Get/set the Power On delay.
10. Get/set the Shutdown delay.

The power subsystem connects to the main system via the COM port. On the Linux platform, the actual port number to which the Power Subsystem connects is determined by the Linux. The default supported COM interfaces on Linux are COM1~COM4. Users must take extra steps to configure Linux kernel in order to support COM ports which do not fall into the range COM1 ~ COM4. Please refer to Appendix A for more information. Users don’t need extraordinary setup on Windows platform to support COM ports.

6.4 I-Button Function

In the API library, we provide a set of I-Button functions. Users can use the functions to:

1. Reset the I-Button.
2. Read data from the I-Button.
3. Write data to the I-Button.

6.5 API List and Descriptions

6.5.1 CAN Bus

1. **Syntax:**

   ```
   i32 getCanFwVer(PicInfo *ver)
   ```

   Descriptions: This function gets the version information of the CAN Bus firmware.
   Parameters: The definition of struct 'PicInfo' is:

   ```
   struct PicInfo {
         u8 info[12];
   }
   ```

   This API returns the version information and store the information in the memory which is pointed at by the pointer 'ver'.

   Return Value: If this function gets the version information successfully, it returns 0, any other returned value stands for error.

2. **Syntax:**

   ```
   i32 getCanBaudRate(u8 *baud)
   ```

   Descriptions: This function gets the current setting of the Baud Rate of the CAN Bus. This function gets an 'unsigned char' to represent the Baud Rate. Here is the table for the Baud Rate:

Unsigned Char	Baud Rate
1	10K
2	20K
3	50K
4	100K
5	125K
6	250K
7	500K
8	800K
9	1000K

Users can use the macros listed below to set the Baud Rate:

```
/* Baud Rate */
#define BAUD_RATE_10K            1
#define BAUD_RATE_20K            2
#define BAUD_RATE_50K            3
#define BAUD_RATE_100K           4
#define BAUD_RATE_125K           5
#define BAUD_RATE_250K           6
#define BAUD_RATE_500K           7
#define BAUD_RATE_800K           8
#define BAUD_RATE_1000K          9
```

Parameters: This function gets a number which represents the specific Baud Rate and stores it at the memory which is pointed at by the pointer ‘baud’.

Return Value: If this function gets the baud rate successfully, it returns 0, any other returned value stands for error.

3. **Syntax:**

```
i32 setCanBaudRate(u8 baud)
```

Descriptions: This function sets the Baud Rate of the CAN Bus.

Parameters: It takes an ‘unsigned char’ as the parameter and sets the Baud Rate according to the value stored at the parameter ‘baud’. The correspondence between the Baud rate and the value to set to the function is the same as the table listed in the previous API ‘getCanBaudRate( )’

Return Value: If this function sets the baud rate successfully, it returns 0, any other returned value stands for error.

4. **Syntax:**

```
i32 sendCanMessage( struct CanMsg *buffer, u8 count )
```

Description: This function sends out CAN packages over the CAN bus.

Parameters: If there is more than one CAN packet to send, these CAN packages are stored in an array of type ‘CanMsg’. This function sends out packets in a sequential fashion. The memory address of the first CAN packet to be sent is pointed at by the parameter ‘buffer’. The number of CAN packets to be sent is indicated by the parameter ‘count’.

Return Value: If this function sends the CAN packet successfully, it returns 0, any other returned value stands for error.

Here is an example:
If the CAN packets in the array ‘canAry[]’ have been initialized. The code listed below will send out the CAN packets in the ‘canAry[]’ over the CAN bus.

```
unsigned int result = 0;
struct CanMsg canAry[30];
/* …
  Initialize the CAN packages in the canAry[30]
*/
  result = sendCanMessages( canAry, 30 );
  if( result != 0)
  fprintf( stderr, “Send CAN package error!\n”);
```

5. **Syntax:**

```
i32 getCanMessage( struct CanMsg *buffer, u8 count )
```

Description: This function receives CAN packets from the CAN bus subsystem.

Parameters: This function stores received CAN packages sequentially at an array of type ‘CanMsg’. The number of packages to receive is indicated by the parameter ‘count’.

Return Value: If this function receives the CAN packet successfully, it returns 0, any other returned value stands for error.

Here is an example:
If the array ‘canAry[]’ of type ‘CanMsg’ has been declared and allocated. The code listed below will receive 30 CAN packages from the CAN bus subsystem and stores the packages in the ‘canAry[]’.

```
unsigned int result = 0;
struct CanMsg canAry[30];

result = getCanMessage( canAry, 30 );
  if( result != 0)
         fprintf( stderr, "Fail to receive CAN packets!\n");
```

6. **Syntax:**

```
i32 getCanMask(struct CanMask *mask)
```

Description: This function gets the current setting of the acceptance masks. Masks are used to determine which bits in the ID field of the CAN packet are examined with the filters. There are two acceptance masks (mask0 and mask1) and six acceptance filters (filter0 ~ filter5) in the CAN Bus subsystem. Filter0 ~ filter1 are associated with mask0. Filter2 ~ filter4 are associated with mask1.

Here is the Mask/Filter truth table:

Mask bit n	**Filter bit n**	**Message ID bit n**	**Accept or reject bit n**
0	x	x	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Note: x = don't care

Parameters: This parameter 'mask' is a pointer to a variable of type 'CanMask'. Users use the field 'maskId' to indicate the mask they want and the API put the setting of the mask in the 'mask' field.

```
struct CanMask {
      u8 maskId; // 0 or 1
      u32 mask;
}
```

Return Value: If this function receives the mask setting successfully, it returns 0, any other returned value stands for error.

For example:

```
    struct CanMask a_mask;
    a_mask.maskId = 0;    // indicate the mask0
    i32 result;
    result = getCanMask(&a_mask);   //  The setting of the mask is put at
                                    // a_mask.mask
    if( result != 0)
          printf("Fail to get mask!\n");
```

7. **Syntax:**

```
i32 setCanMask(struct CanMask mask)
```

Description: This function sets the bit patterns to the indicated mask. The target mask is indicated by the 'maskId' field in a CanMask variable.

Parameters: This functions takes a variable of type 'CanMask'. User set the bit patterns they want to the 'mask' field in a 'CanMask' variable.

```
struct CanMask {
      u8 maskId;      // 0 or 1
      u32 mask;
}
```

For example:

```
struct CanMask varMask;
i32 result;

varMask.maskId = 1;
varMask.mask = 0x12345678;
result = setCanMask(varMask);
```

Return Value: If this function sets the mask setting successfully, it returns 0, any other returned value stands for error.

8. **Syntax:**

```
i32 getCanFilter(struct CanFilter *varFilter)
```

Description: This function gets the current setting of the acceptance filter. Use the ‘filterId’ field in a ‘CanFilter’ variable to indicate the filter you want and the API puts the setting of the indicated filter in the ‘filter’ field in the CanFilter variable ‘varFilter’.

Parameters: This function takes a pointer to a ‘CanFilter’ type variable.
For example:

```
struct CanFilter varFilter;
i32 result;

result = getCanFilter(&varFilter);
if(result != 0)
        printf(“Fail to get the filter!\n”);
```

Return Value: If this function gets the filter successfully, it returns 0, any other returned value stands for error.

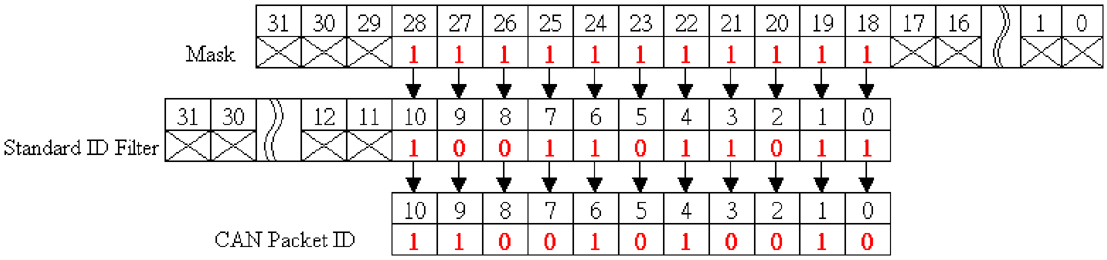
9. **Syntax:**

```
i32 setCanFilter(struct CanFilter *varFilter)
```

Description: This function sets the bit pattern to the filter. By indicating the ‘filterType’ field in the ‘varFilter’ variable, the bit pattern in the ‘filter’ field will be taken as an ‘Standard ID’ filter or ‘Extended ID’ filter.

```
struct CanFilter {
        u8          filterId;    // There are six filters so the filterId = 0 ~ 5
        u8          filterType;    // filterType = STD_ID or filterType = EXT_ID
        u32  filter;
}
```

If a filter is configured as a ‘Standard ID’ filter, only bit18 ~ bit28 in the mask take effect when filtering the CAN packet.



Mask

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	1	0
☒	☒	☒	1	1	1	1	1	1	1	1	1	1	1	☒	☒	☒	☒

Standard ID Filter

31	30	12	11	10	9	8	7	6	5	4	3	2	1	0
☒	☒	☒	☒	1	0	0	1	1	0	1	1	0	1	1

CAN Packet ID

10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	0	0	1	0

Parameters: This function takes a pointer to a variable of type ‘CanFilter’ as the parameter. Users set up the ‘filterId’. There are six filters so the ‘filterId’ could be 0 ~ 5. Filter0 and filter1 are associated with mask0. Filter2 ~ filter5 are associated with mask1.

By setting up ‘filterType’, users indicate the type of the filter. Filter type could be ‘STD_ID’ or ‘EXT_ID’.

Depending on the filter type, the ‘filter’ field in the CanFilter variable could be 0x0 ~ 0x7FF (11 bits) when filter type is ‘STD_ID’. If the filter type is ‘EXT_ID’, the ‘filter’ field in the CanFilter variable could be 0x0 ~ 0x1FFFFFFF (29 bits).

For example:

```
struct CanFilter varFilter;
i32    r  esult;
varFilter.filterId = 3;
varFilter.filterType = STD_ID;
varFilter.filter = 0x555;

result = setCanFilter(&varFilter);
if( result != 0)
      printf(“Fail to set up the filter!\n”);
```

Return Value: If this function sets the filter successfully, it returns 0,   any other returned value stands for error.

10. **Syntax:**

Full Mode Enable

Description: Enable the Function can receive 11bit and 29bit data.

Parameters: The Function suggestion Use Test mode and debug.

11. **Syntax:**

Full Mode Disable

Description: The Function is setting default.

Parameters: The Function suggestion Use Test mode and debug.

6.5.2 GPIO and Watchdog

6.5.2.1 GPIO

1. **Syntax:**

```
i32 getChLevel(u8 *val )
```

Description: Get the status of GPIO input pins and output pins, and put the value at *val.

Parameters:

This function takes a pointer to an unsigned char variable as the parameter. The bit0 ~ bit3 in the pointed variable '*val' is the status of the output pins. The bit4 ~ bit7 in the pointed variable '*val' is the status of the input pins.

For example:

```
u8 val;
i32 result;

result = getChLevel( &val);
if(result != 0)
printf("Fail to get GPIO status!\n");
```

Return Value: If the function gets the value successfully, it returns 0, any other returned value stands for error.

1. **Syntax:**

```
i32 setChLevel(u8 val )
```

Description: Set the status of GPIO Output pins.

Parameters:

This function takes an unsigned char as the parameter. The bit0 ~ bit3 in variable 'val' represent the status of the output pins. The bit3 ~ bit7 in the variable 'val' are of no use and can be neglected.

For example:

```
        u8 val = 0xf;

    i32 result;

    result = setChLevel(val);
    if(result != 0)
        printf("Fail to set GPIO!\n");
```

Return Value: If the function sets the values successfully, it returns 0, any other returned value stands for error.

6.5.2.2 Watchdog

1. **Syntax:**

```
        u8 getWtdTimer(void)
```

Description: This function read the value of the watchdog time counter and returns it to the caller.
Parameters: None.
Return Value: This function returns the value of the time counter and returns it to the caller as an unsigned character.

2. **Syntax:**

```
        void setWtdTimer( u8 val )
```

Description: This function sets the watchdog timer register to the value 'val' and starts to count down. The value could be 0 ~ 255. The unit is second. Setting the timer register to 0 disables the watchdog function and stops the countdown.
Parameters: The parameter 'val' is the value to set to watchdog timer register. The range is 0 ~ 255.
Return Value: None.

6.5.3 Power Subsystem

1. **Syntax:**

```
        i32 getPwrFwVer(struct PicInfo *ver)
```

Description: This function gets the version information of the firmware of the Power Subsystem.

Parameters: The definition of struct ‘PicInfo’ is:

```
struct PicInfo {
      u8 info[12];
}
```

This API returns the version information and store the information in the memory which is pointed at by the pointer ‘ver’.

2. **Syntax:**

```
i32 setPicDefault( void )
```

Description: The function restores the Power Subsystem to the default values. After calling this API, the items listed below are restored to its default value:

Remote Switch → Default: Disabled
Battery Monitor → Default: Disabled
Battery Voltage Delta Value → Default: 1.5V
System Soft Off Delay → Default: 5 seconds
System Hard Off Delay → Default: 1 minute
System Power On Delay → Default: 8 seconds
OS Shutdown Delay → Default: 3 minutes

Parameters: None.

Return Value: If this function works successfully, the function will return 0, any other value standards for error.

3. **Syntax:**

```
i32 getRemoteSwitch(u8 *val)
```

Description: The function gets the status of the Remote Switch.

Parameters: This function takes a pointer to an unsigned char variable as the parameter. After calling this function, the status of the Remote Switch will be put at the memory which is pointed by the parameter ‘val’. If the Remote Switch is enabled, ‘*val’ is 0x5A.

If the Remote Switch is disabled, the ‘*val’ is 0xA5. Users can use the macros ‘ENABLED’ (0x5A) and ‘DISABLED’(0xA5) to test the status value ‘*val’.

For example:

```
u8 val;
i32 result;

result = getRemoteSwitch(&val);
if(result == 0) {
      if(val == ENABLED)

            printf(“Remote Switch is enabled.\n”);
      else if( val == DISABLED )

            printf(“Remote Switch is disabled.\n”);
}
```

Return Value: If this function works successfully, it returns 0, any other value standards for error.

4. **Syntax:**

   i32 setRemoteSwitch(u8 val)

   Descriptions: The function sets the status of the Remote Switch.

   Parameters: This function takes an unsigned char as the parameter. The value of this parameter can be ‘ENABLED’ (0x5A) or ‘DISABLED’(0xA5).

   Return Value: If this function works successfully, it returns 0, any other value standards for error.

5. **Syntax:**

   i32 getBattValt(float *vol)

   Description: This function gets the battery voltage ant put it in the memory which is pointed at by the pointer ‘vol’.

Parameters: This function takes a pointer to a 'float' variable as the parameter. The reading of the battery voltage is put at the memory which is pointed at by the parameter 'vol'.

Return Value: If this function works successfully, it returns 0, any other value standards for error.

6. **Syntax:**

    i32 getBattMonitor(u8 *val)

    Description: The function gets the status of the Battery Monitor.

    Parameters: This function takes a pointer to an unsigned char variable as the parameter. After calling this function, the status of the Battery Monitor will be put at the memory which is pointed by the parameter 'val'. If the Battery Monitor is enabled, '*val' is 0x5A. If the Battery Monitor is disabled, the '*val' is 0xA5. Users can use the macros 'ENABLED' (0x5A) and

    'DISABLED'(0xA5) to test the status value '*val'.

    Return Value: If this function works successfully, it returns 0, any other value standards for error.

7. **Syntax:**

    i32 setBattMonitor(u8 val)

    Description: The function sets the status of the Battery Monitor.

    Parameters: This function takes an unsigned char as the parameter. The value of this parameter can be 'ENABLED' (0x5A) or 'DISABLED'(0xA5).

    Return Value: If this function works successfully, it returns 0, any other value standards for error.

8. **Syntax:**

    i32 getBattDelta(float *val)

Description: This function gets the delta value. The delta value is the maximum voltage deviation of the power from its nominal voltage. If the function of Battery Monitor is ON, the Power Subsystem shuts the system down when the voltage deviation of the power is larger than the delta value.

Parameters: This function takes a pointer to a float variable as the parameter. The delta value will be put at the memory which is pointed by the parameter ‘val’.

Return Value: If this function works successfully, it returns 0, any other value standards for error.

9. **Syntax:**

   i32 setBattDelta(float val)

Description: This function sets the voltage delta value. The range is 0.5V ~ 3.0V. The granularity is 0.5V.

Parameters: This function takes a float variable as the parameter.

Return Value: If this function works successfully, it returns 0, any other value standards for error.

10. **Syntax:**

    i32 setSoftOffDelay(u32 setTime)

Description: The Soft Off Delay is the interval between that the system receives a power off signal and that the system generates a power off signal. This function sets up the interval in seconds.
Parameters: The parameter is of the type of unsigned long. The value of the parameter ranges from 3~3600. The unit of the value of the parameter is seconds.

Return Value: If this function works successfully, it returns 0, any other value stands for error.

11. **Syntax:**

    i32 setHardOffDelay(u32 setTime)

Description: The Hard Off Delay is the interval between that the system is off and that the power 5VSB is off. This functions set up the interval in seconds.

Parameters: The parameter is of the type of unsigned long. The value of the parameter ranges from 3~3600. The unit of the value of the parameter is seconds.

.

Return Value: If the function works successfully, it returns 0, any other value stands for error.

12. **Syntax:**

    i32 getSoftOffDelay(u32 *Time)

Description: The Soft Off Delay is the interval between that the system receives a power off signal and that the system generates a power off signal. This function gets the interval.

Parameters: The parameter is a pointer which points to an unsigned long variable. The returned value is stored at this variable. The unit of the returned value is in seconds.

Return Value: If this function works successfully, the function returns 0, any other value stands for error.

13. **Syntax:**

    i32 getHardOffDelay(u32 *Time)

Description: The Hard Off Delay is the interval between that the system is off and that the power 5VSB is off. This function gets the interval.

Parameters: The parameter is a pointer which points to an unsigned long variable. The returned value is stored at this variable. The unit of the returned value is in seconds.

Return Value: If this function works successfully, the function returns 0, any other value stands for error.

14. **Syntax:**

    i32 getPowerOnDelay(u32 *val)

    Description: This function gets the Power On delay.

    Parameters: This function takes a pointer to an unsigned long variable as the parameter. The delay time will be put at the memory which is pointed by the ‘val’.

    Return Value: If this function works successfully, the function returns 0, any other value stands for error.

15. **Syntax:**

    i32 setPowerOnDelay(u32 val)

    Description: This function sets the Power On delay.

    Parameters: This function takes an unsigned long variable as the parameter. The range of the Power On delay is 8 ~ 60 seconds.

    Return Value: If this function works successfully, the function returns 0, any other value stands for error.

16. **Syntax:**

    i32 getShutdownDelay(u32 *val)

    Description: This function gets the Shutdown delay.
    Parameters: This function takes a pointer to an unsigned long variable as the parameter. The delay time will be put at the memory which is pointed by the parameter ‘val’.

    Return Value: If this function works successfully, the function returns 0, any other value stands for error.

17. **Syntax:**

    i32 setShutdownDelay(u32 val)

    Description: This function sets the Shutdown delay.

Parameters: This function takes an unsigned long variable as the parameter. The range of the delay is 120 ~ 3600 seconds.

Return Value: If this function works successfully, the function returns 0, any other value stands for error.

6.5.4 I-Button

1. **Syntax:**

    i32 resetIbutt(void)

    Description: This function resets the I-Button.

    Parameters: None.

    Return Value: If this function works successfully, the function returns 0, any other value stands for error.

2. **Syntax:**

    i32 readIbutt(u8 *data)

    Description: This function reads data from the I-Button.

    Parameters: This function takes a pointer to an unsigned char variable. The data to be read from the I-Button is put at the memory which is pointed by the parameter 'data'.

    Return Value: If this function works successfully, the function returns 0, any other value stands for error.

3. **Syntax:**

    i32 writeIbutt(u8 data)

    Description: This function writes command to the I-Button.

    Parameters: This function takes an unsigned char variable as the parameter. The command to be written to the I-Button is the value of the parameter ‘data’.

Return Value: If this function works successfully, the function returns 0, any other value stands for error.

Appendix A

Users have to modify the boot loader configuration to support COM6. Take the grub configuration file as an example. Add ‘8250.nr_uarts=XX noirqdebug’ at the setting of kernel. Here, XX represents the number of COM ports the system will support. Because the power subsystem connects to main system via COM6, the XX must be greater or equal to 6.

1. Modify the grub.conf.

```
[root@linux ~]# vi /boot/grub/grub.conf
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Fedora Core (2.6.27.5.117.FC10)
root (hd0,0)
kernel /vmlinuz-2.6.27.5.117.FC10 ro root=/dev/hda2 rhgb quiet
8250.nr_uarts=6 noirqdebug
initrd /initrd-2.6.27.5.117.FC10.img
```

3. List the status of the COM ports in the system.

```
# setserial -g /dev/ttyS*
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
/dev/ttyS1, UART: 16550A, Port: 0x02f8, IRQ: 3
/dev/ttyS2, UART: 16550A, Port: 0x03e8, IRQ: 11
/dev/ttyS3, UART: 16550A, Port: 0x02e8, IRQ: 10
```

```
/dev/ttyS4, UART: 16550A, Port: 0x04f8, IRQ: 11
/dev/ttyS5, UART: 16550A, Port: 0x04e8, IRQ: 10
```

The node ‘/dev/ttyS5’ corresponds to COM6. The IO port is 0x4e8, IRQ 10.