AX5234P

96/192 Bit
DIO Board

User's Manual

Disclaimers

The information in this manual has been carefully checked and is believed to be accurate. AXIOM Technology Co., Ltd. assumes no responsibility for any infringements of patents or other rights of third parties which may result from its use.

AXIOM Technology assumes no responsibility for any inaccuracies that may be contained in this document. AXIOM Technology makes no commitment to update or to keep current the information contained in this manual.

AXIOM Technology reserves the right to make improvements to this document and/or product at any time and without notice.

No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of AXIOM Technology Co., Ltd.

July 2000, Version A1
Printed in Taiwan

ESD Precautions

Integrated circuits on computer boards are sensitive to static electricity. To avoid damaging chips from electrostatic discharge, observe the following precautions:

- Do not remove boards or integrated circuits from their anti-static packaging until you are ready to install them.
- Before handling a board or integrated circuit, touch an unpainted portion of the system unit chassis for a few seconds. This helps to discharge any static electricity on your body.
- Wear a wrist-grounding strap, available from most electronic component stores, when handling boards and components.

Trademarks Acknowledgments

AXIOM is a trademark of AXIOM Technology Co., Ltd.

IBM is a registered trademark of International Business Machines Corporation.

MS-DOS, Microsoft C and QuickBasic are trademarks of Microsoft Corporation.

TURBO C is a trademark of Inprise Inc.

BASIC is a trademark of Dartmouth College.

Intel is a trademark of Intel Corporation.

Other brand names and trademarks are the properties and registered brands of their respective owners.

Unpacking

The AX5234P is packed in an anti-static bag. The PCI Bus board has components that are easily damaged by static electricity. Do not remove the anti-static wrapping until proper precautions have been taken. Safety instructions in front of this User's Manual describe anti-static precautions and procedures.

After unpacking the PCI Bus board, place it on a raised surface and carefully inspect the board for any damage that might have occurred during shipment. Ground the board and exercise extreme care to prevent damage to the board from static electricity.

Integrated circuits will sometimes come out of their sockets during shipment. Examine all integrated circuits, to ensure that they are firmly seated. The AX5234P 96 Bit DIO Board package includes the following:

- AX5234P Board
- Flat cable 50p 1M x 4
- AS59099 DAC Driver CD
- AX5234P user's manual
- Warranty card

The AX5234A (optional) 96 Bit DIO Extension Card package includes the following:

- AX5234A Board
- Flat cable 50p 1M x 4
- Flat cable 30p 6CM x 1
- Warranty card

Make sure that all of the items listed above are present.

What To Do If There Is A Problem

If there are damaged or missing parts, contact your supplier and/or dealer immediately. Do not attempt to apply power to the board if there is damage to any of its components.

Table of Contents

Chapter 1 Introduction

1.1 General Description 1
1.2 Applications 1
1.3 Specifications 2
1.3.1 Input and Output 2
1.3.2 Electrical Characteristics.. 2
1.3.3 Interface Characteristics 2
1.3.4 Power Requirements 2
1.3.5 Physical/Environmental 2
1.4 Accessories Guide 3

Chapter 2 Board Configuration and Installation

2.1 Locator Diagram...5
2.2 Jumper Settings 7
2.2.1 JP1.. 7
2.2.2 J1 ~ J4.. 7
2.3 Connector Pin Assignments 8
2.3.1 CON1 Pin Assignments... 8
2.3.2 CON2 ~ CON4 Pin Assignments 9
2.4 Hardware Installation................................10
2.4.1 Board Installation 10
2.4.2 AX5234P Extension Board Cable Connection 11

Chapter 3 Register Format and Description

3.1 I/O Address Mapping....................................13
3.2 Register Description 15
Base address + 0 (write) 15
Base address + 2 (write) 15
Base address + 3 (write) 15
Base address + 7 (read)...15
3.2.1 Group_0 registers 15
Base address + 0xc0 1 Port A..15
Base address + 0xc4 1 Port B..16
Base address + 0xc8 1 Port C..16
Base address + 0xcc Control Register...16
Base address + 0xd0 2 Port A 16
Base address + 0xd4 2 Port B 16
Base address + 0xd8 2 Port C 16
Base address + 0xdc Control Register...16
Base address + 0xe0 3 Port A 17
Base address + 0xe4 3 Port B 17

Base address + 0xe8 3 Port C 17
Base address + 0xec..17
Base address + 0xf0 4 Port A 17
Base address + 0xf4 4 Port B 17
Base address + 0xf8 4 Port C 17
Base address + 0xfc Control Register 18
3.2.2 Group_2 registers (AX5234A)..................................... 18
Base address + 0xc0 5 Port A...18
Base address + 0xc4 5 Port B...18
Base address + 0xc8 5 Port C...18
Base address + 0xcc Control Register...18
Base address + 0xd0 6 Port A 19
Base address + 0xd4 2 Port B 19
Base address + 0xd8 2 Port C 19
Base address + 0xdc Control Register...19
Base address + 0xe0 7 Port A 19
Base address + 0xe4 7 Port B 19
Base address + 0xe8 7 Port C 19
Base address + 0xec..20
Base address + 0xf0 8 Port A 20
Base address + 0xf4 8 Port B 20
Base address + 0xf8 8 Port C 20
Base address + 0xfc Control Register 20

Chapter 4 Device Driver

4.1 How to Install the Device Driver 21
4.1.1 Example 21

Chapter 5 Examples

5.1 Turbo C 25
5.2 Turbo PASCAL...26
5.3 Qbasic 4.5 27
5.4 Application 28
5.4.1 Demo 1 28
5.4.2 Demo 2 29

Appendix A Block Diagram

Chapter 1
Introduction

1.1 General Description

The AX5234P is a 96 bits digital input and output board that plugs into the computer via PCI (Peripheral Component) slot which can be extended to 192 digital I/O channels by connecting with its extension board AX5234A. The board can be used with TTL low-level input/output circuitry or with solid state relay module such as AX1416 or AX1424, and provides 2500V isolation for interfacing with high level AC and DC signals.

The 96 digital I/O lines are arranged into 4 separated connectors. Each connector is further divided into three 8-bit ports. These ports can be functionally programmed as either digital input or digital output ports.

There is a unique feature associated with AX5234P: users can select the extension card AX5234A to add DIO channels and become 192 bits.

1.2 Applications

- Sense and control high level signals through I/O module
- Sense low-level (TTL) switches or signals
- Drive indicator light or control recorders
- Parallel data transfer to PC

1.3 Specifications

1.3.1 Input and Output

Input/Output Lines	96
Input/Output Mode	Pair
Improved Noise Margins	Hysteresis VT+-VT-=0.4 (TYP.)
Input/Output Level	TTL/DAT Compatible

1.3.2 Electrical Characteristics

VIH	2V (min.)
VIL	0.8V (min.)
IIH	20µA (max.) at VI=2.7V
IIL	-0.2mA (max.) at VIL=0.4V
VOH	2.4V (max.) at IOL=-3mA
VOL	0.4V (max.) at IOL=12mA
IOH	-15mA (max.)
IOL	24mA (max.)

1.3.3 Interface Characteristics

I/O Connector	50 pin mating header
I/O Cable Type	
Ribbon Twisted Pair Cable	Zo=50 to 100Ω (TYP.)
Ribbon Twisted Pair Cable	Zo=30 to 80Ω (TYP.)
Compatible Bus	PCI bus
Data Path	8 bits
Configured Address and Interrupt	Plug & Play

1.3.4 Power Requirements

+5VDC	2.0A (max.)

1.3.5 Physical/Environmental

Dimensions	175 x 106 (mm)
Operating Temperature Range	0°C to 60°C
Storage Temperature Range	-25°C to 85°C
Relative Humidity	Up to 90%; non-condensing

1.4 Accessories Guide

- **AX951A**
  Screw terminal board for all digital I/O connection. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.
- **AX754**
  24-channel opto-isolated D/I panel for signal connection and conditioning with AX5234P digital I/O connections. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.
- **AX756**
  24-channel electromechanical single-pole, double-throw (SPDT) that can be driven by the AX5234P. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.
- **AX755**
  8-channel electromechanical single-pole, double-throw (SPDT) and 16-channel opto-isolated digital I/O panel compatible with AX5234P digital I/O connections. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.
- **AX755/24**
  24-channel Relay Actuator and 24-channel Opto-isolated digital input panel compatible with AX5234P digital I/O connections. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.
- **AX5234A**
  This is the AX5234P extension board. AX5234P can be extended to 192 digiatl I/O channels by connecting an AX5234A.

This page does not contain any information.

Chapter 2

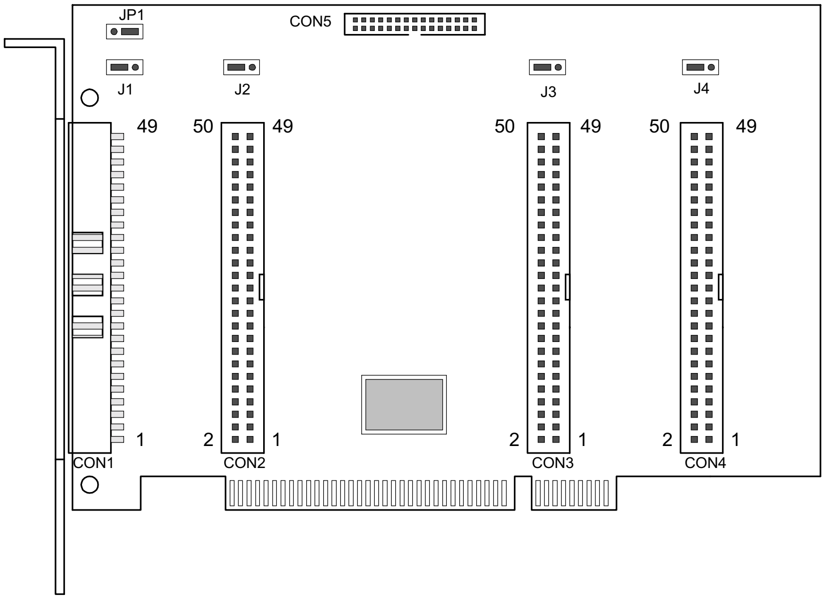
Board Configuration and Installation

WARNING: *When power is ON, hazardous voltages may be present in the AX5234P, do not touch the board and its wiring to prevent shock hazard*

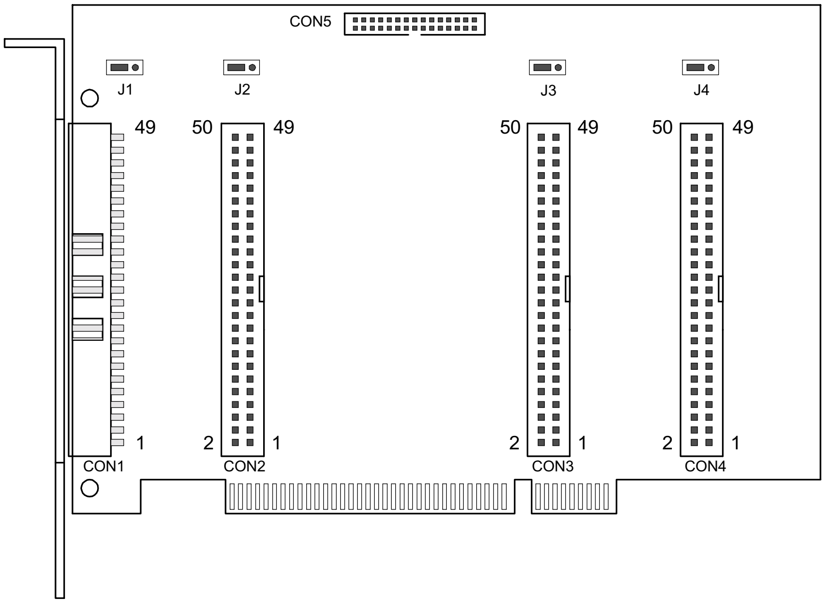
2.1 Locator Diagram

The following figure shows the jumper and connector locations onboard the AX5234P and AX5234A.

AX5234P

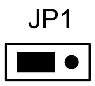


AX5234A



2.2 Jumper Settings

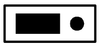
2.2.1 JP1



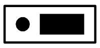
Connect CON1 pin50 to DGND

2.2.2 J1 ~ J4

J1(CON1),J2(CON2),J3(CON3),J4(CON4)



Connect CONx pin2,4 to GND



Connect CONx pin2,4 to +12V

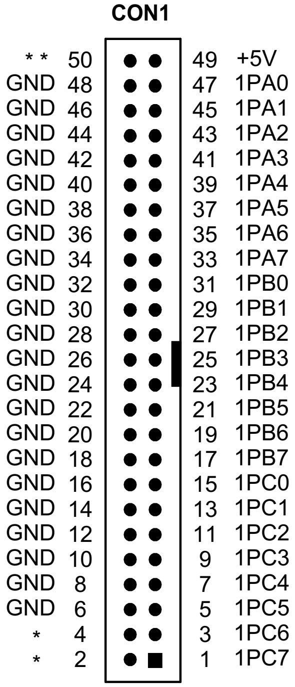
Jumpering +12V PC power to the connector allows the AX5234P to provide +12V power for direct relay driving voltage or input pull high voltage. When using the AX5234P with a standard Opto-22 interface panel board, pins 2 and 4 of AX5234P's 50-pin connector must be connected to ground.

NOTE: *As the AX5234P is connected to other boards/panels through the six 50-pin connectors, users must check with the pins of the boards/panels and make sure that these jumpers are set to the proper side.*

2.3 Connector Pin Assignments

The AX5234P PCI card has four I/O connectors, labeled CON1 through CON4, that are accessible from the expansion slot rear panel of the PC. The pin assignments for CON1 through CON4 are listed on the following subsections.

2.3.1 CON1 Pin Assignments



CON1

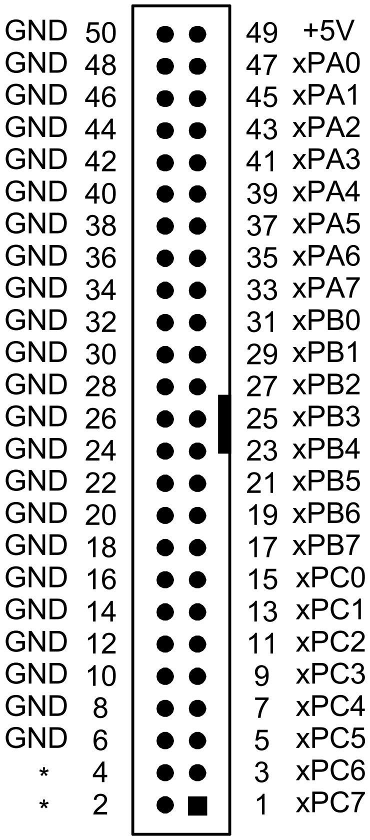
* *	50	49	+5V
GND	48	47	1PA0
GND	46	45	1PA1
GND	44	43	1PA2
GND	42	41	1PA3
GND	40	39	1PA4
GND	38	37	1PA5
GND	36	35	1PA6
GND	34	33	1PA7
GND	32	31	1PB0
GND	30	29	1PB1
GND	28	27	1PB2
GND	26	25	1PB3
GND	24	23	1PB4
GND	22	21	1PB5
GND	20	19	1PB6
GND	18	17	1PB7
GND	16	15	1PC0
GND	14	13	1PC1
GND	12	11	1PC2
GND	10	9	1PC3
GND	8	7	1PC4
GND	6	5	1PC5
*	4	3	1PC6
*	2	1	1PC7

* : GND or +12V selectable by J1

Pin Name	Description (CON1)
1PA0-1PA7	Port A 0~7
1PB0-1PB7	Port B 0~7
1PC0-1PC7	Port C 0~7
GND	Ground

2.3.2 CON2 ~ CON4 Pin Assignments

CON x



GND	50	49	+5V
GND	48	47	xPA0
GND	46	45	xPA1
GND	44	43	xPA2
GND	42	41	xPA3
GND	40	39	xPA4
GND	38	37	xPA5
GND	36	35	xPA6
GND	34	33	xPA7
GND	32	31	xPB0
GND	30	29	xPB1
GND	28	27	xPB2
GND	26	25	xPB3
GND	24	23	xPB4
GND	22	21	xPB5
GND	20	19	xPB6
GND	18	17	xPB7
GND	16	15	xPC0
GND	14	13	xPC1
GND	12	11	xPC2
GND	10	9	xPC3
GND	8	7	xPC4
GND	6	5	xPC5
*	4	3	xPC6
*	2	1	xPC7

* : GND or +12V selectable by Jx

Pin Name	Description (CON2-CON4)
xPA0-xPA7	CONx Port A 0~7
xPB0-xPB7	CONx Port B 0~7
xPC0-xPC7	CONx Port C 0~7
GND	Ground

2.4 Hardware Installation

The AX5234P boards are shipped with protective electrostatic cover. When unpacking, touch the board's electrostatically shielded packing with the metal frame of your computer to discharge the accumulated static electricity prior to touching the board.

The following section summarizes the procedures when installing AX5234P:

WARNING: *Turn OFF the PC and all accessories connected to the PC whenever installing or removing any peripheral board including the AX5425(0) series board.*

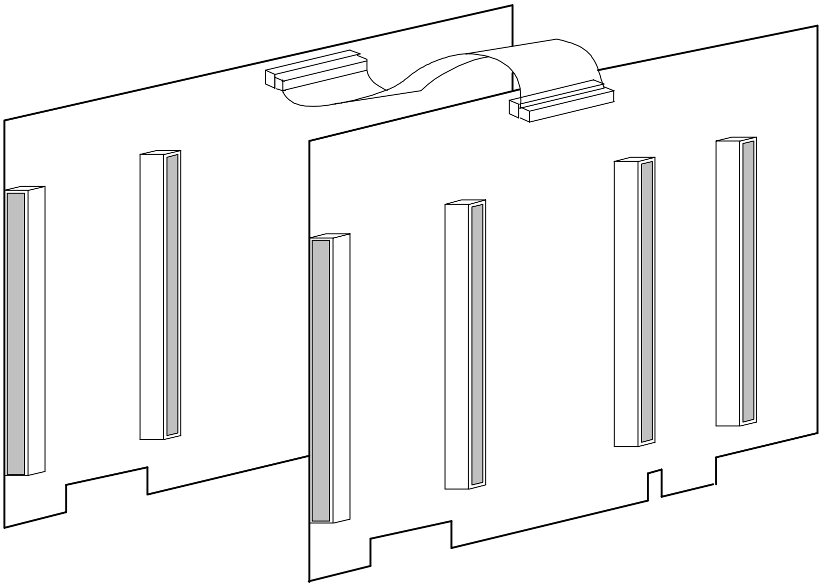
2.4.1 Board Installation

The following lists the instructions to follow when installing the AX5234P card.

1. Turn OFF the PC and all accessories power.
2. Unplug all power cords and entire cables from the rear of the PC.
3. Remove the PC's cover (see your PC Operation Guide if you are not skillful about it).
4. Find an unused expansion slot. Remove the blank expansion slot cover and save the screw for affixing retaining bracket.
5. Grab the upper edge of the AX5234P board. Align the AX5234P board's retaining bracket with the expansion slot rear panel, and straighten the board's gold finger with the expansion slot. Gently push the board into slot.
6. Restore the screw to the expansion slot-retaining bracket.
7. Replace the PC's cover and connect the cables you detached in step2.
8. Turn ON the power of the PC and other peripheral device.

2.4.2 AX5234P Extension Board Cable Connection

Refer to the following illustration for the proper cable connection from AX5234P card to the AX5234A extension board.



This page does not contain any information.

Chapter 3

Register Format and Description

3.1 I/O Address Mapping

The AX5234P uses some non-consecutive addresses in I/O space. All registers are 8 bits wide. The base address (or starting address) is determined during the installation by CPU auto setting. This chapter describes each register's format and functions. Each register can be accessed easily by using direct I/O instructions of any application language available (Assembly, Basic, Pascal, C, etc.). Don't operate any other UNLISTED I/O, or else it will result in errors.

Location	Function	Type
Base Address + 0	Enable external reset	W
Base Address + 2	Group select control	W
Base Address + 3	Group select control	W
Base Address + 7	Group select control register status	R

GROUP 0

Location	Function	Type
Base Address + 0xc0	1PA0 ~ 1PA7	R/W
Base Address + 0xc4	1PB0 ~ 1PB7	R/W
Base Address + 0xc8	1PC0 ~ 1PC7	R/W
Base Address + 0xcc	Control Register 1	W
Base Address + 0xd0	2PA0 ~ 2PA7	R/W
Base Address + 0xd4	2PB0 ~ 2PB7	R/W
Base Address + 0xd8	2PC0 ~ 2PC7	R/W
Base Address + 0xdc	Control Register 2	W
Base Address + 0xe0	3PA0 ~ 3PA7	R/W
Base Address + 0xe4	3PB0 ~ 3PB7	R/W
Base Address + 0xe8	3PC0 ~ 3PC7	R/W
Base Address + 0xec	Control Register 3	W

Continued . . . . .

Location	Function	Type
Base Address + 0xf0	4PA0 ~ 4PA7	R/W
Base Address + 0xf4	4PB0 ~ 4PB7	R/W
Base Address + 0xf8	4PC0 ~ 4PC7	R/W
Base Address + 0xfc	Control Register 4	W

GROUP 2 (AX5234A)

Location	Function	Type
Base Address + 0xc0	1PA0 ~ 1PA7	R/W
Base Address + 0xc4	1PB0 ~ 1PB7	R/W
Base Address + 0xc8	1PC0 ~ 1PC7	R/W
Base Address + 0xcc	Control Register 1	W
Base Address + 0xd0	2PA0 ~ 2PA7	R/W
Base Address + 0xd4	2PB0 ~ 2PB7	R/W
Base Address + 0xd8	2PC0 ~ 2PC7	R/W
Base Address + 0xdc	Control Register 2	W
Base Address + 0xe0	3PA0 ~ 3PA7	R/W
Base Address + 0xe4	3PB0 ~ 3PB7	R/W
Base Address + 0xe8	3PC0 ~ 3PC7	R/W
Base Address + 0xec	Control Register 3	W
Base Address + 0xf0	4PA0 ~ 4PA7	R/W
Base Address + 0xf4	4PB0 ~ 4PB7	R/W
Base Address + 0xf8	4PC0 ~ 4PC7	R/W
Base Address + 0xfc	Control Register 4	W

3.2 Register Description

Base address + 0 (write)

X	X	X	X	X	X	X	RST

RST = 0 disable RESET function on AX5234P
RST = 1 enable RESET function on AX5234P, when system (PC) reset, the AX5234P will be reset.

Base address + 2 (write)

X	X	X	INT	X	SG2	SG1	SG0

When the start or initiation of the program, users must output a value 0x07(or 0x0f) to Base Address + 0x02 to set the INT signal as input, and the SG0, SG1, SG2 as output from the PCI bridge to the local functions.

Base address + 3 (write)

X	X	X	X	X	SG2	SG1	SG0

SG2, SG1, SG0 are low actions

SG2	SG1	SG0	Description
1	1	0	Select/Enable Group 0
1	0	1	Select/Enable Group 1
0	1	1	Select/Enable Group 2

Base address + 7 (read)

X	X	X	IRQ	X	SG2	SG1	SG0

This group selects control register status.

3.2.1 Group_0 registers

NOTE: *When users program the registers in Group_0, they must check if the I/O port Base address + 2 already sent an output value 0x07(or 0x0f), and the I/O port Base address + 3 has an output value 0x06 to enable functions in Group_0.*

Base address + 0xc0 1 Port A

D7	D6	D5	D4	D3	D2	D1	D0
1PA7	1PA6	1PA5	1PA4	1PA3	1PA2	1PA1	1PA0

Base address + 0xc4   1 Port B

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1PB7	1PB6	1PB5	1PB4	1PB3	1PB2	1PB1	1PB0

Base address + 0xc8   1 Port C

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1PC7	1PC6	1PC5	1PC4	1PC3	1PC2	1PC1	1PC0

Base address + 0xcc   Control Register

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
X	X	X	D4	D3	X	D1	D0

1 Port A
1 Port C (Upper)
1 Port B
1 Port C (Lower)

0=Output, 1=Input, X=Don't care

Base address + 0xd0   2 Port A

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
2PA7	2PA6	2PA5	2PA4	2PA3	2PA2	2PA1	2PA0

Base address + 0xd4   2 Port B

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
2PB7	2PB6	2PB5	2PB4	2PB3	2PB2	2PB1	2PB0

Base address + 0xd8   2 Port C

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
2PC7	2PC6	2PC5	2PC4	2PC3	2PC2	2PC1	2PC0

Base address + 0xdc   Control Register

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
X	X	X	D4	D3	X	D1	D0

2 Port A
2 Port C (Upper)
2 Port B
2 Port C (Lower)

0=Output, 1=Input, X=Don't care

Base address + 0xe0   3 Port A

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
3PA7	3PA6	3PA5	3PA4	3PA3	3PA2	3PA1	3PA0

Base address + 0xe4   3 Port B

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
3PB7	3PB6	3PB5	3PB4	3PB3	3PB2	3PB1	3PB0

Base address + 0xe8   3 Port C

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
3PC7	3PC6	3PC5	3PC4	3PC3	3PC2	3PC1	3PC0

Base address + 0xec

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
X	X	X	D4	D3	X	D1	D0

3 Port A
3 Port C (Upper)
3 Port B
3 Port C (Lower)

0=Output, 1=Input, X=Don't care

Base address + 0xf0   4 Port A

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
4PA7	4PA6	4PA5	4PA4	4PA3	4PA2	4PA1	4PA0

Base address + 0xf4   4 Port B

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
4PB7	4PB6	4PB5	4PB4	4PB3	4PB2	4PB1	4PB0

Base address + 0xf8   4 Port C

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
4PC7	4PC6	4PC5	4PC4	4PC3	4PC2	4PC1	4PC0

Base address + 0xfc   Control Register

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
X	X	X	D4	D3	X	D1	D0

4 Port A
4 Port C (Upper)
4 Port B
4 Port C (Lower)

0=Output, 1=Input, X=Don't care

3.2.2 Group_2 registers (AX5234A)

NOTE: *When users program the registers in Group_2, they must check if the I/O port Base address + 2 has an output value 0x07(or 0x0f), and the I/O port Base address + 3 has an output value 0x03 to enable the functions in Group_0.*

Base address + 0xc0   5 Port A

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
5PA7	5PA6	5PA5	5PA4	5PA3	5PA2	5PA1	5PA0

Base address + 0xc4   5 Port B

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
5PB7	5PB6	5PB5	5PB4	5PB3	5PB2	5PB1	5PB0

Base address + 0xc8   5 Port C

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
5PC7	5PC6	5PC5	5PC4	5PC3	5PC2	5PC1	5PC0

Base address + 0xcc   Control Register

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
X	X	X	D4	D3	X	D1	D0

5 Port A
5 Port C (Upper)
5 Port B
5 Port C (Lower)

0=Output, 1=Input, X=Don't care

Base address + 0xd0 6 Port A

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
6PA7	6PA6	6PA5	6PA4	6PA3	6PA2	6PA1	6PA0

Base address + 0xd4 2 Port B

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
6PB7	6PB6	6PB5	6PB4	6PB3	6PB2	6PB1	6PB0

Base address + 0xd8 2 Port C

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
6PC7	6PC6	6PC5	6PC4	6PC3	6PC2	6PC1	6PC0

Base address + 0xdc Control Register

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
X	X	X	D4	D3	X	D1	D0

6 Port A
6 Port C (Upper)
6 Port B
6 Prt C (Lower)

0=Output, 1=Input, X=Don't care

Base address + 0xe0 7 Port A

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
7PA7	7PA6	7PA5	7PA4	7PA3	7PA2	7PA1	7PA0

Base address + 0xe4 7 Port B

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
7PB7	7PB6	7PB5	7PB4	7PB3	7PB2	7PB1	7PB0

Base address + 0xe8 7 Port C

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
7PC7	7PC6	7PC5	7PC4	7PC3	7PC2	7PC1	7PC0

Base address + 0xec

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
X	X	X	D4	D3	X	D1	D0

7 Port A
7 Port C (Upper)
7 Port B
7 Port C (Lower)

0=Output, 1=Input, X=Don't care

Base address + 0xf0   8 Port A

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
8PA7	8PA6	8PA5	8PA4	8PA3	8PA2	8PA1	8PA0

Base address + 0xf4   8 Port B

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
8PB7	8PB6	8PB5	8PB4	8PB3	8PB2	8PB1	8PB0

Base address + 0xf8   8 Port C

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
8PC7	8PC6	8PC5	8PC4	8PC3	8PC2	8PC1	8PC0

Base address + 0xfc   Control Register

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
X	X	X	D4	D3	X	D1	D0

8 Port A
8 Port C (Upper)
8 Port B
8 Port C (Lower)

0=Output, 1=Input, X=Don't care

Chapter 4

Device Driver

Device driver is suitable for Plug & Play in DOS environment to get some information from PCI BIOS. This chapter describes in detail on how to install the device driver and use the device driver command to get base address, IRQ level, slot number. Also examples programs are provided only for reference.

After getting their information successfully, you can use the information to act as parameter for driver function described on the next chapter. All operations within this section will not work unless the device driver AX5234.SYS is successfully installed.

4.1 How to Install the Device Driver

Before executing any application program (including the following examples), the device driver below must be installed.

```
SETUP [SOURCEDRIVE]  [TARGET DRIVE]  [DIRECTORY]
```

This will copy the device driver into your designated directory. And then add the following command line to your config.sys:

```
DEVICE = [PATH] AX5234.SYS
```

4.1.1 Example

If you insert this diskette in drive A: and want to copy the file into C:\AX5234P. You must execute the following command line from the DOS prompt.

```
A:\> SETUP   A:  C:   AX5234P  [ENTER]
```

And then you must add the following line in your config.sys file.

```
DEVICE = C:\AX5234P\AX5234.SYS
```

Reboot your computer.

If there is any AX5234P plugged in your system, the following message will appear:

```
**************************************************************************************
*                Copyright 2000 by AXIOM Technology Co., Ltd                         *
*                                                           Ver 1.0                  *
*                     AX5234 DEVICE DRIVER INSTALLED                                 *
**************************************************************************************
```

Now AX5234P acts like a file. You can OPEN, CLOSE, WRITE (command), READ (base address, IRQ level, slot number) it via this device driver.

If there is no AX5234P in your system, the following message will appear:

```
AX5234P or PCI BIOS NOT FOUND!!
Any OPEN to device driver will fail!
```

The device driver allows user to generate the BASE ADDRESS, IRQ LEVEL, SLOT NUMBER of the AX5234P plugged in your system. Before accessing the device driver, open it is needed. And after accessing the device driver, close it also as required. To get any information (BASE ADDRESS, IRQ LEVEL, SLOT NUMBER), first you must Write Command to device driver for the needed data to be read from device driver.

There are three commands for user to get base address, IRQ level and slot number. The number following the command indicates card number. To get base address, you must write the command string "B?" to Device driver and then read a word (two bytes) from device driver. This is the base address you need. To get IRQ level, you must write the command string "I?" to device driver and then read a word (two bytes) from device driver. This is the IRQ level you need. To get a slot number, you must write the command String "S?" to device driver and then read a WORD (two Bytes) from device driver. This is the slot number you need.

Please note that the question mark '?' must be replace with card number. If base address returns to 0, it means all information retrieved by that card number are not available.

NOTE: *This device driver supports programs written in Microsoft QuickBasic, Microsoft C, Borland Turbo C, and Turbo, Pascal.*

This page does not contain any information.

Chapter 5

Examples

5.1 Turbo C

```
**********************************************************************************
*                    Example program for turbo C language                        *
*                         To get BASE ADDRESS                                    *
*                              IRQ LEVEL                                         *
*                     SLOT NUMBER via device driver                              *
*                Before executing this program, device driver                    *
*                       must be installed successfully.                          *
**********************************************************************************
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <fcntl.h>
#include <io.h>
main()
{
   int fd;
   int base,slotno,irqno;
   unsigned int i,j,dat;
   if ( ( fd=open ("5234drv",O_RDWR) ) == -1 )
    {
      printf("AX5234P open fail ! \n");
      exit(0);
     }
   else      printf("OK\n");
   write(fd,"b1",2);
   read(fd,&base,sizeof(int));
   write(fd,"i1",2);
   read(fd,&irqno,sizeof(int));
   write(fd,"s1",2);
   read(fd,&slotno,sizeof(int));
   close(fd);
   printf("BASE ADDRESS: %x\n",base);
   printf("IRQ LEVEL: %x\n",irqno);
   printf("SLOT NUMBER: %x\n",slotno);
   if(base==0)
```

```
     {
       printf("ERROR INFORMATION!\n");
       exit(0);
      }
    }
```

5.2 Turbo PASCAL

```
*************************************************************************************
*                   Example program for Turbo PASCAL language                      *
*                            To get BASE ADDRESS                                    *
*                                 IRQ LEVEL                                         *
*                       SLOT NUMBER via device driver                              *
*                      Before executing this program, device                       *
*                      driver must be installed successfully.                      *
*************************************************************************************
   PROGRAM TP_DEMO(input,output);
   uses dos,crt;
       var
       fdw:text;
       fdr:file of integer;
       addr,irqno,slotno:integer;
   begin
       clrscr;
       assign(fdw,'5234drv');
       assign(fdr,'5234drv');
       rewrite(fdw);
       writeln(fdw,'b1');
       reset(fdr);
       read(fdr,addr);
       rewrite(fdw);
       writeln(fdw,'i1');
       reset(fdr);
       read(fdr,irqno);
       rewrite(fdw);
       writeln(fdw,'s1');
       reset(fdr);
       read(fdr,slotno);
       close(fdw);
       close(fdr);
     writeln('BASE ADDRESS:',addr:10);
     writeln('IRQ NUMBER  :',irqno:10);
     writeln('SLOT NUMBER :',slotno:10);
```

```
    if addr <> 0 then writeln('The information are correct');
    end
```

5.3 Qbasic 4.5

```
****************************************************************************************
*                       Example Program for QB45 language                              *
*                            To get BASE ADDRESS                                       *
*                                 IRQ LEVEL                                            *
*                         SLOT NUMBER via device driver                                *
*                    Before executing this program, device driver                      *
*                          must be installed successfully.                             *
****************************************************************************************
  OPEN "5234DRV" FOR OUTPUT AS #1
  OPEN "5234DRV" FOR BINARY AS #2
  PRINT #1,"B1"
  GET #2,1,BL%
  GET #2,1,BH%
  PRINT #1,"I1"
  GET #2, ,I%
  PRINT #1,"S1"
  GET #2, ,S%
  CLOSE #1
  CLOSE #2
  BL=BL%
  BH=BH%
  ADDR=BH*256+BL
  PRINT "BASE ADDRESS:",ADDR
  PRINT "IRQ LEVEL:",I%
  PRINT "SLOT NUMBER :",S%
  IF ADDR <> 0 THEN PRINT "The information are correct"
```

5.4 Application

5.4.1 Demo 1

```
/************************************************************************************/
/*                  This program test AX5234's DIO function.                        */
/************************************************************************************/

#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <fcntl.h>
#include <io.h>
int base,busno,irqno;
main()
{
 int fd;
 unsigned char   i;
 if((fd=open("5234drv",O_RDWR))==-1)
   {
    printf("AX5234P open fail! \n");
    exit(0);
    }
else
    printf("ok\n");
write(fd,"b1",2);
read(fd,&base,sizeof(int));
write(fd,"i1",2);
read(fd,&irqno,sizeof(int));
write(fd,"s1",2);
read(fd,&busno,sizeof(int));
close(fd);
printf("base address: %x\n",base);
printf("inrqlevel: %x\n",irqno);
printf("slot number: %x\n",busno);
if(base==0)
  {
  printf("error information!\n");
  exit(0);
  }
outportb(base+5,0x00);       /* disable interrupt */
outportb(base+0,0x01);       /* enable reset */
outportb(base+2,0x07);      /* Group select control */
outportb(base+3,0x06);      /* Group select control , select Group 0 */
outportb(base+0xcc,0xff);   /* Set Port A,B,C to be Input */
while(!kbhit()) {
i=inportb(base+0xc4);
 printf("%x\n",i);

}
}
```

5.4.2 Demo 2

```
/**************************************************************************************************************************/
/*          This program tests AX5234's DIO function.                         */
/*          Connect con1 and con2 with 50-pin flat cable.                     */
/*          Connect con3 and con4 with 50-pin flat cable.                     */
/*          The program outputs data from group1 to group2                    */
/*          and from group3 to group4 and than inverted.                      */
/*          If any error, print " Error ! Error ! " and                       */
/*          exit the program.                                                 */
/**************************************************************************************************************************/

#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <fcntl.h>
#include <io.h>
int base,busno,irqno;
main()
{
 int fd;
 unsigned int i,j,k,ii,jj,kk;
 if((fd=open("5234drv",O_RDWR))==-1)
   {
    printf("AX5234P open fail! \n");
    exit(0);
    }
else
    printf("ok\n");
write(fd,"b1",2);
read(fd,&base,sizeof(int));
write(fd,"i1",2);
read(fd,&irqno,sizeof(int));
write(fd,"s1",2);
```

```
read(fd,&busno,sizeof(int));
close(fd);
printf("base address: %x\n",base);
printf("inrqlevel: %x\n",irqno);
printf("slot number: %x\n",busno);
if(base==0)
  {
  printf("error information!\n");
  exit(0);
  }
outportb(base+5,0x00);     /* disable interrupt */
outportb(base+0,0x01);     /* enable reset */
outportb(base+2,0x07);     /* Group select control  */
outportb(base+3,0x06);     /* Group select control, select Group 0  */
clrscr();
while(!kbhit()) {
  outportb(base+0xcc,0x00);   /* set connector1 ------  outport  */
  outportb(base+0xdc,0xff);    /* set connector2 ------  inport   */
  outportb(base+0xec,0x00);   /* set connector3 ------  outport  */
  outportb(base+0xfc,0xff);     /* set connector4 ------  inport   */

  gotoxy(32,10);printf("G1PA --> G2PA");
  for(i=0;i<256;i++) {
    outportb(base+0xc0,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xd0));
    delay(150);
    if((char)i != inportb(base+0xd0)) {
     printf("\nError ! Error !");
     exit();
    }
  }

  gotoxy(32,10);printf("G1PB --> G2PB");
  for(i=0;i<256;i++) {
    outportb(base+0xc4,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xd4));
    delay(150);
    if((char)i != inportb(base+0xd4)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  gotoxy(32,10);printf("G1PC --> G2PC");
  for(i=0;i<256;i++) {
    outportb(base+0xc8,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xd8));
    delay(150);
    if((char)i != inportb(base+0xd8)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  gotoxy(32,10);printf("G3PA --> G4PA");
  for(i=0;i<256;i++) {
    outportb(base+0xe0,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xf0));
    delay(150);
    if((char)i != inportb(base+0xf0)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  gotoxy(32,10);printf("G3PB --> G4PB");
  for(i=0;i<256;i++) {
    outportb(base+0xe4,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xf4));
    delay(150);
    if((char)i != inportb(base+0xf4)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  gotoxy(32,10);printf("G3PC --> G4PC");
  for(i=0;i<256;i++) {
    outportb(base+0xe8,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xf8));
    delay(150);
    if((char)i != inportb(base+0xf8)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  /*******************************************************************/

  outportb(base+0xcc,0xff);   /* set connector1 ------  inport  */
  outportb(base+0xdc,0x00);  /* set connector2 ------  outport   */
  outportb(base+0xec,0xff);    /* set connector3 ------  inport  */
  outportb(base+0xfc,0x00);   /* set connector4 ------  outport   */

  gotoxy(32,10);printf("G2PA --> G1PA");
  for(i=0;i<256;i++) {
    outportb(base+0xd0,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xc0));
    delay(150);
    if((char)i != inportb(base+0xc0)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  gotoxy(32,10);printf("G2PB --> G1PB");
  for(i=0;i<256;i++) {
    outportb(base+0xd4,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xc4));
    delay(150);
    if((char)i != inportb(base+0xc4)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  gotoxy(32,10);printf("G2PC --> G1PC");
  for(i=0;i<256;i++) {
    outportb(base+0xd8,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xc8));
    delay(150);
    if((char)i != inportb(base+0xc8)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  gotoxy(32,10);printf("G4PA --> G3PA");
  for(i=0;i<256;i++) {
    outportb(base+0xf0,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xe0));
    delay(150);
    if((char)i != inportb(base+0xe0)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  gotoxy(32,10);printf("G4PB --> G3PB");
```

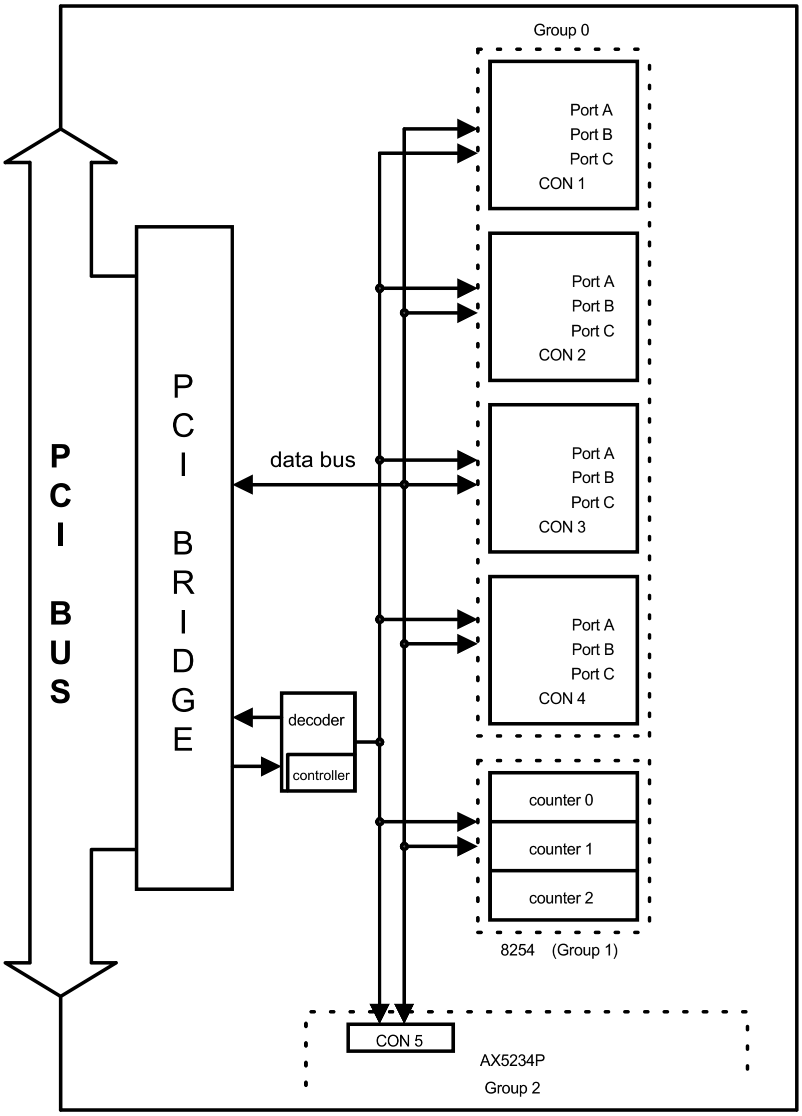
```
  for(i=0;i<256;i++) {
    outportb(base+0xf4,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xe4));
    delay(150);
    if((char)i != inportb(base+0xe4)) {
     printf("\nError ! Error !");
     exit();
    }
   }

  gotoxy(32,10);printf("G4PC --> G3PC");
  for(i=0;i<256;i++) {
    outportb(base+0xf8,i);
    gotoxy(32,11);printf("%3x      %3x",i,inportb(base+0xe8));
    delay(150);
    if((char)i != inportb(base+0xe8)) {
     printf("\nError ! Error !");
     exit();
    }
   }
 }
}
```

This page does not contain any information.

Appendix A

Block Diagram



This page does not conatain any information.