

AX10420
48 Bits DIO Module
User's Manual

Disclaimers

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

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

AXIOMTEK 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 AXIOMTEK Co., Ltd.

Copyright© 1994 by AXIOMTEK Co., Ltd.

All rights reserved.

November 1994, 2nd Edition

Printed in Taiwan

Trademarks Acknowledgments

AXIOMTEK is a trademark of AXIOMTEK 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 Borland 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.

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.

Unpacking

The AX10420 is packed in an anti-static bag. The 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.

Inventory and Inspection

After unpacking the 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, particularly the BIOS, processor and keyboard controller chip to ensure that they are firmly seated.

The AX10420 48 Bits DIO Module package includes the following:

- AX10420 Board
- Screw 3mm (x4)
- Bronze stick 6mm (x4)
- AS59099 DAC Driver CD

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
1.1 General Description.....	1
1.2 Applications	1
1.3 Specifications	2
1.4 Accessories Guide	3
Chapter 2 Module Configuration and Installation.....	4
2.1 Component Locator Diagram	4
2.2 Base Address Switch	5
2.3 +12V or Ground Selection Jumper.....	6
2.4 IRQ Level Jumper	7
2.5 Interface Setting Jumper.....	7
2.6 Connector Pin Assignments.....	9
2.7 Resistor Pack.....	10
2.8 Hardware Description.....	11
2.9 Module Installation	11
Chapter 3 Register Structure and Format	13
3.1 AX10420 I/O Address Map	13
3.2 AX10420 Register Description.....	14
Chapter 4 Programming.....	16
4.1 Digital Input and Output.....	16
4.2 Interrupt.....	17
Chapter 5 Application.....	22
5.1 Event Trigger	22
5.2 Polling 4*4 Keypad	25
Appendix A PC I/O Port Mapping	28
Appendix B Block Diagram	29
Appendix C Technical Reference	30
Appendix D PC/104 Mechanical Specification.....	33

Chapter 1

Introduction

1.1 General Description

The AX10420 is a PC/104 module which is primary intended to PC embedded application in industrial environment, containing 48-bit digital input and output. It 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 48 TTL/DTL/CMOS compatible digital I/O lines are arranged into two separated groups. Each group supports 8255 PPI (Programmable Peripheral Interface) chip mode 0 but with stronger driving capability and consists of three 8-bit ports; Port A, Port B, and Port C. These ports can be functionally programmed as either digital inputs or digital outputs. Of the three ports, only Port C is further divided into Port C-upper (4-bit) and Port C-lower (4-bit) which can be independently configured for input or output port.

There is a unique feature associated with AX10420: an interrupt on change of state. Interrupt occurs when Port C bit 3 or bit 7 of each group changes state. This feature frees up the PC to do other activities since there is no need to poll the digital input port for an event to occur.

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

Input and Output

- Input/Output Lines : 48
- Operation Mode : 8255 MODE 0
- Input/Output Mode : Pair
- Interrupt Options : Jumper-selectable to level 9(2), 5, 10, 11, 12, or 15
- Improved Noise Margins : Hysteresis
 $V_{T+} - V_{T-} = 0.4 \text{ typ.}$
- Input/Output Level: TTL/DTL compatible
- Added Pull-up Resistor : CMOS/dry contact compatible
- Electrical Characteristics
 - ⊙ V_{IH} : 2V min.
 - ⊙ V_{IL} : 0.8V max.
 - ⊙ I_{IH} : 20uA max. at $V_I = 2.7V$
 - ⊙ I_{IL} : -0.2mA max. at $V_{IL} = 0.4V$
 - ⊙ V_{OH} : 2.4V min at $I_{OH} = -3mA$
 - ⊙ V_{OL} : 0.4V max at $I_{OL} = 12mA$
 - ⊙ I_{OH} : -15mA max.
 - ⊙ I_{OL} : 24mA max.

Interface Characteristic

- I/O Connector : 50-pin male mating connector
- I/O Cable Type :
 - ⊙ Ribbon Twisted Pair Cable : $Z_0 = 50\Omega$ to 100Ω typ.
 - ⊙ Ribbon Stripline Cable : $Z_0 = 30\Omega$ to 80Ω typ.
- Compatible Bus : PC/104 bus
- Interface Type : I/O mapped with 10-bit addressing (A9 – A0)
- Number of Locations Occupied : 8 consecutive addresses
- Data Path : 8 bits

Power Requirements

- +5V : 0.4A typ.

Physical/Environmental

- Dimensions : 95mm X 90mm
- Weight : 200g
- Operating Temperature Range: 0°C to 60°C
- Storage Temperature Range: -25°C to 85°C
- Relative Humidity : To 90%, non-condensing

1.4 Accessories Guide

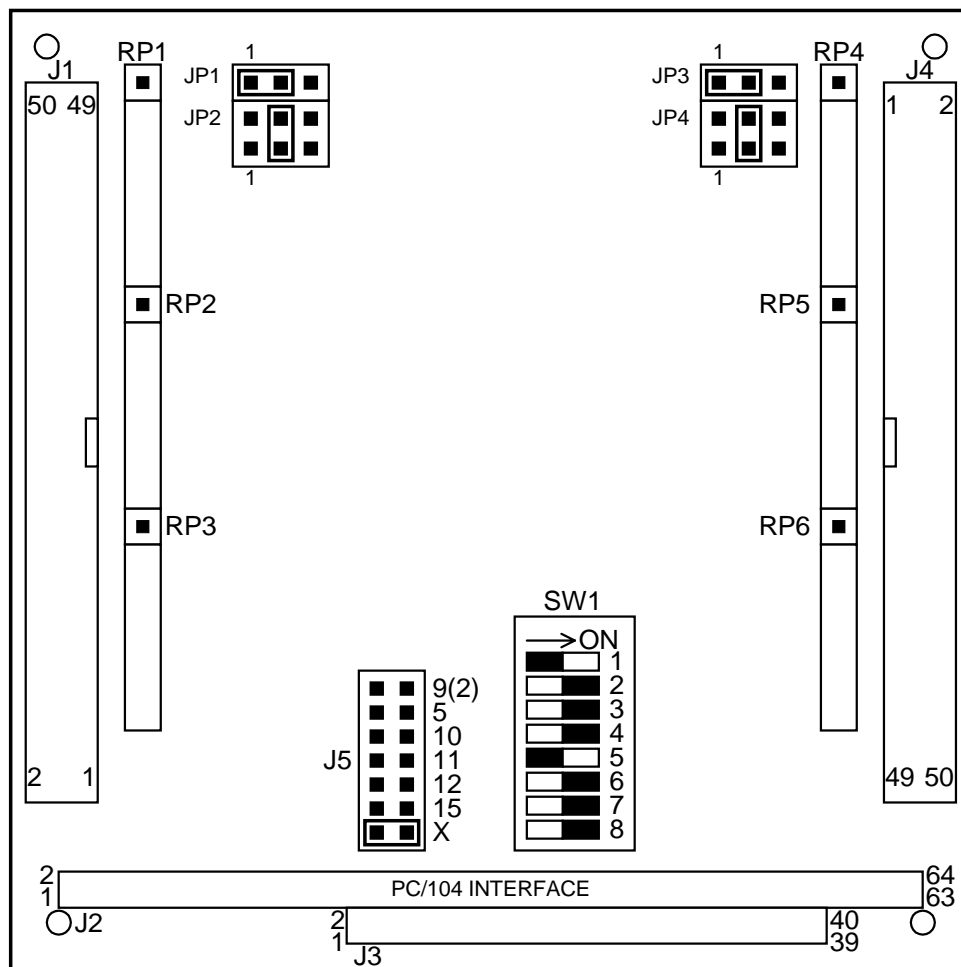
- **AX751**
Screw terminal board for all digital I/O connections. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.
- **AX754**
24-channel opto-isolated D/I panel for signal connection and conditioning with the AX10420. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.
- **AX755**
8-channel electromechanical single-pole, double-throw(SPDT) and 16-channel opto-isolated digital I/P panel which is compatible with the AX10420. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.
- **AX756**
24-channel electromechanical single-pole, double-throw(SPDT) which can be driven by the AX10420. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.

Chapter 2

Module Configuration and Installation

2.1 Component Locator Diagram

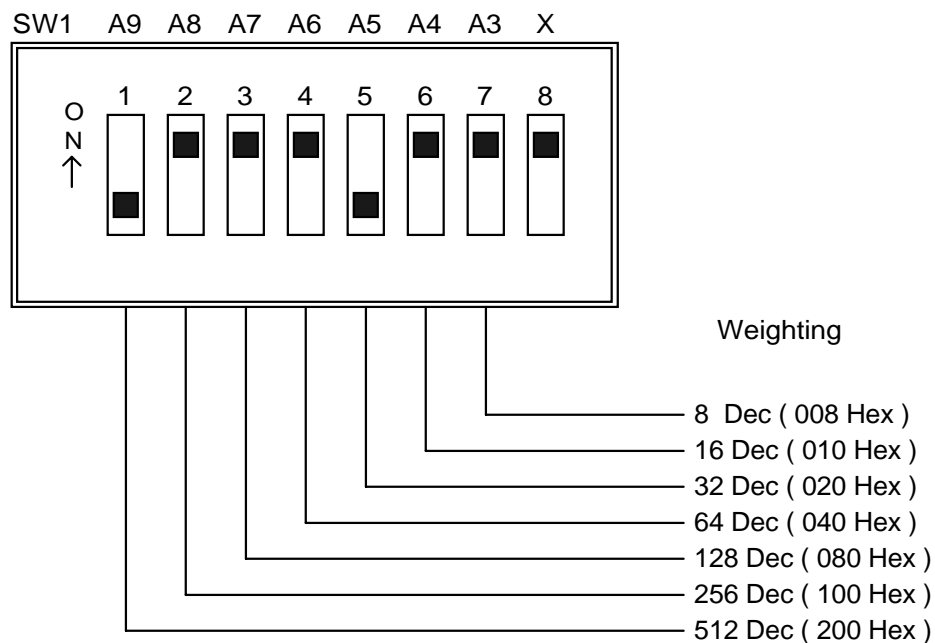
The following figure shows the location of AX10420's components. All switch and jumper settings in this figure are the factory default setting.



2.2 Base Address Switch

The AX10420 module occupies 8 consecutive locations in I/O address space. The first address or base address is selected via a 8-position DIP switch labeled SW1. If more than one module are to be installed to the embedded system, each module must be given its own distinct I/O address or base address. No more than one module may use the same base address. It would be better if you check with **Appendix A** for I/O port distribution to avoid conflicting with other installed devices. In factory, the AX10420 base address is set for 220 Hex or 544 Dec.

To set to appropriate base address, switch the individual switches into the ON or OFF position. The following figure shows DIP switch default setting, 220 Hex, where switches 1 and 5 are moved to the OFF position while leaving all other switches in the ON position. A table for DIP switch setting is given in the following page.



NOTE X : Not used.

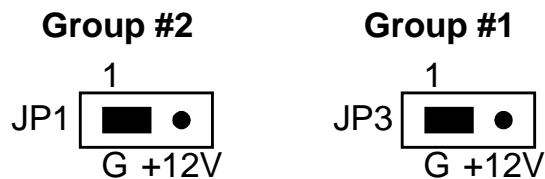
Each switch represents one address weight. The desired base address is determined by adding the weight of the switches flipped at OFF position. The base address calculation is as follows:

$$\begin{aligned}\text{Base Address} &= 512 + 32 = 544 \text{ Dec} \\ &= 220 \text{ Hex}\end{aligned}$$

I/O Port Range	DIP Switch Position							
Hexadecimal	1	2	3	4	5	6	7	8
	A9	A8	A7	A6	A5	A4	A3	X
200 – 207	1	0	0	0	0	0	0	X
208 – 20F	1	0	0	0	0	0	1	X
210 – 217	1	0	0	0	0	1	0	X
218 – 21F	1	0	0	0	0	1	1	X
220 – 227 (*)	1	0	0	0	1	0	0	X
.
3F0 – 3F7	1	1	1	1	1	1	0	X
3F8 – 3FF	1	1	1	1	1	1	1	X

NOTE 0 = ON, 1 = OFF,
(*) : Factory default setting

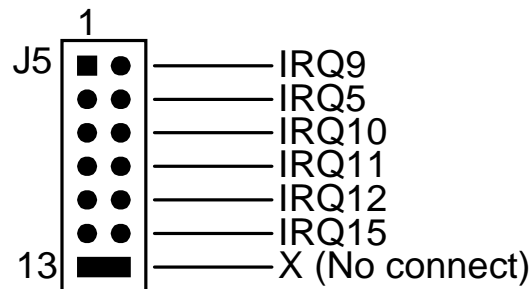
2.3 +12V or Ground Selection Jumper



Jumper cap should be placed at pins 1 and 2 of JP1 (JP3) to connect pins 2 and 4 of J1(J4) connector to Ground. With this configuration, the AX10420 is compatible with AX1416 and AX1424 Opto-22 interface panels. When jumper cap is placed at pins 2 and 3 of JP1, pins 2 and 4 of J1(J4) connector are connected to +12V. With this configuration, the AX10420 is compatible with AX754, AX755 and AX756 accessory board. The above figure shows the factory default setting for JP1 and JP3.

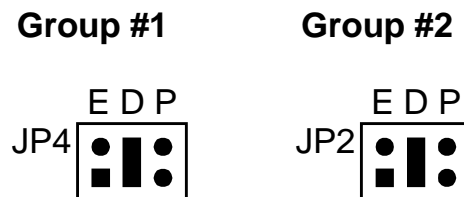
2.4 IRQ Level Jumper

Jumper labeled J5 is used for selecting IRQ level (9(2), 5, 10, 11, 12 15). Below figure gives the jumper configuration and default setting of J5. Place jumper cap at “X” position, if no interrupt is required.



2.5 Interface Setting Jumper

AX10420 provides hardware interrupt function for applications. Port C bit 3 (PC3) and bit 7 (PC7) are in charge of the task. INTERRUPT ENABLE (IENx) and PROGRAMMABLE INTERRUPT(INPx) are selectable via two jumpers. The interrupt signal can be lead to any of the six interrupt request lines (IRQ level 9(2), 5, 10, 11, 12, 15) by J5 jumper. Below figure gives the default setting for JP4 and JP2 jumpers.



NOTE

D : Disable Interrupt,

E : Enable Interrupt,



P : Programmable Interrupt

■ Disable Interrupt Jumper (“D” position)

When this jumper is set, any interrupt input on this group will be disabled.

■ Enable Interrupt Jumper ("E" position)

When this jumper is set, any change from 0 to 1 on this group's PC3 will generate an interrupt. The status is illustrated as follows:





PC3	Interrupt	
	YES	(*)
	NO	

NOTE

(*) After interrupt immediately pull down PC3, thus enable other interrupt to happen.

■ Programmable Interrupt Jumper ("P" position)

When this jumper is set, a programmable interrupt function can be raised via PC3 and PC7. The status is illustrated as follows:

PC3	PC7	Interrupt	
	0	YES	(*)
	0	NO	
X	1	NO	
0	X	NO	
1		YES	(**)
1		NO	

NOTE

(*) After interrupt immediately pull down PC3, thus enable other interrupt to happen.

(**) After interrupt immediately pull high PC7, thus enable other interrupt to happen.

2.6 Connector Pin Assignments

The AX10420's 48 DI/O lines are divided into two groups; Group #1 and Group #2. The 24 DI/O lines of Group #1 are built in **J4** 50-pin connector while the 24 DI/O lines of Group #2 are built in **J1** 50-pin connector. Both connector pin assignments are the same and shown in below figure. Through these connectors, the AX10420 module can be directly connected to AXIOMTEK's AX751, AX754, AX755 and AX756 accessory boards or standard Opto-22 interface panel.

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

Pin Name	Description
+5V	+5V PC power supply
PA0 – PA7	Port A – eight digital I/O lines
PB0 – PB7	Port B – eight digital I/O lines
PC0 – PC3	Port C-lower – four digital I/O lines. The PC3 has interrupt handling capability. Refer to <i>Interrupt Setting Jumper</i> section.
PC4 – PC7	Port C-upper – four digital I/O lines. The PC7 has interrupt handling capability. Refer to <i>Interrupt Setting Jumper</i> section.
OPT	These pins can be connected to +12V PC power or Ground by jumpering JP1 and JP3. Refer to <i>+12V or Ground Selection Jumper</i> section.

WARNING *As pin 2 and pin 4 can be connected to +12V or Ground (refer to **+12V or Ground Selection Jumper** section), thus when the AX10420 is connected to other board through this 50-pin connector, user must pay attention to the connector pin assignment (especially pin 2 and pin 4) of the corresponding board.*

2.7 Resistor Pack

As mentioned before the 8-bit port of each group can be configured for input or output port (refer to **Chapter 3**). Initially the digital I/O lines are left floating. When any of these ports is set to input port, user is suggested to pull high it's input lines by installing RP(s). Onboard there are six reserved spaces, marked as RP1 – RP6 (refer to below table, the RP is approximately 4.7K). If a port is configured as output as output lines, just leave the corresponding RP unoccupied.

DI/O Lines	RP
Group #1 Port A	RP6
Group #1 Port B	RP5
Group #1 Port C	RP4
Group #2 Port A	RP1
Group #2 Port B	RP2
Group #2 Port C	RP3

NOTE *In some situations, i.e. environment ground is not stable, the digital output resets frequently. User is suggested to isolated system circuitry from external signal. Let the external signal to go through AX754 (24 Channel Opto-isolated D/I Panel) or AX755 (8 Channel Relay Output and 16 Channel Opto-isolated D/I panel) before reaching the system circuitry.*

2.8 Hardware Description

PC/104 module can be of two bus types, 8 bit and 16 bit. These correspond to the PC and PC/AT buses, respectively. The detailed mechanical dimensions of these two PC/104 bus types are provided in **Appendix D**.

Basically the AX10420 belongs to 16 bit bus option which design only to by pass PC/AT bus signal in order to compatible to PC/AT type PC/104 module. The AX10420 uses only IRQ lines on J3 40-pin connector. If this module is going to plug onto PC type PC/104 bus, do not use IRQ line above 10.

Besides bus option, there are stackthrough and non-stackthrough difference. The stackthrough version provides a self-stacking PC bus. It can be placed any where in a multi-module stack. The non-stackthrough version offers minimum thickness, by omitting bus stackthrough pins. It must be positioned at one end of a stack.

For convenience, the AX10420 is equipped with stackthrough version only.

NOTE *For safety, you are suggested to cut bus stackthrough pins of the last module on condition; that you are sure you won't add/plug any module to the module stack in the future.*

2.9 Module Installation

The AX10420 board is shipped with protective electrostatic cover. When unpacking, touching the board electrostatically shielded packaging with the metal frame of your computer to discharge the accumulated static electricity prior to touching the board.

Following description summarizes the procedures for installing the AX10420:

WARNING *Turn off the PC and all accessories connected to the PC whenever installing or removing any peripheral board including the AX10420 module.*

Installation Procedures:

1. Turn off the system power.
2. Unplug all power cords.
3. Remove the case cover if necessary.
4. Remove the top module if it is a non-stackthrough module.
5. Put the AX10420 module in line with the present module as described in ***Appendix D***.
6. Install four spacers and fasten them if necessary.
7. Crush between the modules until inside distance is SPACER's height (0.6"). Restore all the screws.
8. Repeat step 6 until all modules are set into position.
9. Connect cable to AX10420 (J1 or J4) if necessary.
10. Restore the case cover and connect all the necessary cables.
11. Turn on the system power.

Chapter 3

Register Structure and Format

The AX10420 occupies 8 consecutive I/O addresses of PC I/O address space. During installation, the first address or base address is determined by setting onboard DIP switch (SW1).

This chapter describes each AX10420 register in terms of function, address, bit structure and bit function. Each register is easy to read and write to by using direct I/O instructions of whatever application languages.

3.1 AX10420 I/O Address Map

The 48 digital I/O lines of AX10420 are arranged into separated groups. Each group supports 8255 PPI chip mode 0.

The AX10420 is programmable through configuration registers. By writing to control registers, the type of each group may be specified. If a group is configured as a write port, the data driver will drive the data value to the corresponding port. If a group is configured as a read port, the data value on corresponding port will be sent to the digital I/O lines. Only Port C of each group is divided into two 4-bit nibbles; Port C-upper and Port C-lower, of which the I/O direction can be determined by programming to the control register.

The following table lists and describes the registers and their locations (R = Read, W = Write, Base = Base address).

Location	Function	Type
Base Address +0	Group #1 Port A	R/W
Base Address +1	Group #1 Port B	R/W
Base Address +2	Group #1 Port C	R/W
Base Address +3	Group #1 Control Register	W
Base Address +4	Group #2 Port A	R/W
Base Address +5	Group #2 Port B	R/W
Base Address +6	Group #2 Port C	R/W
Base Address +7	Group #2 Control Register	W

3.2 AX10420 Register Description

Group #1 Data and Control Registers (Base +0 to +3)

■ Port A Data Register (Base +0, R/W)

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

■ Port B Data Register (Base +1, R/W)

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

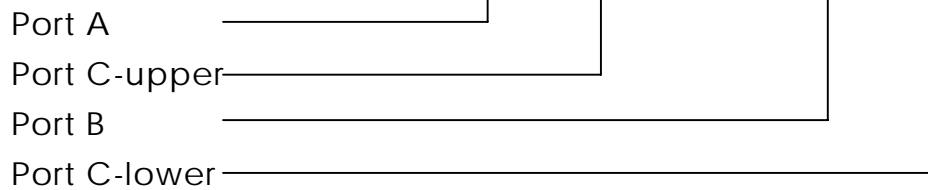
■ Port C Data Register (Base +2, R/W)

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

PC0 – PC3: Port C-lower, PC4 – PC7: Port C-upper

■ Control Register (Base +3, W)

base	7	6	5	4	3	2	1	0
+3	1	0	0	D4	D3	0	D1	D0



NOTE

1) PA0 – PA7, PB0 – PB7 and PC0 – PC7 bits are associated to pins at J4 connector.

2) For D0, D1, D3, D4 : 1 → Input, 0 → Output

Group #2 Data and Control Register (Base +4 to +7)

■ Port A Data Register (Base +4, R/W)

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

■ Port B Data Register (Base +5, R/W)

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

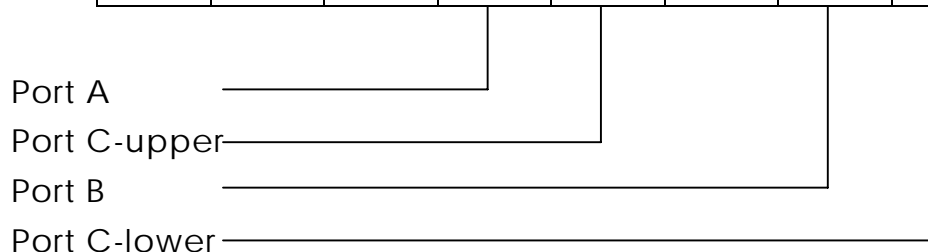
■ Port C Data Register (Base +6, R/W)

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

PC0 – PC3: Port C-lower, PC4 – PC7: Port C-upper

■ Control Register (Base +7, W)

base	7	6	5	4	3	2	1	0
+7	1	0	0	D4	D3	0	D1	D0



NOTE

1) PA0 – PA7, PB0 – PB7 and PC0 – PC7 bits are associated to pins at J1 connector.

2) For D0, D1, D3, D4 : 1 → Input, 0 → Output

Chapter 4

Programming

4.1 Digital Input and Output

AX10420 provides 48-bit digital I/O lines arranged into two groups. Each group contains three 8-bit ports; Port A, B and C. Port C is divided into two 4-bit nibbles; Port C-upper, Port C-lower. The I/O direction of the ports (Port A, B, C-upper and C-lower) can be determined by programming to the control register.

Programming Examples

The following BASIC program configures Group #1 Port A and B input port (install corresponding RP's), Port C as output port. An increasing pattern is sent to Port C. It is expected that user will connect both Port A and Port B to Port C before running this program.

```
10  CLS
20  PORT% = &H220          ' REM Base address
30  OUT PORT%+3, &H92      ' REM Port A, B: input, C: output
40  FOR J =0 to 255        ' REM Decimal value from 00 to FF
50  OUT PORT %+2, J        ' REM Output data to Port C
60  B = INP (PORT%)        ' REM Read data on Port A
70  C = INP (PORT%+1)      ' REM Read data on Port B
80  PRINT B, C, J          ' REM Check data versus Port A and B
90  NEXT J
100 END
```

The following program configures Group #1 Port A, B and C as output ports. Data value of 00 to FF Hex are sent to all ports and read back from output latch to ensure that the transfer is successful.

```

10  PORT% = &H220
20  OUT PORT% +3, &H80      ' REM Base address
30  FOR J = PORT% To PORT% +2 ' REM Port A, B, C are all output
40  FOR X=0 to 255          ' REM Decimal value for Port A to C
50  OUT J, X                ' REM Decimal value for 00 to FF
                               Hex
60  B = INP (J)             ' REM Output value X to port J
70  PRINT X, B, J           ' REM Read back from latch
80  NEXT X                  ' REM Print input, output value, port
90  NEXT J
100 END

```

4.2 Interrupt

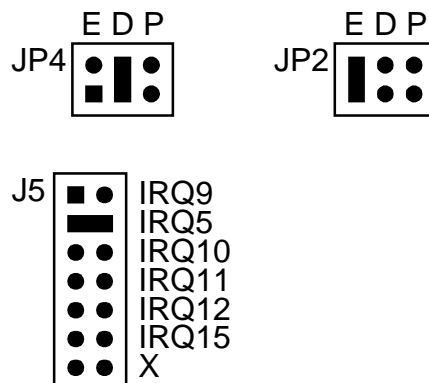
The AX10420's built-in interrupt control circuitry allows either Port C PC3 bit or PC7 bit of each group to cause an interrupt request. The group where interrupt comes from can be discovered by polling data from each group's Port C bit 3 and bit 7.

Interrupt can be caused by external input to Port C or by direct output to Port C. This feature can be used to detect external critical signal or to generate an interrupt from program.

The priority of interrupt is not defined in the interrupt control circuitry, user must define the priority via software control. Note that the inputs are not latched until the input(read) is executed.

Programming Examples

Here we provide a demo program written in Turbo C. Before executing this program, Group #2 PA3 pin must be wired to PC3 pin. This program give a brief description of the interrupt handling capability of the AX10420. It sets Group #2 Port A for output port and Port C for input port (install corresponding RP(s) if necessary) and sends 0 followed by 1 to PA3. The IRQ level selected is IRQ5. If interrupt occurs then shows 'Interrupt' on screen. The jumper configurations are as follows:



/* DEMO PROGRAM IN TURBO C */

```
#include    <dos.h>
#include    <stdio.h>
#include    <conio.h>

#define intp 0x0d    /* IRQ5 interrupt number */
#define mimr 0xdf    /* Enable Peripheral Interrupt Controller for IRQ5 */
#define base 0x220    /* Default base address */
#define ctr_r 0x89

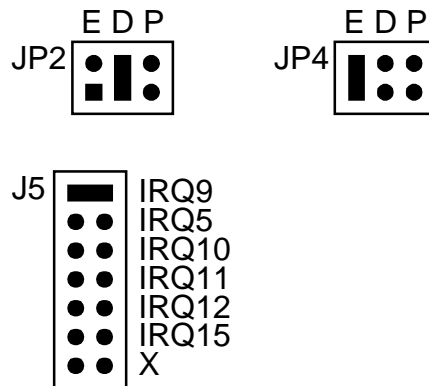
void interrupt (*oldisr)();
void interrupt far newisr();
void restore_isr(void);
int Intolsr;

main()
{
    clrscr();
    oldisr = getvect(intp);
    setvect(intp, newisr);
    outp(0x21, inp(0x21)& mimr); /* Enable IRQ5 */
    outp(base+7, ctr_r);        /* Set Port A as output, Port C as input */
    do {
        Intolsr=0;
        outp(base+4,0x0);        /* Generate a pulse from Port A */
        outp(bae+4,0x08);
        delay(50);
        outp(base+4,0x0);
        if (Intolsr==1) {printf("interrupt\n");Intolsr=0;}
    } while (kbhit()==0);
    restore_isr();                /* Free ISR */
    outp(0x21,0xb8);            /* Disable IRQ5 */
}

void interrupt far newisr()        /* ISR routine */
{
    Intolsr=1;
    outp(0x20,0x20);
}

void restore_isr(void)
{
    setvect(intp,oldsr);
}
```

The following is another demo program written in Turbo Pascal. It is similar to the proceeding program in Turbo C except here we select Group #1 and IRQ9. The jumpers settings are thus differ as follows:



{ DEMO PROGRAM IN PASCAL }

```
{ $M 1024,0,0 }
PROGRAM AX10420_IRQ9_INT;
USES DOS,CRT;
CONST
  INTP=$0A;           {IRQ9}
  MIMR=$FB;           {Enable peripheral interrupt controller for IRQ9}
  BASE=$220;          {Default base setting}
  CTR_R=$89;
VAR
  INTOISR:INTEGER;
  OLDISR:POINTER;
  RKB:CHAR;

PROCEDURE NEW_ISR; INTERRUPT;
BEGIN
  INTOISR:=1;
  PORT[$20]:=$20;
END

BEGIN
  GETINIVEC(INTP,OLDISR);      {Get old interrupt vector and save it}
  SETINIVEC(INTP,@NEW_ISR);    {Install the new handler}
  PORT[$21]:=MIMR AND PORT[$21]; {Enable IRQ9}
  PORT[BASE+3]:=CTR_R;         {Set Port A as output, Port C as input}
  WRITELN('ENTER key to continue, ENTER ESC to end');
  INTOISR:=0;
```

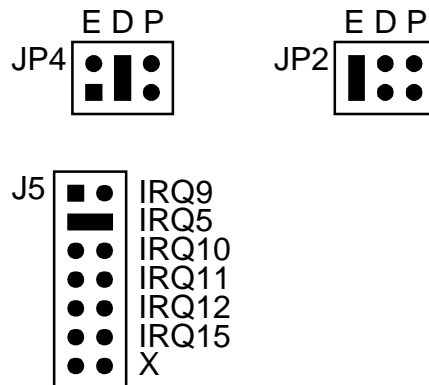


```

READ(RKB);
REPEAT
  BEGIN
    PORT[BASE+0]:=$00;           {Generate a pulse to trigger interrupt}
    PORT[BASE+0]:=$08;
    PORT[BASE+0]:=$00;
    IF INTOISR=1 THEN
      BEGIN
        WRITELN('Interrupt');
        INTOISR:=0;
      END;
    READ(RKB);
  END;
UNTIL(readkey = #27);
SETINIVEC(INTP,OLDISR);         {Free interrupt $0b}
PORT[$21]:=PORT[$21]+8;         {Disable IRQ9}
END.

```

Here is another demo program in Microsoft C which is similar to the preceeding program in Turbo C. This program configures Group #2 as output port and write 0 followed by 1 to its PC3 bit to generate interrupt. The IRQ level selected is IRQ5. Configure the jumpers as follows:



/* DEMO PROGRAM IN MICROSOFT C */

```

#include <dos.h>
#include <conio.h>
#include <stdio.h>

#define BASE 0x220           /* Default base address setting */

```

```

    int pc3_high=0;
    static short int_num;
    static void *old_int;
    void interrupt far isr();

void initiate(void)                /* Initiate interrupt */
{
    _disable();
    int_num = 0x0d;                /* IRQ5 interrupt number */
    old_int = _dos_getvect(int_num); /* Get old interrupt vector and save it */
    _dos_setvect(int_num,isr);      /* Install the new handler */
    outp(0x21,inp(0x21)&0xdf);      /* Unmask IRQ5 */
    _enable();
}
void interrupt far isr(void)        /* Interrupt service routine */
{
    _enable();
    pc3_high=1;                    /* If interrupt occur set pc3_high to 1 */
    outp(0x20,0x20);              /* Interrupt completed */
}
void close_int(void)
{
    _disable();
    _dos_setvect(0x0d,old_int);    /* Restore original interrupt vector */
    outp(0x21,inp(0x21)|0x20);    /* Mask IRQ5 */
    _enable();
}
main()
{
    initiate();
    outp(BASE+7,0x80);             /* Assign Group #2 as output */
    outp(BASE+6,0);               /* Set Group #2:PC3 low */
    outp(BASE+6,0x08);            /* Set Group #2:PC3 high */
    outp(BASE+6,0);
    while (!kbhit())
    {
        if (pc3_high==1)
        {
            printf("Interrupt");
            pc3_high=0;
        }
    }
    close_int();
}

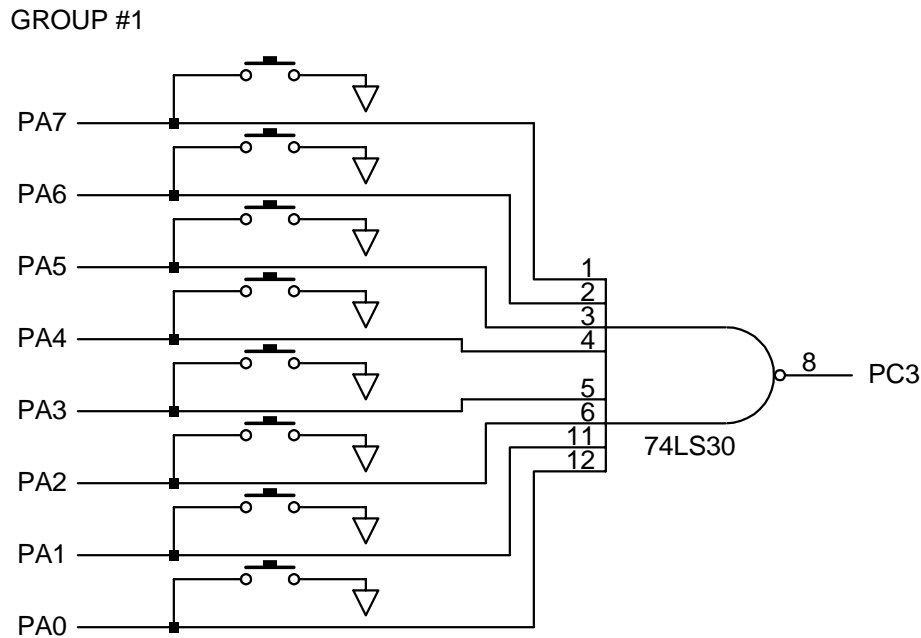
```

Chapter 5

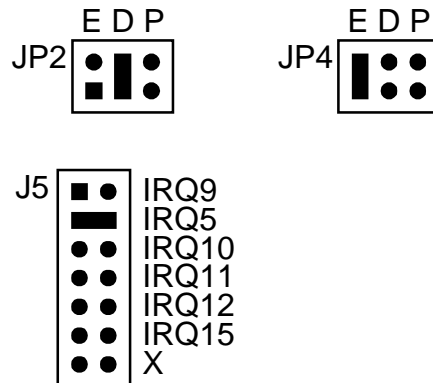
Application

In this chapter, two simple application examples are provided.

5.1 Event Trigger



Configured Group #1 as input port and set jumpers as described in figure below. Install the corresponding RP(s). If any of the push button is pressed, an interrupt is acknowledged via PC3. Reading back the data value on Port A will tell us which one of the PA0 through PA7 is low (pressed down). A demo program in Turbo C is provided in the following page.



/* DEMO PROGRAM IN TURBO C FOR EVENT TRIGGER */

```
#include <dos.h>
#include <conio.h>
#include <stdio.h>

#define BASE 0x220                                /* Default base address setting */

static int pc3_high=0;
static short int_num;
static inp PA;

void interrupt far isr();
static void interrupt (*old_int)();

void initiate(void)
{
    disable();
    int_num = 0x0d;                                /* IRQ5 interrupt number */
    old_int = getvect(int_num);                    /* Get old interrupt vector and save it */
    setvect(int_num,isr);                          /* Install the new handle */
    outp(0x21,inportb(0x21)&0xdf);                /* Unmask IRQ5 */
    enable();
}

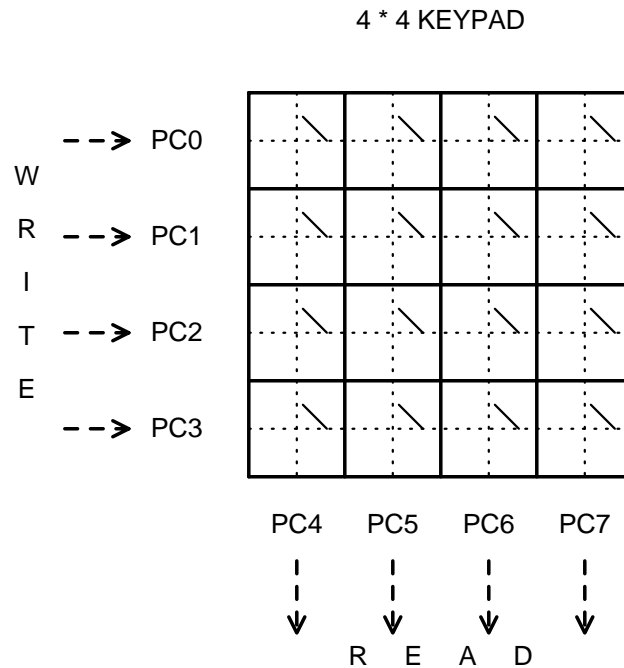
void interrupt far isr(void)
{
    enable();
    if (pc3_high==0)
    {
        PA=inp(BASE);                            /* Read status */
        pc3_high=1;                              /* If interrupt occur set pc3_high to 1 */
    }
    outp(0x20,0x20);                              /* Interrupt completed */
}

void close_int(void)
{
    disable();
    setvect(0x0d,old_int);                        /* Restore original interrupt vector */
    outp(0x21,inp(0x21)|0x20);                    /* Mask IRQ5 */
}
```

```
enable();
}

main()
{
    clrscr();
    initiate();
    outp(BASE+3,0x9B);          /* Assign Group#1 as input */
    while (!kbhit())
    {
        if (pc3_high==1)
        {
            if ((PA&0x01)==0) printf("PA0 pressed! ");
            if ((PA&0x02)==0) printf("PA1 pressed! ");
            if ((PA&0x04)==0) printf("PA2 pressed! ");
            if ((PA&0x08)==0) printf("PA3 pressed! ");
            if ((PA&0x10)==0) printf("PA4 pressed! ");
            if ((PA&0x20)==0) printf("PA5 pressed! ");
            if ((PA&0x40)==0) printf("PA6 pressed! ");
            if ((PA&0x80)==0) printf("PA7 pressed! ");
            delay(250);
            pc3_high=0;
        }
    }
    close_int();
}
```

5.2 Polling 4*4 Keypad



Install the corresponding RP(s) to pull high the digital input lines. Configure Group #1 Port C-upper as read port and Port C-lower as write port. In a loop, send below patterns to Group #1 Port C:

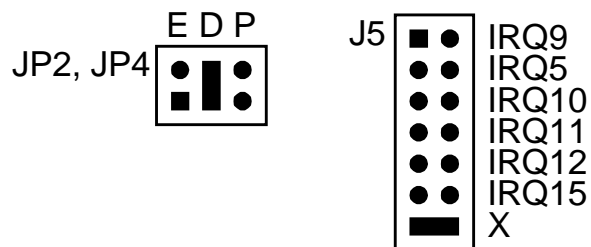
XXXX 1110

XXXX 1101

XXXX 1011

XXXX 0111

Each time after sending a pattern, check whether keypad is pressed by reading back the data in Port C-upper. Any 0 value in this 4-bit nibble means keypad is pressed. Then find out the pressed position. A demo program in Turbo C is provided in the following page.



/* DEMO PROGRAM IN TURBO C FOR POLLING KEYPAD */

```
#include <conio.h>
#include <dos.h>
#include <stdio.h>

#define BASE 0x220                                /* Default base setting */

void re_key(void)                                  /* Find which key pressed */
{
    int gl_pc;

    gl_pc=inp(BASE+2);
    if (gl_pc & 0x10)== 0)        printf("PC4)      ");
    else if ( (gl_pc & 0x20)== 0)    printf("PC5)      ");
    else if ( (gl_pc & 0x40)== 0)    printf("PC6)      ");
    else if ( (gl_pc & 0x80)== 0)    printf("PC7)      ");
}

vod main()
{
    clrsc();
    outp(BASE+3,0x88);
    whiel (!kbhit())
    {
        oupt(BASE+2,0xfe);          /* Send pattern 1111 1110 */
        if ((inp(BASE+2)>> 4) != 0x0f) /* Check if key pressed */
        {
            printf("(PC0,");
            re_key();
            delay(250);
        }
        while ((inp(BASE+2)>> 4) != 0x0f; /* Wait until key released */

        outp(BASE+2,0xfd);          /* Send pattern 1111 1101 */
        if ((inp(BASE+2)>> 4) != 0x0f) /* Check if key pressed */
        {
            printf("(PC1,");
            re_key();
            delay(250);
        }
    }
```

```
while ((inp(BASE+2)>>4) != 0x0f);      /* Wait until key released */

outp(BASE+2,0xfb);                      /* Send pattern 1111 1011 */
if ((inp(BASE+2)>>4) != 0x0f)           /* Check if key pressed */
{
    printf("(PC2,");
    re_key();
    delay(250);
}
while ((inp(BASE+2)>>4) != 0x0f);      /* Wait until key released */

outp(BASE+2,0xf7);                      /* Send pattern 1111 0111 */
if ((inp(BASE+2)>>4) != 0x0f)           /* Check if key pressed */
{
    print("(pc3,");
    re_key();
    delay(250);
}
while ((inp(BASE+2)>>4) != 0x0f);      /* Wait until key released */
}
}
```

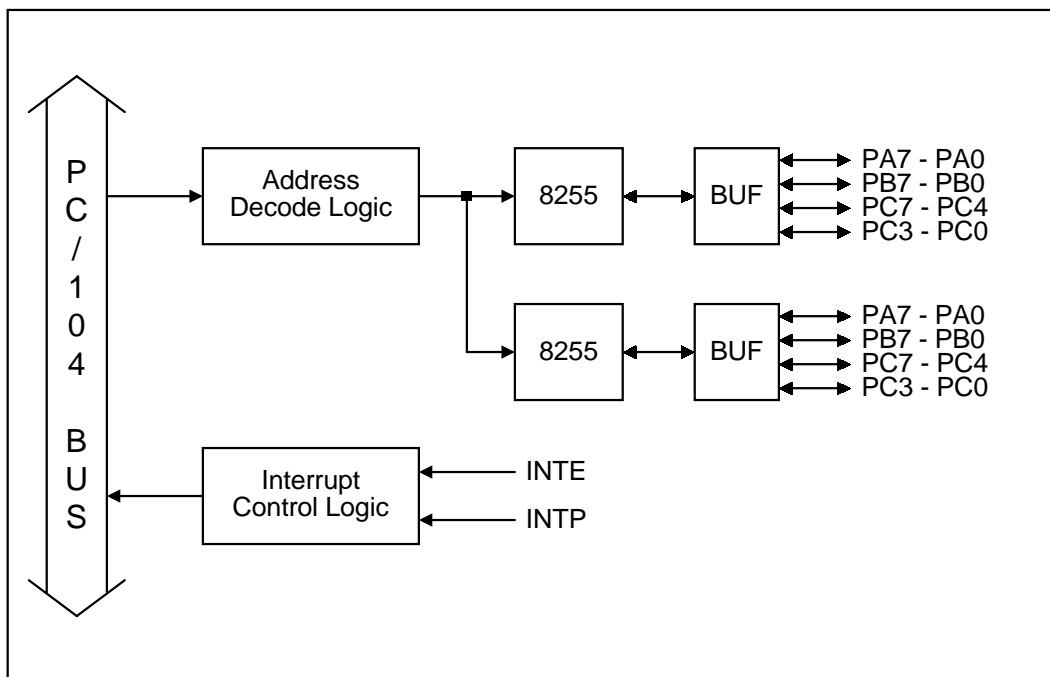

Appendix A

PC I/O Port Mapping

I/O Port Address Range	Function
000 – 1FF	PC reserved
200 – 20F	Game controller (Joystick)
278 – 27F	Second parallel printer port (LPT2)
2E1	GPIB controller
2F8 – 2FF	Second serial port (COM2)
320 – 32F	Fixed disk (XT)
378 – 37F	Primary parallel printer port (LPT1)
380 – 38F	SDLC communication port
3B0 – 3BF	Monochrome adapter/printer
3D0 – 3DF	EGA, reserved
3F0 – 3F7	Color/graphics adapter
3F8 – 3FF	Floppy disk controller
	Primary serial port (COM1)

Appendix B

Block Diagram



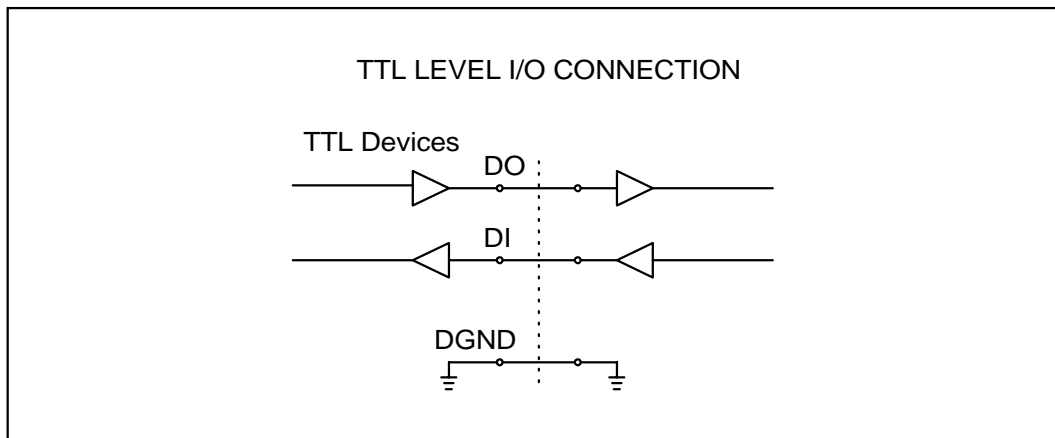
Appendix C

Technical Reference

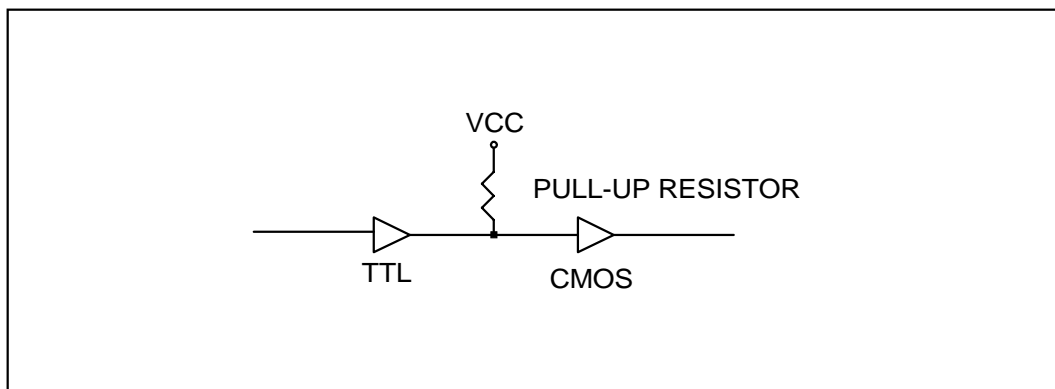
General Usage of Digital Input and Output

Digital signals are usually used for detecting logical status or controlling devices, a brief description is given below. TTL level signals are developed by most DAS systems.

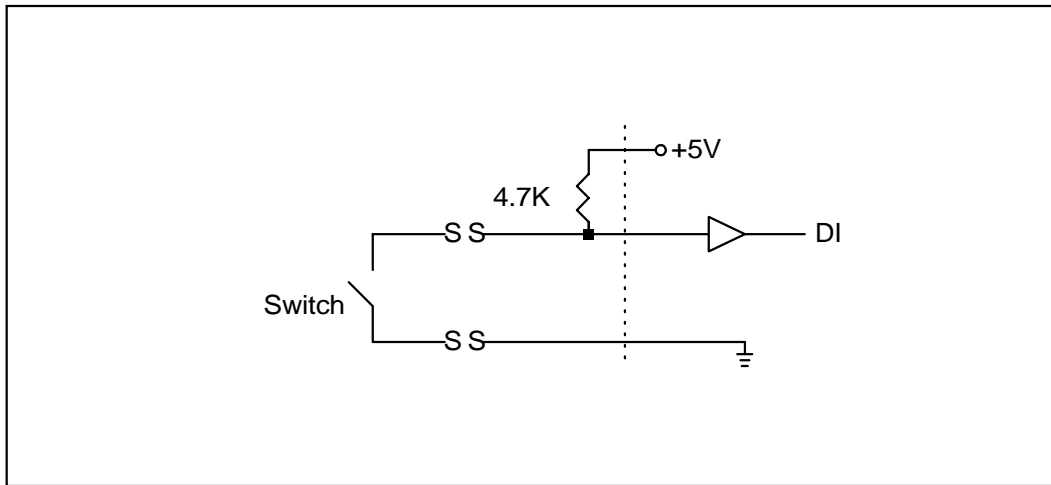
- TTL or LSTTL Level I/O Connections



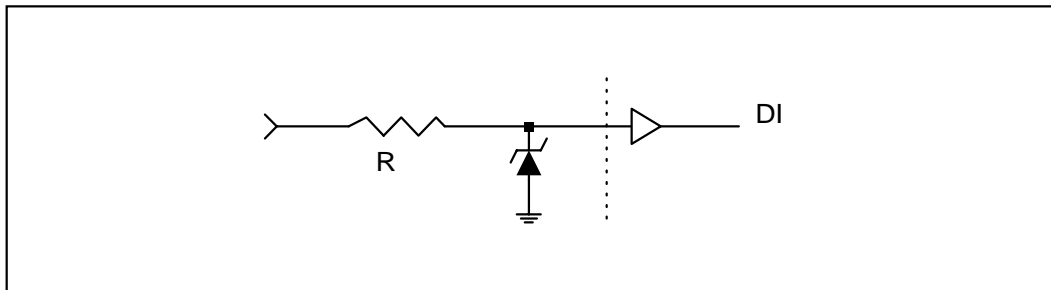
Connection with CMOS Device – Use a pull-up resistor if you wish to interface to CMOS devices. This will raise the logic high output level from its minimum TTL level of 2.4V to +5V suitable for CMOS interface.



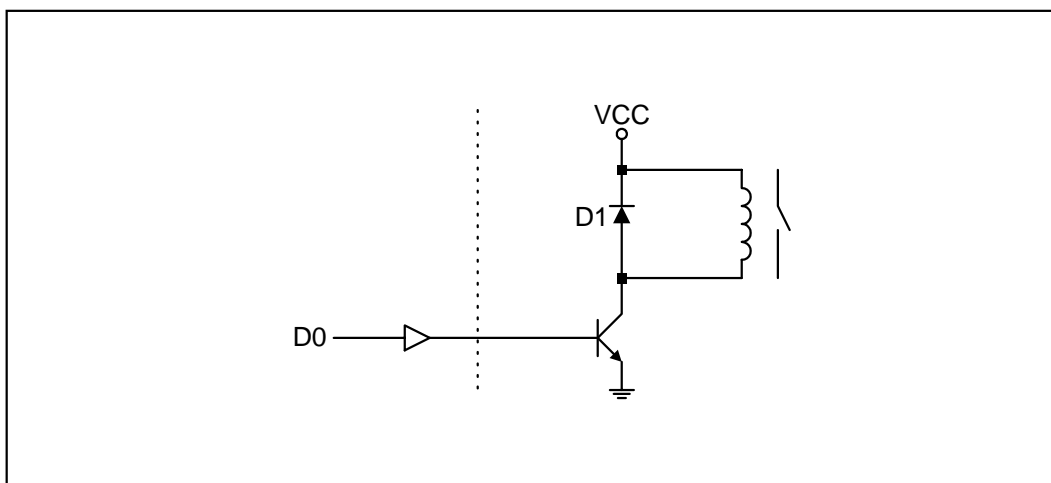
Digital Input for Open/Short Switch Detection – A pull-up resistor must be connected, especially at long distance wiring, to ensure logic high input level.



■ Digital Input for Large Signal

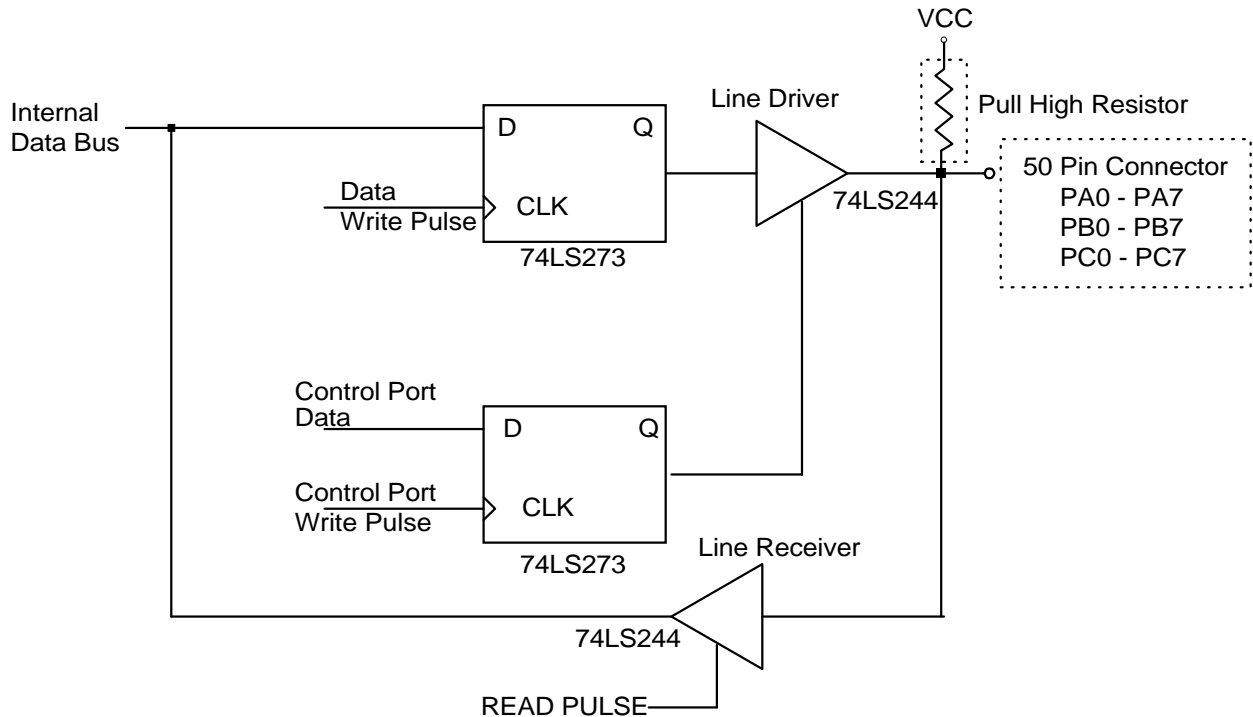


Digital Output for Relay Driving - The D1 diode is added to protect the IC driver against the inductive “kickback” from the relay coil.




AX10424 Port A, B and C Basic Definition

1. Equivalent ckt of Port A, B and C.



NOTE:

 Optional Resistor Pack
< Not Installed in Factory >

2. Any port is programmable to input or output.
3. Outputs are driven by 74LS244 and latched by 74LS273.
4. Inputs are received by 74LS244 but not latched.
5. Interrupt handling capability at PC3 and PC7.
6. All inputs and outputs are buffered by standard line drivers and line receivers.
7. The initial state and default setting of port A, B and C are tri-state.

Appendix D

PC/104 Mechanical Specification

PC/104 General Description

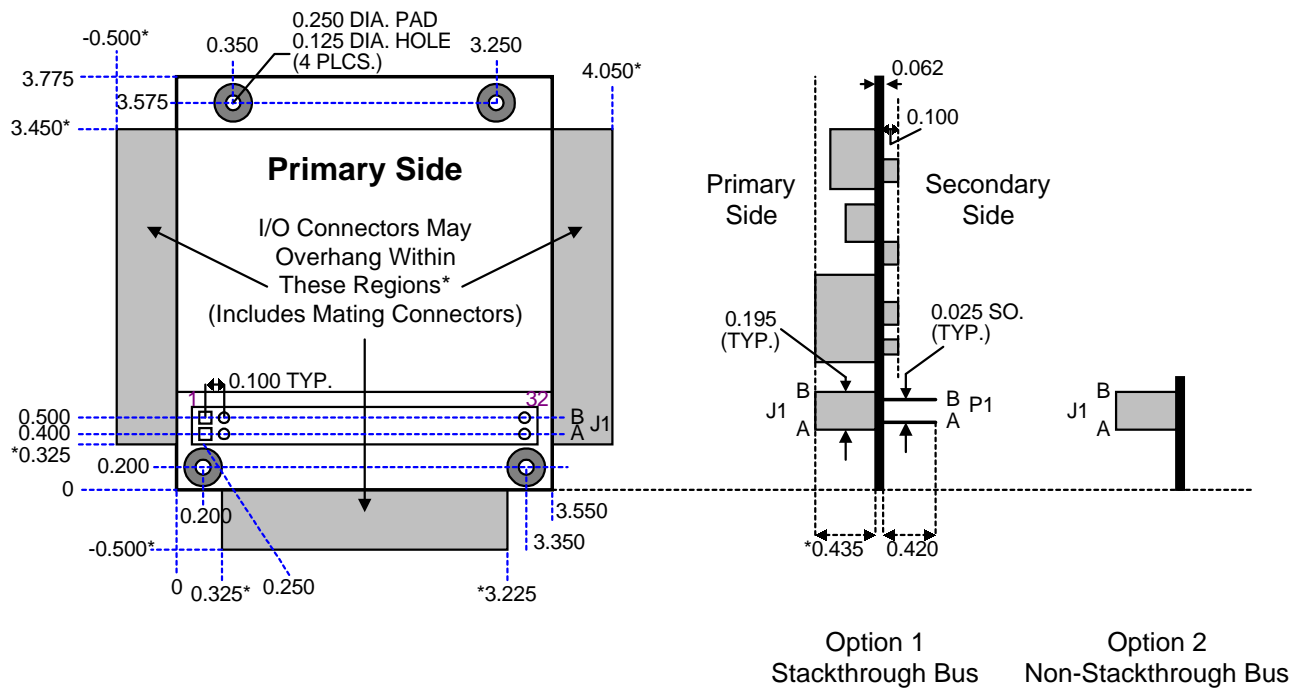
While the PC and PC/AT architectures have become extremely popular in both general purpose (desktop) and dedicated (non-desktop) applications, its use in embedded microcomputer applications has been limited due to the large size of standard PC and PC/AT motherboards and expansion cards.

This document supplies the mechanical and electrical specifications for a compact version of the PC/AT bus, optimized for the unique requirements of embedded systems applications. The specification is herein referred to as "PC/104", based on the 104 signal contacts on the two bus connectors (64 pins on J2 plus 40 pins on J3).

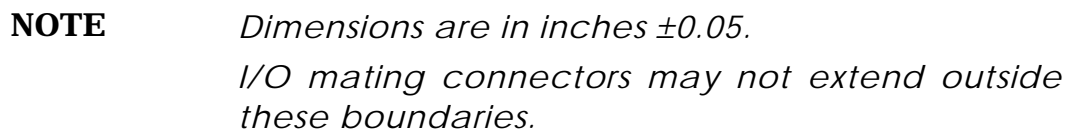
Module Dimensions

PC/104 modules can be of two bus types, 8-bit and 16-bit. These correspond to the PC and PC/AT buses, respectively.

■ PC/104 8-Bit Module Dimensions



35



■ Typical Module Stack

Figure 1 illustrates a typical module stack of 8-bit modules, and shows the use of the "stack-through" and "non-stackthrough" J2 bus connector options.

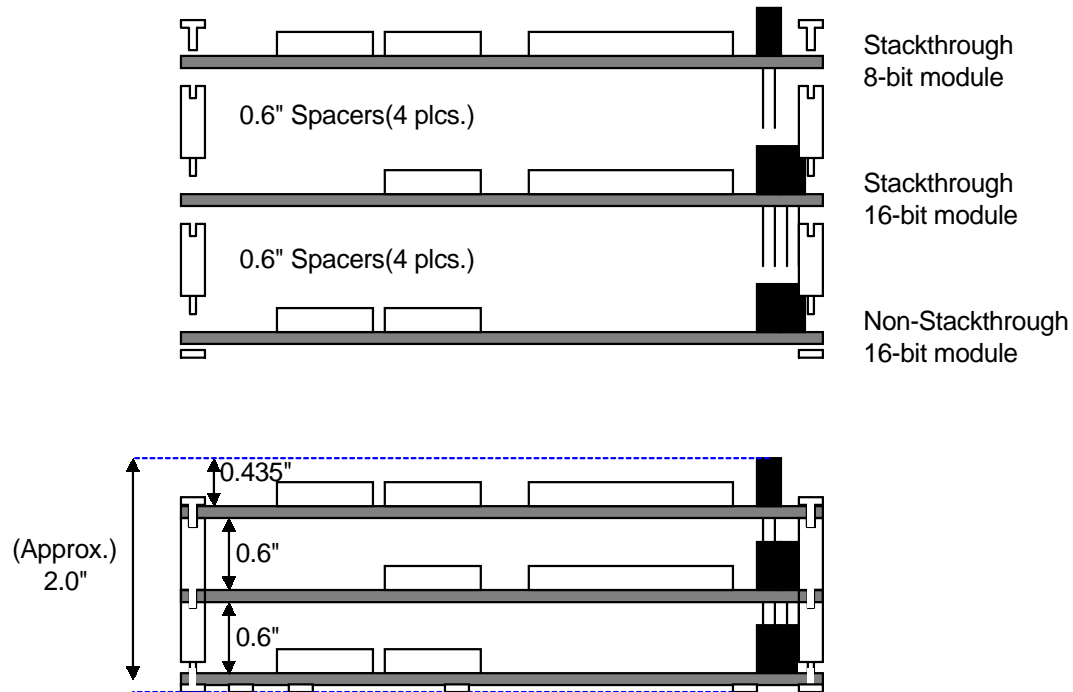


Figure 1 Typical Module Stack