EX – 9920

48 Bit DIO Module

CHECK LIST

Before getting started, check if your EX–9920 48 Bit DIO Module package includes the following items:

- **EX–9920 board**
- **Screw 3mm (x 4)**
- **Bronze stick 6mm (x 4)**
- **EX–9920 user's manual**

If anything is missing, please contact your dealer.

Table of Contents

CHAPTER 1 INTRODUCTION..... 1
- **General Description 1**
- **Applications 1**
- **Specifications..... 2**
- **Accessories Guide 4**

CHAPTER2 MODULE CONFIGURATION AND INSTALLATION 5
- **Component Locator Diagram 5**
- **Base Address Switch.....6**
- **+12V or Ground Selection Jumper 8**
- **IRQ Level Jumper 8**
- **Interrupt Setting Jumper.....9**
- **Connector Pin Assignments 11**
- **Resistor Pack.....13**
- **Hardware Description.....14**
- **Module Installation.....14**

CHAPTER 3 REGISTER STRUCTURE AND FORMAT 16
- **EX–9920 I/O Address Map 16**
- **EX-9920 Registers Description.....17**

CHAPTER 4 PROGRAMMING 19
- **Digital Input and Output 19**
- **Interrupt 20**

CHAPTER 5 APPLICATION 26
- **Event Trigger 26**
- **Polling 4*4 Keypad 29**

APPENDIX A PC I/O PORT MAPPING.....32

APPENDIX B BLOCK DIAGRAM 32

APPENDIX C TECHNICAL REFERENCE.....34
- **General Usage of Digital Input and Output.....34**
- **EX-9920 Port A, B and C Basic Definition.....36**

APPENDIX D PC/104 MECHANICAL SPECIFICATION.....37
- **PC/104 General Description 37**
- **Module Dimensions 37**

CHAPTER 1 INTRODUCTION

General Description

The EX-9920 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 used with TTL low-level input/output circuitry or with solid state relay module such as EX-9416 or EX-9424 and provides 2500V isolation for interfacing with high level AC and DC signals.

The 48 TTL/DTL/COMS 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 EX-9920: 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 on need to poll the digital input port for an event to occur.

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**

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 $V_T+ - V_T - =0.4$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 CHARACTERCTIC

I/O Connector	**50-pin male mating connector**
I/O Cable Type	
Ribbon Twisted Pair Cable	**Z_O = 50Ω to 100R typ**
Ribbon Stripline Cable	**Z_O = 30Ω to 80R typ**
Compatible Bus	**PC/104 bus**
Interface Type	**I/O mapped with 10-bit addressing (A9 – 10)**
Number of Locations occupied	**8 consecutive addresses**
Data Path	**8 bits**

Power Requirements

+5V	**0.4A typ**

Physical/Environmental

Dimensions	**95mm * 90mm**
Weight	**200g**
Operating Temperature Rang	**0 to 60℃**
Storage Temperature Rang	**–25 to85℃**
Relative Humidity	**To 90%, non-condensing**

Accessories Guide

EX-9951

Screw terminal board for all digital I/O connections. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.

EX-9954

24-channel opto-isolated D/I panel for signal connection and conditioning with the EX-9920. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.

EX-9955

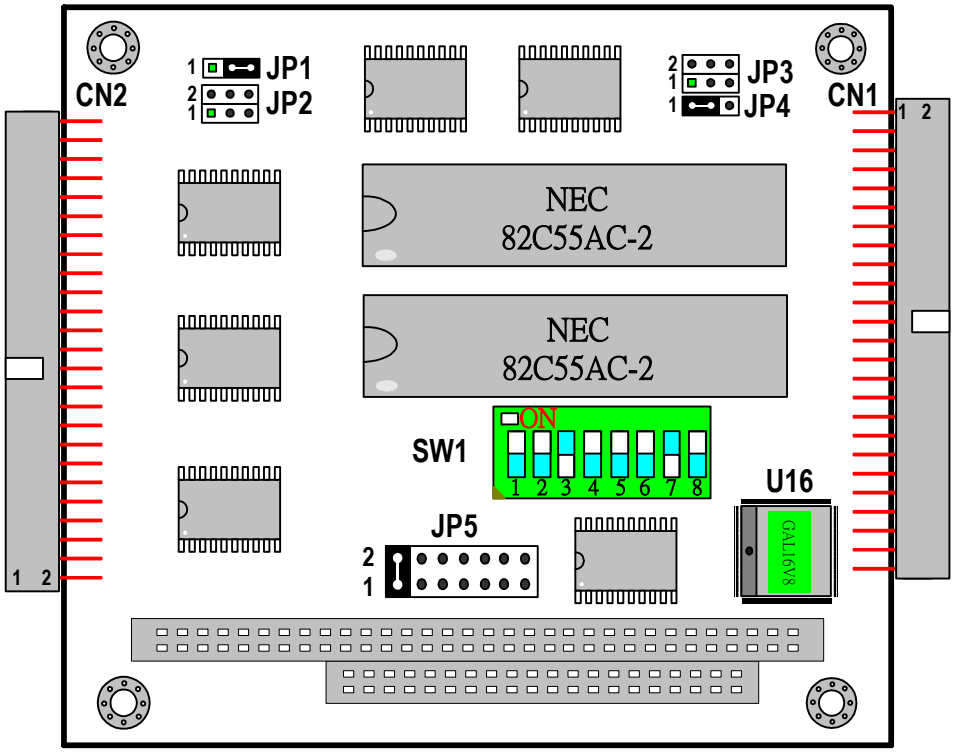
8-channel electromechanical single-pole, double-throw (SPDT) and 16-channel opto-isolated digital I/P panel which is compatible with the EX-9920 Shipped with 3.3 feet (1meter) cable and 50-pin connector.

EX-9956

24-channel electromechanical single-pole, double-throw (SPDT) which can be driven by the EX-9920. Shipped with 3.3 feet (1 meter) cable and 50-pin connector.

Component Locator Diagram

The following figure shows the location of EX-9920's components. All switch and jumper settings in this figure are factory default setting.

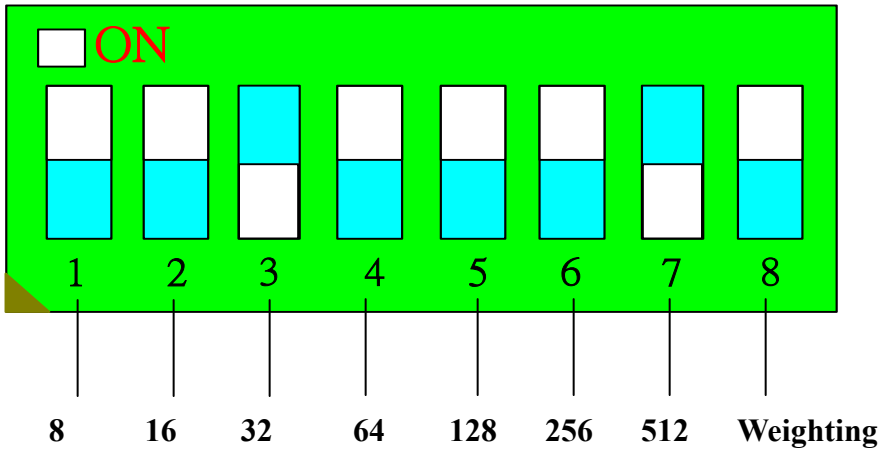


Base Address Switch

The EX-9920 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 EX-9920 base address is set for 220Hex 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.

BASE ADDRESS SWITCH SETTING



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 follow:

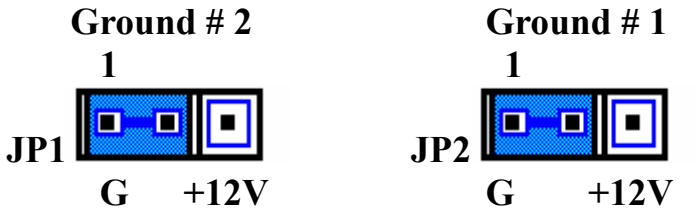
Base Address = 512 + 32 = 544 (Decimal)
=220 (Hexadecimal)

I/O PORT RANGE	DIP SWITCH POSITION							
HEXADECIMAL	1	2	3	4	5	6	7	8
	A1	A2	A3	A4	A5	A6	A7	X
200 – 207	0	0	0	0	0	0	1	X
208 – 20F	1	0	0	0	0	0	1	X
210 – 217	0	1	0	0	0	0	1	X
218 – 21F	1	1	0	0	0	0	1	X
220 – 227 (*)	0	0	1	0	0	0	1	X
.				.				
.				.				
300 – 307	0	0	0	0	0	1	1	X
.				.				
.				.				
3F0 – 3F7	0	1	1	1	1	1	1	X
3F8 – 3FF	1	1	1	1	1	1	1	X

0 = ON, 1 = OFF, X = don't care

(*): Factory Default Setting

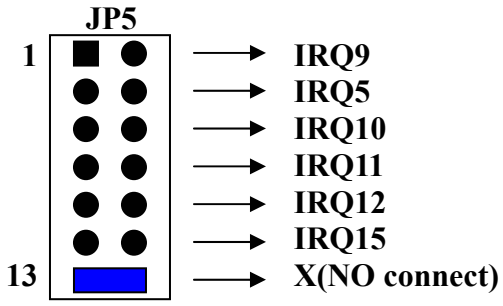
+12V or Ground Selection Jumper



Jumper cap should be placed at pins 1 and 2 of JP1(JP2) to connect pins 2 and 4 of CN2(CN1) connector to Ground. With this configuration, the EX-9920 is compatible with EX9416 and EX-9910 Opto-22 interface panels. When jumper cap is placed at pins 2 and 3 of JP1, pin 2 and 4 of CN2 (CN1) connector are connected to +12V. With this configuration, the EX-9920 is compatible with EX-9954, EX9955 and EX9956 accessory board. The above figure shows the factory default setting for JP1 and JP2.

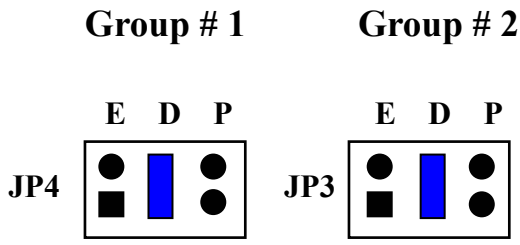
IRQ Level Jumper

Jumper labeled JP5 is used for selecting IRQ level (9(2),5, 10, 11, 12, 15). Below figure gives the jumper configuration and default setting of JP5. Place jumper cap at "x" position, if no interrupt is required.



Interrupt Setting Jumper

EX-9920 provides hardware interrupt function for applications. Port C bit 3(PC3) and bit 7(PC7) are in charge of the task. INTERRUPT ENABLE (INEx) 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 JP5 jumper. Below figure gives the default setting for JP4 and JP3 jumpers.



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

(*): 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	

(*): After interrupt immediately pull down PC3, thus enable other interrupt to happen.
(**): After interrupt immediately pull high PC7, thus enable other interrupt to happen.

Connector Pin Assignment

The EX-9920's 48 DI/O line are divided into two groups; Group #1 and Group #2. The 24 DI/O lines of Group #1 are built in CN1 50-pin connector while the 24 DI/O lines of Group #2 are built in CN2 50-pin connector. Both connector pin assignments are the same and shown in below figure. Through these connectors, the EX-9920 module can be directly connected to OUR's EX9951,EX9954, EX9956 and EX-9956 accessory boards or standard Opto –22 interface plane.

NAME	PIN	PIN	NAME
Ground	50	49	+ 5V
Ground	48	47	PA 0
Ground	46	45	PA 1
Ground	44	43	PA 2
Ground	42	41	PA 3
Ground	40	39	PA 4
Ground	38	37	PA 5
Ground	36	35	PA 6
Ground	34	33	PA 7
Ground	32	31	PB 0
Ground	30	29	PB 1
Ground	28	27	PB 2
Ground	26	25	PB 3
Ground	24	23	PB 4
Ground	22	21	PB 5
Ground	20	19	PB 6
Ground	18	17	PB 7
Ground	16	15	PC 0
Ground	14	13	PC 1
Ground	12	11	PC 2
Ground	10	9	PC 3/INT
Ground	8	7	PC 4
Ground	6	5	PC 5
OPT	4	3	PC 6
OPT	2	1	PC 7/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 handing capability. Refer to Interrupt Setting Jumper section
PC4 – PC7	Port C-upper – four digital I/O lines. The PC7 has interrupt handing capability. Refer to Interrupt Setting Jumper section.
OPT	These pins can be connected to +12V PC power or Ground by jumpering JP1 and JP2. Refer to +12V or Ground Selection Jumper 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 EX-9920 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.

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 lines, just leave the corresponding RP unoccupied.

DI/O LNES	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 isolate system circuitry from external signal. Let the external signal to go through EX-9950 (24 channel opto-isolated D/O panel) or EX9955 (8 channel Relay Output and 16 channel opto-isolated D/I Panel) before reaching the system circuitry.

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 EX-9920 belongs to 16 bit bus option which is designed only to by pass PC/AT bus signal in order to compatible to PC/AT type PC/104 module. The EX-9920 uses only IRQ lines on P2 40-pin connector. If this module is going to plug onto PC type PC/104 bus, do not use IRQ line above 10.

Beside 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 EX-9920 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.)

Module Installation

The EX-9920 board is shipped witch 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 summarize the procedures for installing the EX-9920.

WARNING

TURN OFF the PC and all accessories connected to the PC whenever installing or removing any peripheral board including the EX-9920 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 a non-stackthrough module.**
5. **Put the EX-9920 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 module until inside distance is SPACER's height (0.6"). Restore all the screws.**
8. **Repeat step 6 until all module are set into position.**
9. **Connect cable to EX-9920 (CN2 or CN1) if necessary.**
10. **Replace the case cover and connect all the necessary cables.**

11. Turn on the system power.

CHAPTER 3 REGISTER STRUCTURE AND FORMAT

The EX-9920 occupies 8 consecutive I/O addresses of PC/IO address space. During installation, the first address or base address is determined by setting onboard DIP switch (SW1)

This chapter describes each EX-9920 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 instruction of whatever application languages.

EX – 9920 I/O Address MAP

The 48 digital I/O lines of EX-9920 are arranged into two separated groups. Each group supports 8255 PPI chip mode 0.

The EX-9920 is programmable through configuration registers. By writing to control register, the type of each group may be specified. If a group is configured as a write port, the data drivers 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 register and their locations (R = Read, W = Write, Base = Base address).

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

EX-9920 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

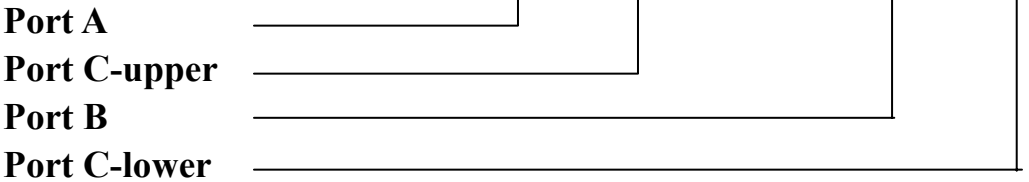
Port C Data Register (Base +2, R/W)

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

Counter Register (Base +3, W)

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



NOTE:

- **PA0 – PA7, PB0 – PB7and PC0 – PC7 bits are associated to pins at CN1 connector.**
- **For D0, D1, D3, D4: 1= Input , 0= Output**

Group #2 Data and Control Registers (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

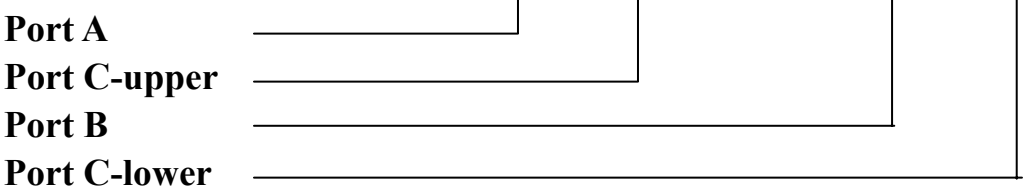
Port C Data Register (Base +6, R/W)

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

Counter Register (Base +7, W)

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



NOTE:

- **PA0 – PA7, PB0 – PB7and PC0 – PC7 bits are associated to pins at CN2 connector.**
- **For D0, D1, D3, D4: 1= Input , 0= Output**

CHAPTER 4 PROGRAMMING

Digital Input and Output

EX-9920 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 as 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, Port 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            'REM Base address
20   Out Port%+3, &H80       'REM Port A, B, C, are all output
30   For J= Port% to Port%+2 'REM Decimal value for Port A to C
40   For X=0 TO 255          'REM Decimal value for 00 to FF Hex
50   Out J, X                'REM Output value X to port J
60   B= INP(J)               'REM Read back from latch
70   PRINT X,B,J             'REM Print input, output value, port
80   NEXT X
90   NEXT J
100  END
```

Interrupt

The EX-9920'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.

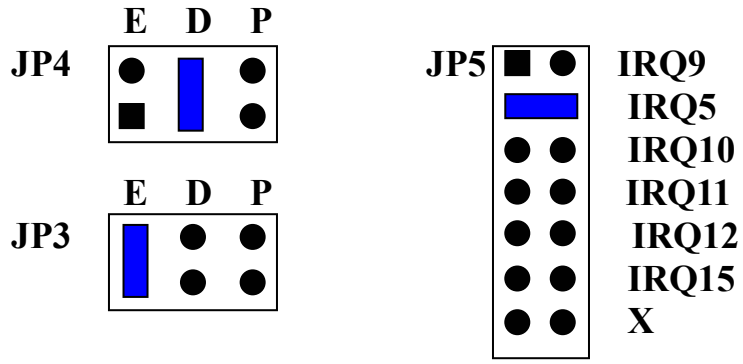
Interrupt can be caused by external input to Port C or by direct output to Port C. This feature can be used to detect external critical signal or to generate an interrupt from program

.

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

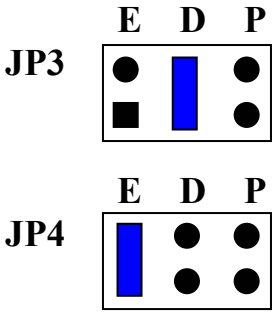
Here we provide a demo program written in Turbo C. Before executing this program, Group #2 PA3 pin must be wired to PC3 pin. This program give a brief description of the interrupt handing capability of the EX-9920. It sets Group #2 Port A for output port and Port C for input port (install corresponding 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:

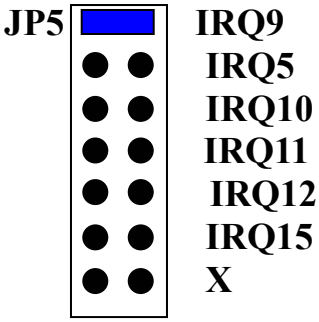


/* DEMO PROGRAM IN TURBO */

```
#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 lntolsr;

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 {
       intoisr= 0;
       outp(base + 4, 0x0);                /* Generate a pulse from Port A */
       outp(base + 4, 0x08);
       delay(50);
       outp(base + 4, 0x0);
       if (intoisr = = 1) {printf("interrupt\n"); intoisr = 0;}
       }while (kbhit ( ) = = 0);
       restore_isr( );                          /* Free ISR */
       outp(0x21, 0xb8);                /* Disable IRQ5 */
}
void interrupt far newisr ( )                /* ISR routine */
{
     intoisr = 1;
     outp (0x20, 0x20);
}

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

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


E D P
JP3
E D P
JP4

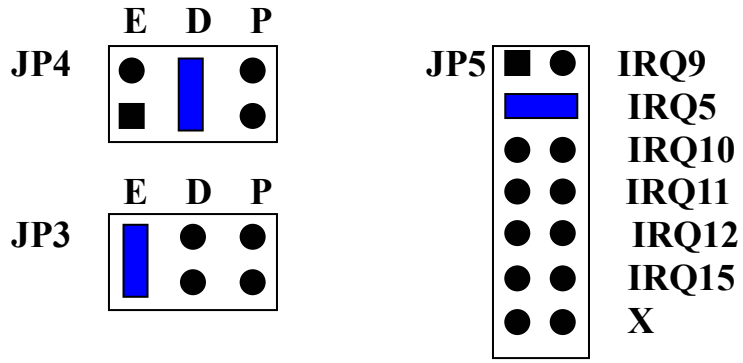

JP5
IRQ9
IRQ5
IRQ10
IRQ11
IRQ12
IRQ15
X

{ DEMO PROGRAM IN PASCAL }

```
{$M 1024, 0, 0}
PROGRAM EX-9920_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
   GETINTVEC (INTP, OLDISR);          {Get old interrupt vector and save it}
   SETINTVEC(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);
SETINTVEC(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 Ground #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:



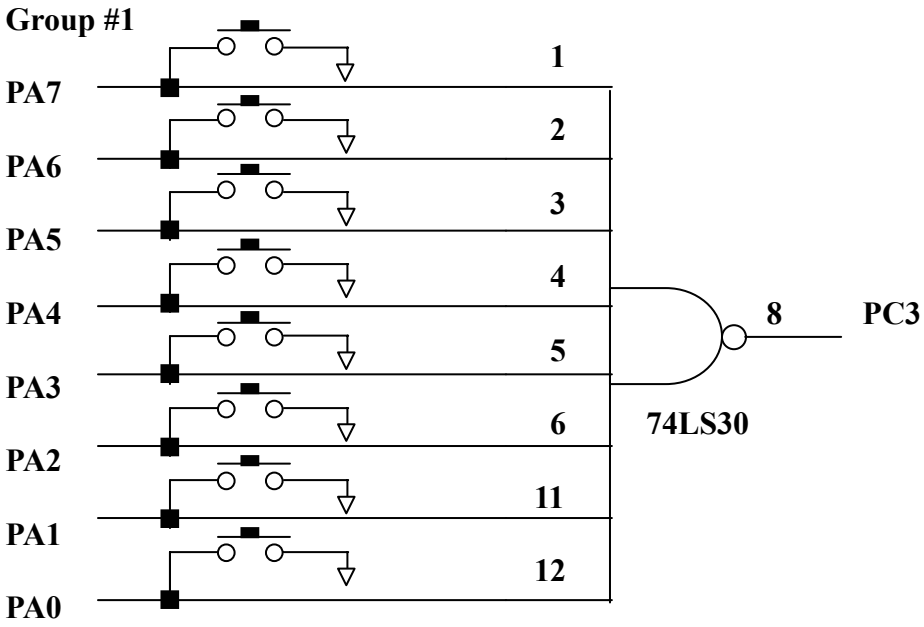
/* DEMO PROGRAM IN MICROSOFT C */

```
   #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(void0                       /* Initiate interrupt */
{
  _disable( );
  int_num=0xod;                           /* 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(void0
{
  _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( );
}
```

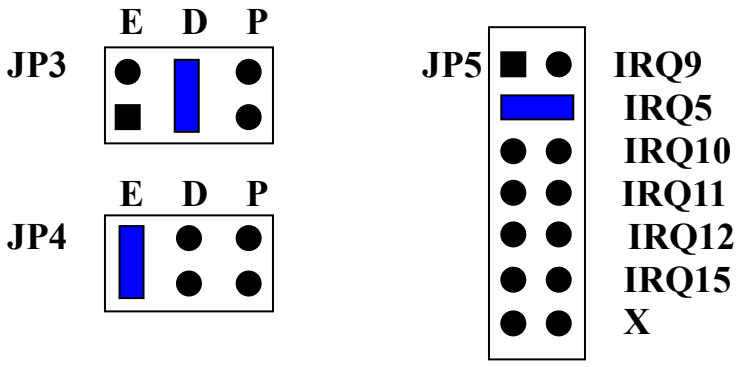
CHAPTER 5 APPLICATION

In this chapter, two simple application examples are provided.

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.

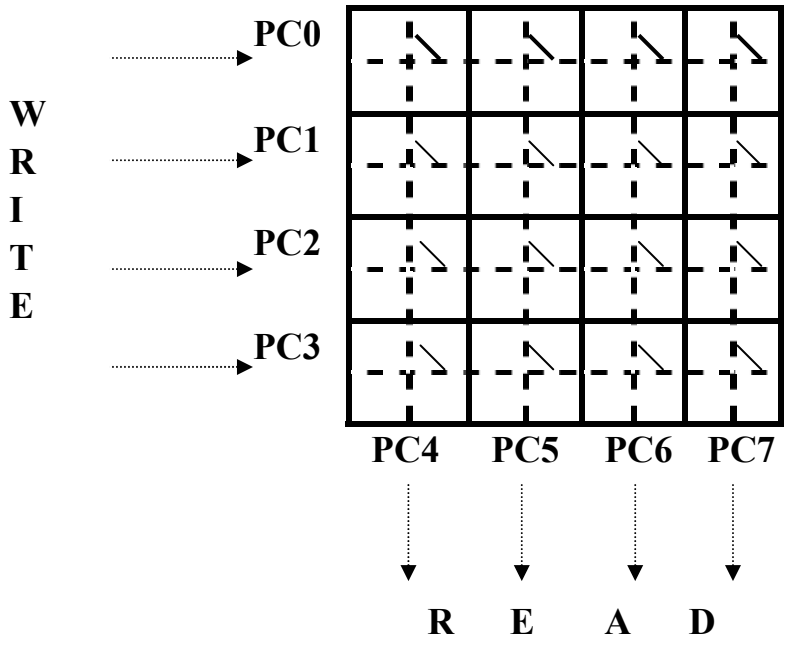


/* DEMO PROGRAM IN TURBO C FOR EVENT TRIGGER */

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

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

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

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

void initiate (void)
{
   disable ( );
   int_num=0xod;                            /* IRQ5 interrupt number */
   old_int=getvect(int_num)                 /* Get old interrupt vector and save it */
   setvect(int_num, isr);                   /* Install the new handler */
   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&0×01)= =0) printf (“PA0 pressed ”);
       if ( (PA&0×02)= =0) printf (“PA1 pressed ”);
       if ( (PA&0×04)= =0) printf (“PA2 pressed ”);
       if ( (PA&0×08)= =0) printf (“PA3 pressed ”);
       if ( (PA&0×10)= =0) printf (“PA4 pressed ”);
       if ( (PA&0×20)= =0) printf (“PA5 pressed ”);
       if ( (PA&0×40)= =0) printf (“PA6 pressed ”);
       if ( (PA&0×80)= =0) printf (“PA7 pressed ”);
       delay (250);
       pc3_high=0;
     }
   }
  close_int ( );
}
```

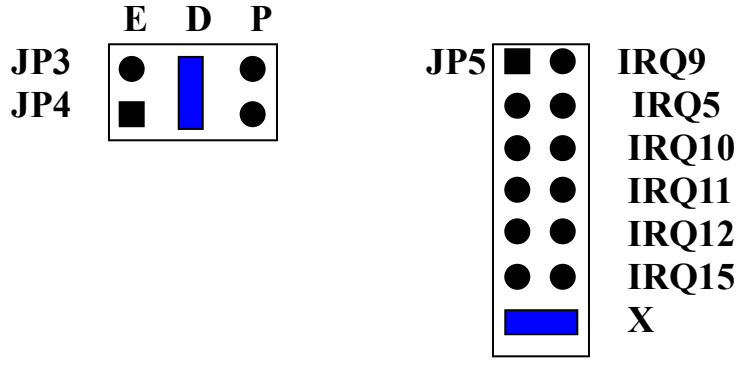
Polling 4*4 Keypad



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. The 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 */

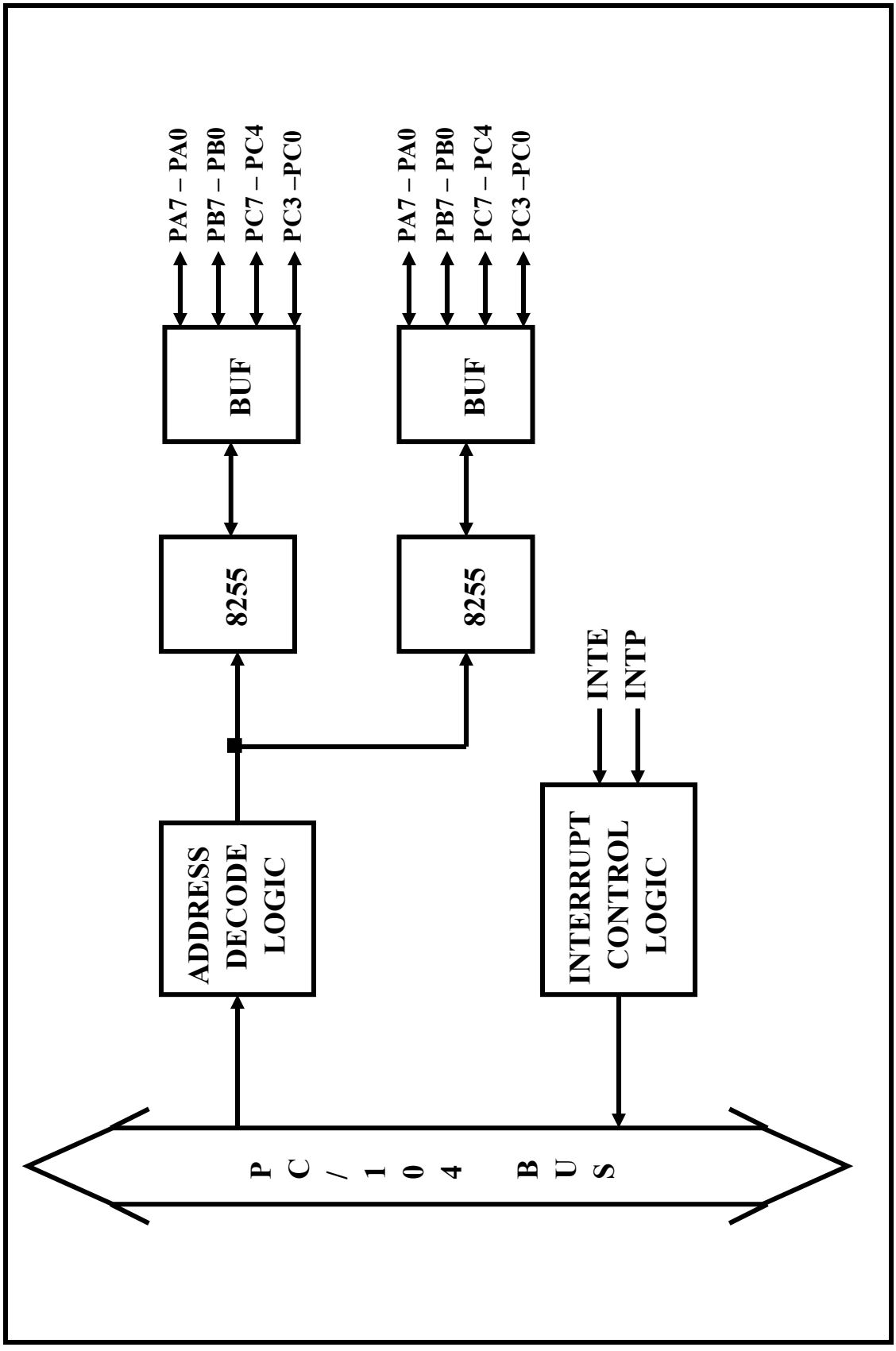
```
#include <conio.h>
#include <dos.h>
#include <stdio.h>

#define BASE 0×220                    /* Default base setting */

void re_key(void)                     /* Find which key pressed */
{
 int g1_pc;
 g1_pc=inp(BASE+2);
 if( (g1_pc & 0×10) = = 0 )          printf(“PC4)     ” );
 else if( (g1_pc & 0×20) = =0 )      printf(“PC5)     ” );
 else if( (g1_pc & 0×40) = =0 )      printf(“PC6)     ” );
 else if( (g1_pc & 0×80) = =0 )      printf(“PC7)     ” );
}

void main ( )
{
 clrscr ( );
 outp(BASE+3, 0×88);
 while(!kbhit ( ) )
 {
  outp(BASE+2, 0×fe);                /* Send pattern 1111    1110 */
  if( (inp(BASE+2) > > 4 ) !=0×0f )   /* Check if key pressed */
  {
   printf(“(PC0,” );
   re_key ( );
   delay(250);
   }
  while( (inp(BASE+2) > > 4 )!=0×0f);   /* Wait until key released */

  outp(BASE+2, 0×fd);                /* Send pattern 1111    1101 */
  if( (inp(BASE+2) > > 4 )!= 0×0f)    /* Check if key pressed */
  {
   printf( “(PC1,” );
   re_key( );
   delay(250);
   }
  while( (inp(BASE+2) > > 4 )!= 0×0f);   /* Wait until key released */

  outp(BASE+2, 0×fd);                /* Send pattern 1111    1011 */
  if( (inp(BASE+2) > > 4 )!= 0×0f)        /* Check if key pressed */
   {
   printf( “(PC2,” );
   re_key( );
   delay(250);
   }
  while( (inp(BASE+2) > > 4 )!=0×0f);    /* Wait until key released */
```

```
  outp(BASE+2, 0×f7);                    /* Send pattern 1111    0111 */
  if( (inp(BASE+2) > > 4 )!= 0×0f)       /* Check if key pressed */
  {
    printf("(PC3," );
    re_key ( );
    delay(250);
    }
while( (inp(BASE+2) > > 4 )!=0×0f);      /* 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
3C0 – 3CF	EGA, reserved
3D0 – 3DF	Color/graphics adapter
3F0 – 3F7	Floppy disk controller
3F8 – 3FF	Primary serial port(COM1)

APPENDIX B BLOCK DIAGRAM

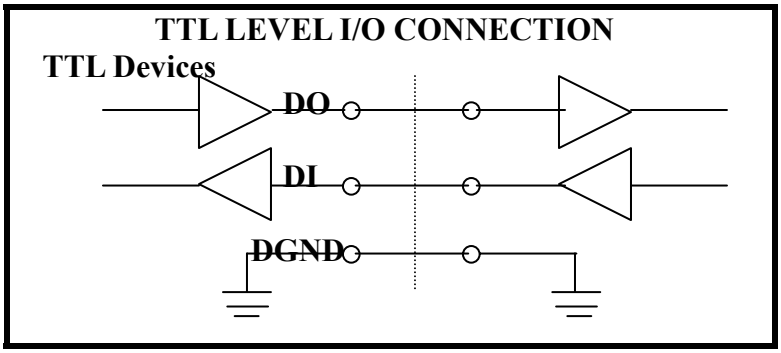


APPENDIX C TECHNICAL REFERENCE

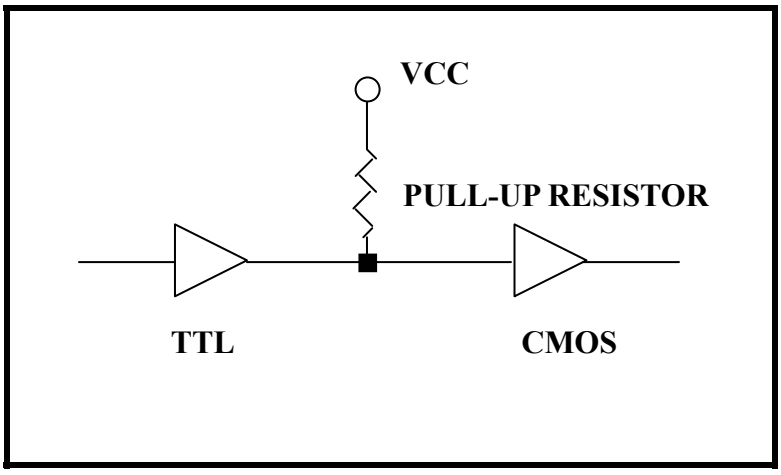
General Usage of Digital Input and Output

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.

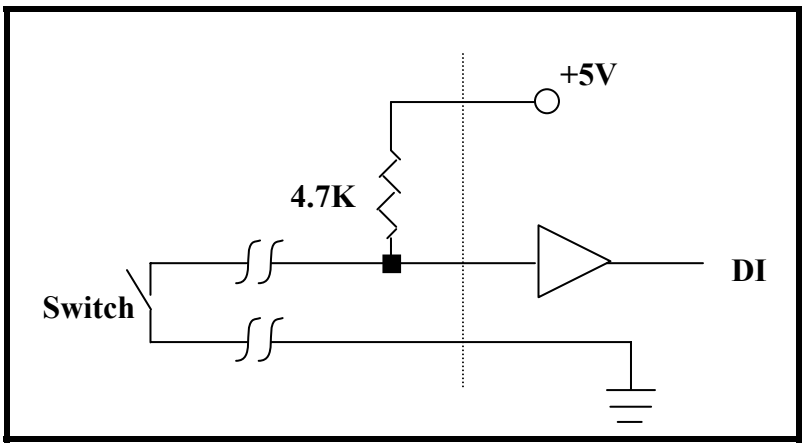
TTL or LSTTL Level I/O Connections



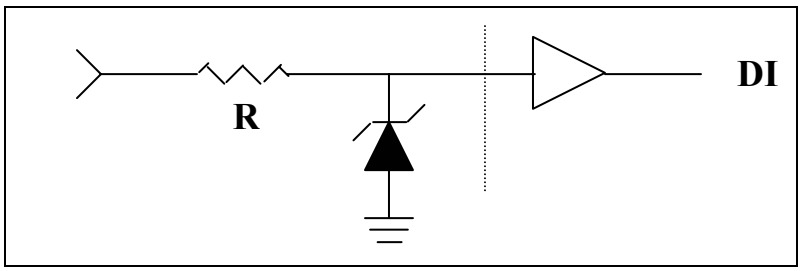
Connection with CMOS Device – Use a pull-up resistor if you wish to interface to CMOS devices. This will raise the logic high output level from its minimum TTL level of 2.4V to +5V suitable for CMOS interface.



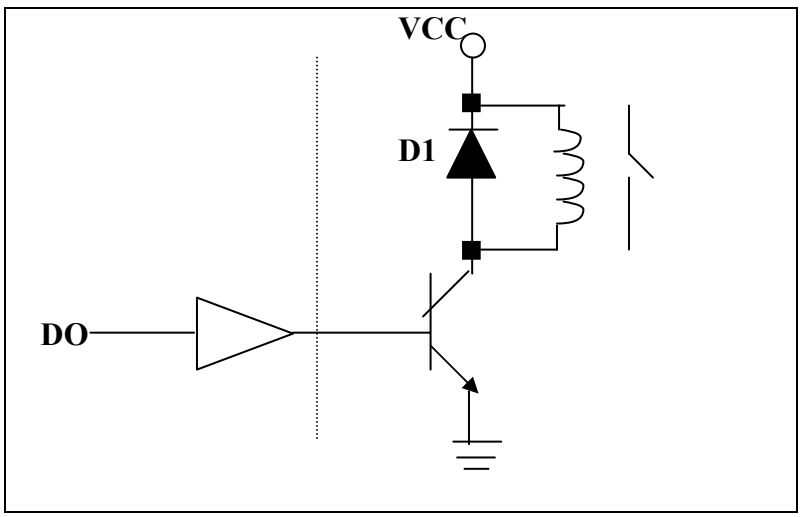
Digital Input for Open/Short Switch Detection – A pull-up resistor must be connected, especially at long distance wiring, to ensure logic high input level.



Digital Input for Large Signal

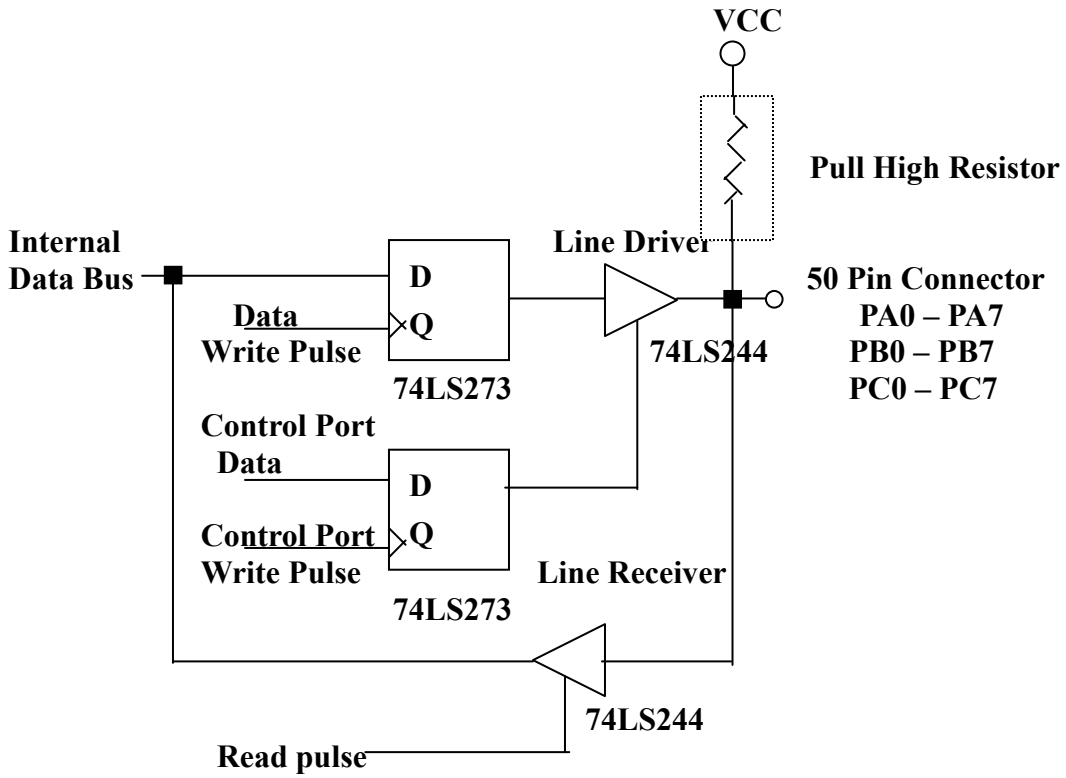


Digital Output for relay Driving – The D1 diode is added to protect the IC driver against the inductive "kickback" from the relay coil.

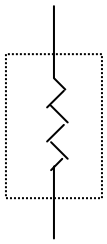


EX9920 Port A, B and C Basic Definition

1.Equivalent ckt for Port A, B and C



NOTE:



Optional Resistor Pack
(Not Installed 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 handing 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 pin on P1 plus 40 pin on P2).

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.

APPENDIX D PROGRAMMING 8254 COUNTER/TIMER

Introduction

NETCOM's EX-9910 module uses INTEL 8254 which consist of three independently programmable 16-bit counters for its timing function. Each counter can be programmed to be divided by number within the range of 2 – 65535. The 8254 is suitable for:

- **Event counter**
- **Delay time generator**
- **Programmable one-shot**
- **Square wave generator**
-

For detailed information, user should refer to the 8254 Programmable Interval Timer data sheet.

Counter Read/Write and Control Register

There are 4 registers need to program 8254 Time/Counter, including there Read/Write and one Control register as follows:

Base +0 through Base +3:

Base +0	**Counter 0**	**Read/Write register**
Base +1	**Counter 1**	**Read/Write register**
Base +2	**Counter 2**	**Read/Write register**
Base +3	**Control word register**	

Read/Write register is used to load divisor to select counter, or Read count from selected counter. Control register is used to determine counter's operation.