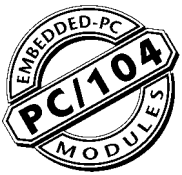
MSI-P900

PC/104 PROTOTYPING CARD

USER MANUAL



PC/104 Embedded
Industrial Series

Microcomputer Systems, Inc.
1814 Ryder Drive ¨ Baton Rouge, LA 70808
Ph (504)769-2154 ¨ Fax (504) 769-2155

CONTENTS

I. INTRODUCTION 1

II. HARDWARE DESCRIPTION 3

A. Card Configuration

B. Card Addressing

C. Example 'C' Program Sequence

APPENDIX 5

Circuit Diagrams

MSI-P900

I. INTRODUCTION

The MSI-P900 is an 8-bit PC/104 stackthrough prototyping card for simple, rapid hardware implementation of user designs. The card is I/O mapped using 16-bit addressing and provides a fully decoded bi-directional 8-bit data bus (D0 - D7), five low enable chip selects (CS0* - CS4*) decoded from jumper selectable addresses A8 - A15. Note: an asterisk following the pin assignment denotes low level active state; e.g., CS1* means CS1 is active lo (0). Also provided are jumper selectable IOR*, IOW*, SA0 - SA3, IRQ3 - IRQ7 & IRQ9, +5V, -5V, +12V, -12V and ground, as shown in the block diagram of Figure 1.

Sixty percent of the card area accommodates user supplied 8-pin thru 40-pin wire-wrap DIPs, SIPs, resistors, capacitors, transistors, etc., as well as standard I/O connectors (up to 40-pin) with 0.1" grid spacings. Solder pad connections are provided on the bottom surface for easy connections of devices to +5V and ground. Top and bottom views of the card are shown in Figure 2.

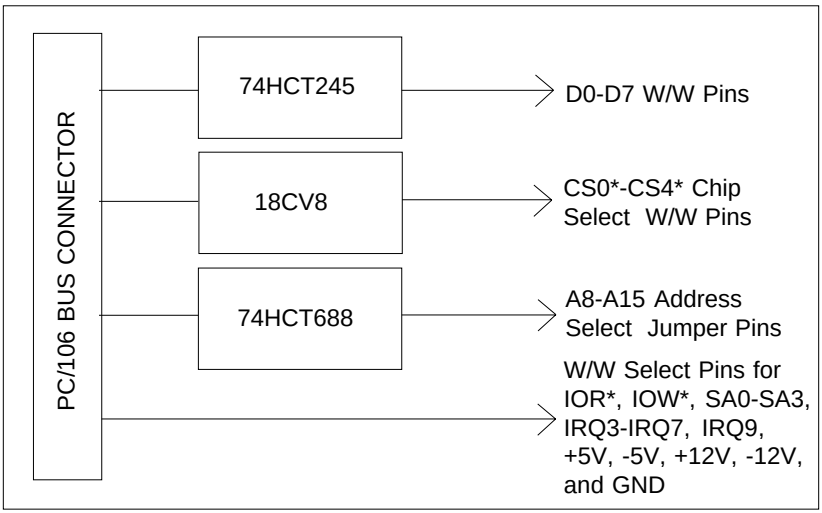


Figure 1. Block diagram of the MSI-P900 Prototyping Card.

TOP VIEW

BOTTOM VIEW

Figure 2. MSI-P900 Layout.

II. HARDWARE DESCRIPTION

A. Card Configuration

The bus interface for the card provides wire-wrap jumpers for the various functions as shown in Figure 2. The connections for D0-D7, IOR*, IOW*, CS0*-CS4*, IRQ2-IRQ7, IRQ9, SA0-SA3, +5V, -5V, +12V, -12V and GND are provided by wire-wrap headers which are positioned on the bottom of the card. This provides easy access to wire-wrap DIP sockets and other components positioned in the prototyping area (see Fig. 2, TOP VIEW). Other signals that are required can be wire-wrapped directly from the PC/104 bus connector pin.

The prototyping area will accommodate various combinations of 0.3" and 0.6" width DIP sockets, SIPs and 0.1" grid connectors at sites P1 and P2. The bottom surface of the card has numerous solder pads (0.062" square) to accommodate connections for GND and +5V.

B. Card Addressing.

The card address is set by installing appropriate jumpers for A7 thru A15. An uninstalled jumper for a given address bit sets the bit to 1 (true) and an installed jumper sets the bit to 0 (false). Addresses A7 thru A15 define the *base address* of the card from 0000H to FF80H on integral 80H boundaries, where H denotes a hexadecimal number. To assign a base address of 3080H, for example, install jumpers A8, A9, A10, A11, A14 and A15.

When assigning a base address, be sure that 80H contiguous addresses are available in which there are no I/O address conflicts. Since 64,536 addresses are available in the address space, this normally should not be a problem. The 80H space is required in order to accommodate the five chip selects CS0* thru CS4*.

The chip select addresses for CS0*-CS3* are the following.

Chip Select	Address
CS0*	base addr + 0
CS1*	base addr + 10H
CS2*	base addr + 20H
CS3*	base addr + 30H
CS4*	base addr + 40H

C. Example 'C' Program Sequence

For a simple 'C' program illustration, consider a case in which the prototype devices are configured for the following parameters.

1) A base address for the card of 7080H (insert jumpers for A8, A9, A10, A11 and A15).

2) An input device at address 7093H (base + 13H, which activates CS1* and A0 =1, A1 = 1, A2 = 0 and A3 = 0). The input device also uses IOR* during a read cycle.

3) An output device at address 70B1H (base + 31H, which activates CS3* and A0 = 1, A1 = 0, A2 = 0 and A3 = 0). The output device also uses IOW* during a write cycle.

A program sequence to read the input device, store the value at input_data, and write the result to the output device is

```
#define     base_address          0x7080
#define     input_device_addr     0x13
#define     output_device_addr    0x31

int input_data;/* storage for input data */
        .
        .
        .
input_data = inp(base_address + input_device_addr);
outp( base_address + output_device_addr, input_data);
        .
        .
        .
```

APPENDIX