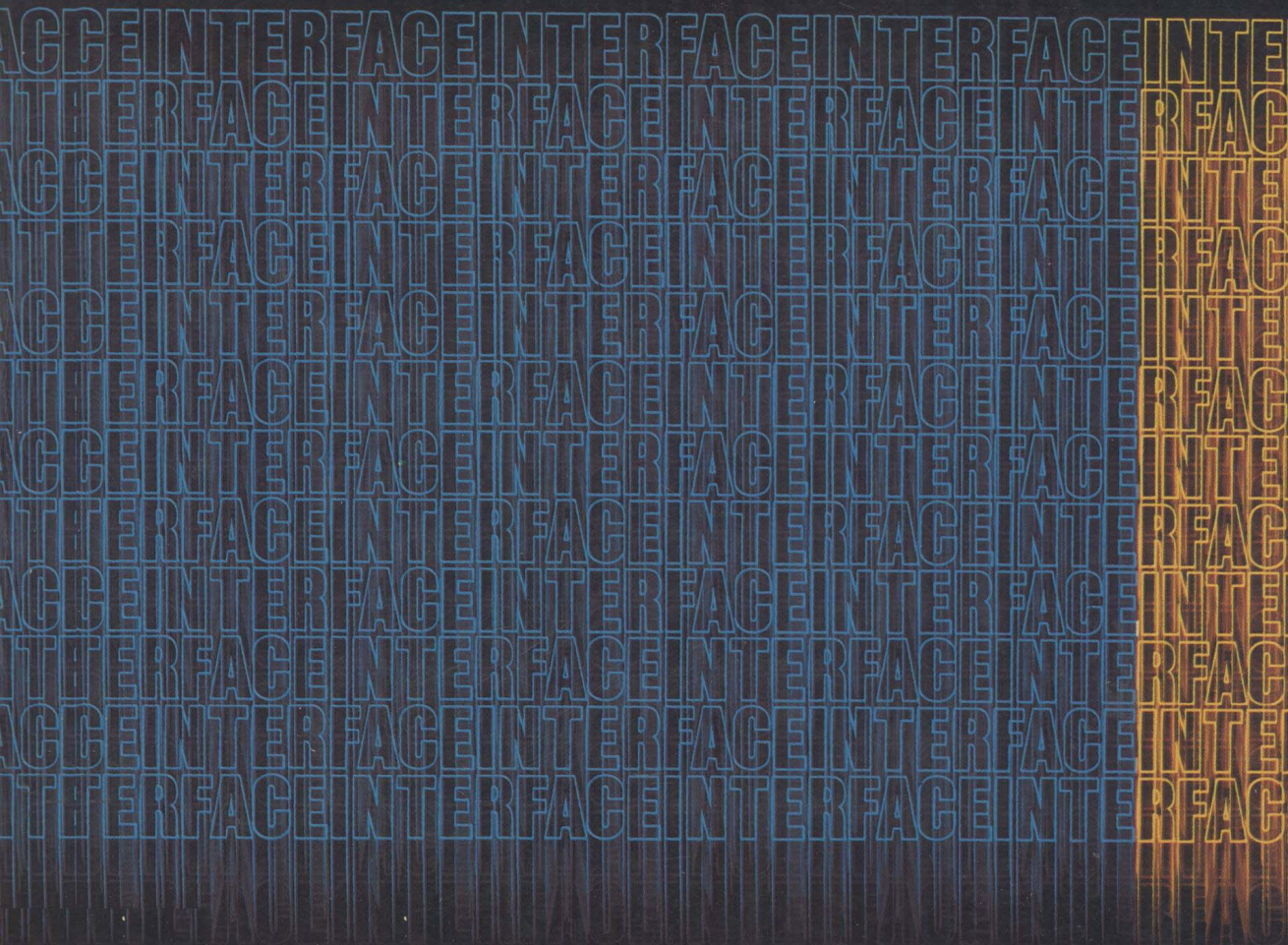
RICHARD C. HALLGRE

interface project
for the TRS-8
(MOD



Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632 A SPECTRUM BOOK

RICHARD C. HALLGREN

INTERFACE PROJECTS FOR THE TRS-80

Library of Congress Cataloging in Publication Data

Hallgren, Richard C.
Interface projects for the TRS-80.

"A Spectrum Book"
Includes bibliographical references and index.
1. Computer interfaces—Amateurs' manuals. 2. TRS-80 (Computer) I. Title.
TK9969.H254 621.3819'592 82-343
ISBN 0-13-469437-6 AACR2
ISBN 0-13-469429-5

ISBN 0-13-469437-6
ISBN 0-13-469429-5 {PBK.}

10 9 8 7 6 5 4 3 2 1

Editorial/production supervision by Alberta Boddy
Page layout by Gail Collis
Manufacturing buyer: Barbara A. Frick

Prentice-Hall International, Inc., *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Whitehall Books Limited, *Wellington, New Zealand*

Richard C. Hallgren, Ph.D., is an assistant professor at Michigan State University. He has written numerous articles and is the author of *Interface Projects for the Apple II* (Prentice-Hall, Inc.).

CONTENTS

1

INTRODUCTION

The rapid expansion of the electronics industry, along with its ability to mass produce reliable, large-scale integrated circuits, has given us scientific calculators and multifunctional digital watches for under $20, as well as "personal computers" starting from $200. In 1971 the Intel Corporation had the honor of initiating this revolution by introducing the first commercially available microprocessor. While the performance of this device was limited by its relatively slow central processing unit and by its use of a 4-bit data bus, its acceptance was so dramatic that, within a two-year period, the cost of a single unit had dropped from $200 to under $20. Today, it is a rare individual who does not have a microprocessor of some type in their home.

As I have talked to owners of personal computers, it has become apparent that a certain group of them would like to utilize their machines as extensively as possible, both at home and at their places of employment. This desire inevitably results in the need for interface circuitry that connects the computer to external devices and then allows users to monitor and control their environment. Based on the response of readers to several articles that I have written, I have come to the conclusion that there is a great need for a book not only that covers the theory behind interfacing a computer to external devices, but that also gives a broad selection of interface circuits that can be built and expected to work. So what I have attempted to do in this book is to provide a

number of examples of interface circuits, ranging from the very simple to the somewhat complicated. I have included software for each circuit and can assure you that both the hardware and the software have not only been tested together, but have been used in some practical applications.

The result is a document that explains the theory behind computer interfacing and that gives you the choice of either building the circuit as is, or of modifying it to meet your specific needs. The hardware for these projects was designed and the components selected so that construction and check-out would be a straightforward matter. The circuits have been chosen to offer something of interest and application to a broad range of individuals: the hobbyist, the experimenter, the manager of a manufacturing plant, and the engineer or scientist in a research facility. The intent is to enable people to more fully utilize the computers in practical and interesting ways.

The book assumes that readers have a fairly good understanding of the commands in TRS-80, Model III BASIC, and have written some of their own programs. Since some of the supporting software will be using Z-80, machine code, you are encouraged to become familiar with the Z-80 instruction set. Programming examples using both Model III BASIC and Z-80 machine language have been provided throughout the book. While construction of the circuits is straightforward, it should be understood that a certain amount of experience and a certain level of expertise are expected. Previous construction of a commercial kit would probably qualify most individuals. So, with all this in mind, let's get started with the job of connecting your Mod III to the outside world.

2

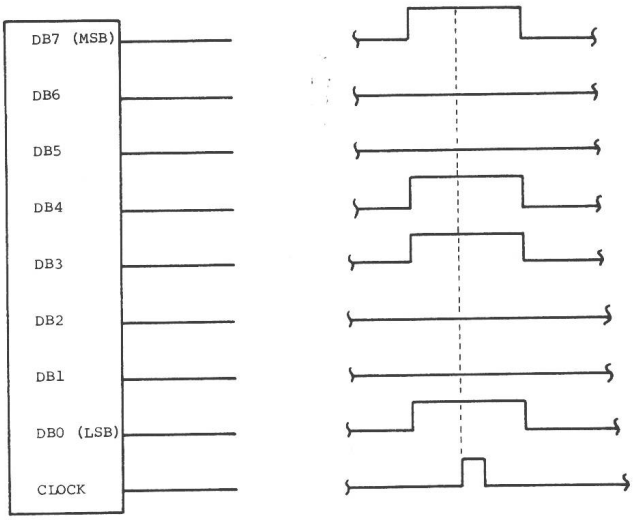
REVIEW OF DATA TRANSFER FORMATS

Before we get into actual circuit designs, we need to spend some time reviewing basic interface concepts. I would recommend that even experienced circuit designers read this section to familiarize themselves with the terminology that I use.

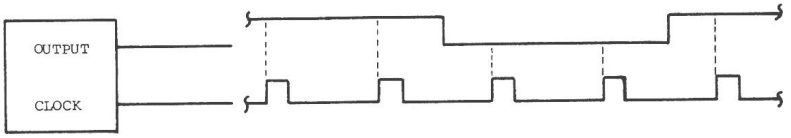
We can define an *input/output (I/O) port* as a collection of electronic circuits, under the control of the computer, which route data to and from an external peripheral device. The key words in this definition are "data" and "external." The route by which the data flow to or from the external device is called the *port.* A line printer is a common example of an external device; the computer sends characters to be printed to the printer, and, in some cases, the printer sends control signals back to the computer. The purpose of these signals is to regulate the flow of the data, the interaction being called *handshaking.*

Ports can be constructed so that data are handled in either a serial or a parallel format (Figure 2.1).

Parallel transfer of data involves the mass transfer, at a given point in time, of several bits of data, where the number of bits being transferred is usually equal to the word size of the computer. For the case of the Model III this is equal to 8 bits. In general, the number of bits transferred at one time will be equal to the size of the data bus. For example, if we were working with a Z8000, which has a 16-bit data bus, a parallel transfer would involve 16 bits. For ap-



a) Parallel format



b) Serial format

FIGURE 2-1 Comparison of parallel and serial out format. The clock pulse is used to signal the external device that the signals are stable. For both examples, the data word being transmitted is binary 10011001.

plications where high rates of data transfer are desired, such as data transfer between a computer and a floppy disk, the transfer is usually handled with a parallel format. Computers handle parallel communication relatively easily, and even though multiconductor cables are required, the total expense is not too great because the distance between devices is usually quite short.

Serial transfer of data involves transmitting individual bits of data, one bit at a time. Microcomputers do not normally communicate in a serial format, so there is usually no single machine language command that facilitates this type of operation. As a result, we need to add either a software subroutine or an additional piece of hardware to accomplish a serial transfer of data. This adds to the system expense, but it gives us the capability of transferring data over increased distances at relatively high data rates. The Electronics Industry Associa-

tion (EIA) RS-232C electrical specification for serial transfer of data is the standard for industrial applications. This specification defines the following voltage levels: a logic level 1 is called a *mark* and is considered to be any voltage level more negative than −3V. A logic level zero is called a *space*, and it is considered to be any voltage level more positive than +3V. In general, designers use +12 and −12-V levels for the logic 0 and 1 states. In addition to specifying voltage levels, the EIA also defines the standard RS-232C connector to be a 25-pin, D subminiature type (commonly referred as a DB-25). The pin assignments and their functions are listed in Table 2.1

Before we look at specific examples, we should take a look at the architecture of the Z-80 to determine how we can communicate in an orderly manner with several peripheral devices that may be connected to the same data bus and to the same address bus. The Model III computer used a Z-80 type of microprocessor to perform the logical, mathematical, and decision-making operations necessary for high-speed operation of the computer. This microprocessor is an 8-bit device that combines the bus structure of the 8080 with an instruction set that includes 644 differant commands. The bidirectional data bus is used to transfer information both to and from the central processing unit. The address bus has 16 lines that can be used to uniquely define 65,536 addressable locations. The Z-80 generates several signals that are used both internally and externally to supervise and manage the flow of information. Since data can flow in only one direction at a time, and since the data are usually directed to a specific

TABLE 2-1 EIA standards for DB-25 connector pin assignments when used for communication between RS-232C systems.

Pin 1	PGND - Protective Ground This is chassis or equipment ground. It may also be tied to signal ground.
Pin 2	TD - Transmit Data This is the serial data from the terminal to the remote receiving equipment. When no data is being sent it is in a marking (1) condition.
Pin 3	RD - Receive Data This is the serial data from the remote equipment which is transmitted to the terminal.
Pin 4	RTS - Request to Send Controls the direction of data transmission. In full-duplex operation an "on" sets transmit mode and an "off" sets non-transmit mode. In half-duplex operation an "on" inhibits the receive mode and an "off" enables it.
Pin 5	CTS - Clear to Send Signal from the modem to the terminal indicating ability to transmit data. An "on" is "Ready" and an "off" is "not ready."
Pin 6	DSR - Data Set Ready Signal from the modem to the terminal. An "on" condition indicates that the modem is ready.
Pin 7	SGND - Signal Ground
Pin 8	CD - Carrier Detect An "on" indicates reception of a carrier from the remote data set; "off" indicates no carrier is being received.
Pin 20	DTR - Data Terminal Ready "On" connects the communication equipment to the communications channel; "off" disconnects the communications equipment from the communications channel.
Pin 22	RI - Ring Indicator An "on" indicates that a ringing signal is being received on the communications channel.

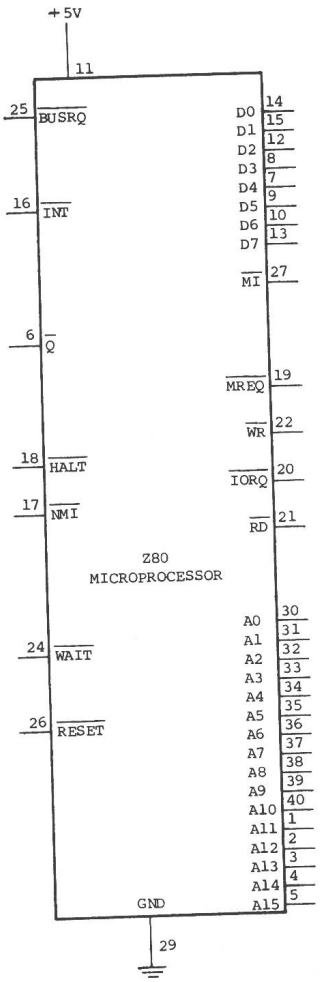


FIGURE 2-2
Z-80 pin configuration.

device or memory location, these control lines provide an essential coordinating function.

Figure 2.2. shows the pin configuration for the Z-80, and Table 2.2 lists the control signals used when communicating with external devices. The Z-80 belongs to the one-address architectural class of microprocessors; that is, most memory-related instructions reference a single memory location. While the computer is officially an 8-bit device, several instructions treat pairs of 8-bit registers as a single, 16-bit word. These 16-bit registers are advantageous for handling some complex calculations.

When the microprocessor is reading a data byte from memory, the address of the desired memory location is placed on the address bus. Information stored in that particular memory location is strobed onto the data bus and flows from

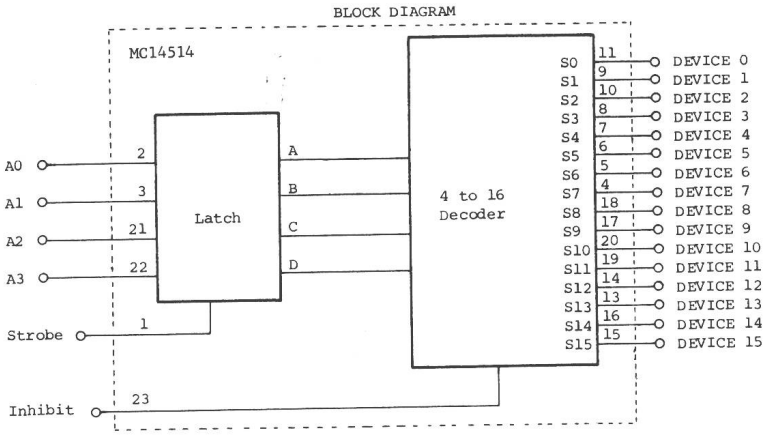
Signal Name	Description
A0	Address Output
A1	Address Output
A2	Address Output
A3	Address Output
A4	Address Output
A5	Address Output
A6	Address Output
A7	Address Output
D0	Bidirectional Data Bus
D1	Bidirectional Data Bus
D2	Bidirectional Data Bus
D3	Bidirectional Data Bus
D4	Bidirectional Data Bus
D5	Bidirectional Data Bus
D6	Bidirectional Data Bus
D7	Bidirectional Data Bus
$\overline{\text{MREQ}}$	Memory Request
$\overline{\text{RD}}$	Read
$\overline{\text{WR}}$	Write
$\overline{\text{IORQ}}$	Input/Output Request

TABLE 2-2 Microprocessor input/output. There are eight address lines which will be used to select a specific external device. There are eight bidirectional data lines which will carry data to and from the CPU and the external devices whether the CPU is going to send data or is expecting to receive data.

memory to a data register in the central processor. Data being written to memory is handled by a reversed set of operations.

The Z-80 has a specific set of instructions that are unique to input and output operations. The operation of these instructions is similar to that of the memory-referenced instructions, with the exception that a different set of control lines is used to indicate to external devices that the central processor wishes to communicate with them and not with the memory. The least significant 8-bits of the address bus are used to uniquely address 256 possible I/O ports. Various methods can be used to decode these lines to signal a specific device that it is being addressed. Figure 2.3 shows a simple but effective method for decoding the four least significant bits of the address bus into 16 unique control lines. For normal operation, the INHIBIT control line would remain low. Upon execution of an input or output command, a processor control signal (IN* or OUT*) would be connected to the STROBE control line, used to strobe the current contents of the address bus into the decoder. The decoded output would then be used to indicate to a specific external device that the data bus was available for its specific use. This particular circuit allows up to 16 devices to be uniquely addressed; if increased addressing capability is required, the circuit of Figure 2.4 can be used.

Four control signals that we will be particularly interested in keeping track of are MREQ*, I/O REQ*, RD*, and WR*. Any memory-referenced command is accompanied by the MREQ* line going to the low state; an I/O command is accompanied by the I/O REQ* line going to the low state. The central processor indicates to the external device whether a read or a write operation is occurring by causing either the RD* or the WR* line to go low. As you might have ex-



DECODE TRUTH TABLE (Strobe = 1)

	DATA INPUTS				SELECTED OUTPUT
INHIBIT	A3	A2	A1	A0	LOGIC "1"
0	0	0	0	0	Device 0
0	0	0	0	1	Device 1
0	0	0	1	0	Device 2
0	0	0	1	1	Device 3
0	0	1	0	0	Device 4
0	0	1	0	1	Device 5
0	0	1	1	0	Device 6
0	0	1	1	1	Device 7
0	1	0	0	0	Device 8
0	1	0	0	1	Device 9
0	1	0	1	0	Device 10
0	1	0	1	1	Device 11
0	1	1	0	0	Device 12
0	1	1	0	1	Device 13
0	1	1	1	0	Device 14
0	1	1	1	1	Device 15
1	X	X	X	X	All Outputs = 0

X = Don't Care

FIGURE 2-3 4-16 line address decoder. Each device is selected by a unique combination of the address lines AO-A3.

pected, when the RD* line goes low, the computer is transferring data from the external device to the central processor. Likewise, when the WR* line is low, the computer is transferring data from the central processor to the external device. Figure 2.5 shows how three of these control lines could be decoded to give signals that will assist us in controlling the flow of data to and from the computer. Fortunately, the designers of the Mod III have decoded these signals for us, making our life a little easier. Our strategy, as we design interface circuits, will be to use the address lines to select a particular device and to use the decoded control lines to tell the device whether the computer is going to send data to it or receive data from it.

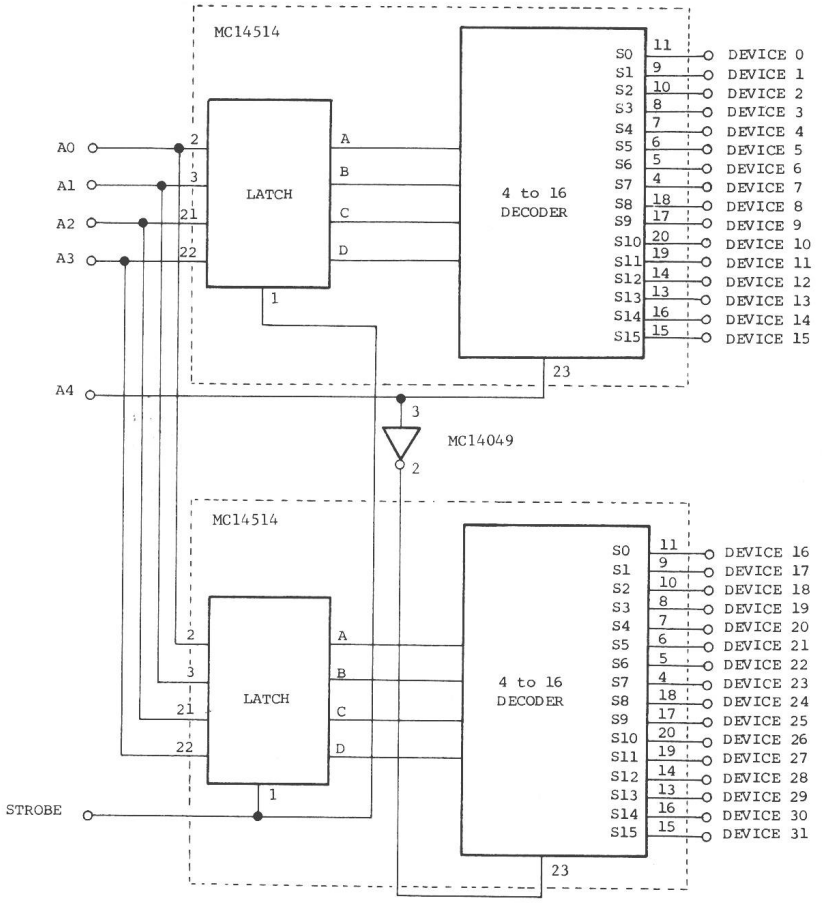
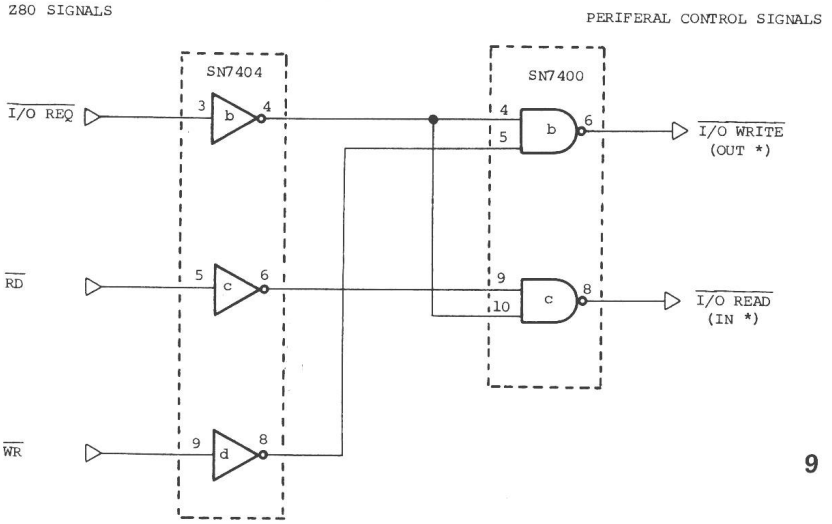


FIGURE 2-4
Extension of a 16 device address decoder into a 32 device decoder.

FIGURE 2-5 Control signals on the Z-80 microprocessor. The Z-80 uses a variety of control signals to keep data flowing at the right time and in the right direction. Three control signals are used as follows: the I/O REQ line goes to a low state when an input/output (I/O) operation is in progress; the RD line goes low when the processor is reading data from memory or from a peripheral device; the WR line goes low when the processor is writing data to memory or to a peripheral device. The RD and WR signals control the direction that data flows along the bidirectional data bus. Monitoring these three lines gives us all the information necessary to support I/O decoding functions.



PARALLEL DATA FORMAT

Transmission and reception of data in a parallel format combines the advantages of high transfer rates and low cost. As mentioned earlier, parallel data transfer involves the mass movement of several bits of data at one time. It becomes a relatively straightforward task to construct both input and output ports since all the control, data, and address lines that we will need are available on the Model III I/0 bus connector (Figure 2.6). Table 2.3 lists the signals appearing on this connector and their functions. Figures 2.7a and 2.7b show examples of practical circuits that could be connected directly to the I/O bus connector, providing one 8-bit, latched output port and one 8-bit input port. To access the I/O port bus, it is necessary to first perform the BASIC statement, OUT 236,16 or the assembly language command, OUT (OECH), 10H. After this, whenever you perform the BASIC statement, OUT N,D or the assembly language command, OUT (N), A, the following happens:

1. The device address (N) is strobed onto the address bus.
2. The contents of the accumulator (D) are strobed onto the data bus.

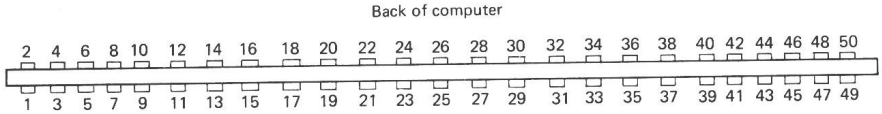
Since the data are valid for only a few clock cycles (perhaps 500 ns) a set of clocked flip-flops (IC3 and IC4) is provided so that the data are latched and consequently stable, until the next OUT command to this port is executed. The circuit works in the following manner:

1. Whenever address line A0 and A1 are high and low respectively and the OUT* line is low, an output strobe pulse will appear on pin #12 of IC1.
2. Whenever this strobed output pulse appears, the contents of the data bus will be transferred into the clocked flip-flops and will appear on the latched data output lines.

will appear on the latched data output lines.

External devices transferring data into the computer can be connected directly to the data bus. But, to add an additional margin of safety (that is, to help keep you from "smoking" your computer), I have included a 3-state buffer. The 3-state buffer is used as a gate that allows signals from the peripheral device to be placed on the data bus at an appropriate time. The Model IIIrequires that the

FIGURE 2-6 Output pin configuration for the Model III I/O bus connector. Refer to Table 2-3 for a description of each pin.



P/N	SIGNAL NAME	DESCRIPTION
1	D0	Bidirectional Data Bus
2	GND	Signal Ground
3	D1	Bidirectional Data Bus
4	GND	Signal Ground
5	D2	Bidirectional Data Bus
6	GND	Signal Ground
7	D3	Bidirectional Data Bus
8	GND	Signal Ground
9	D4	Bidirectional Data Bus
10	GND	Signal Ground
11	D5	Bidirectional Data Bus
12	GND	Signal Ground
13	D6	Bidirectional Data Bus
14	GND	Signal Ground
15	D7	Bidirectional Data Bus
16	GND	Signal Ground
17	A0	Address Output
18	GND	Signal Ground
19	A1	Address Output
20	GND	Signal Ground
21	A2	Address Output
22	GND	Signal Ground
23	A3	Address Output
24	GND	Signal Ground
25	A4	Address Output
26	GND	Signal Ground
27	A5	Address Output
28	GND	Signal Ground
29	A6	Address Output
30	GND	Signal Ground
31	A7	Address Output
32	GND	Signal Ground
33	IN*	Peripheral Read Strobe Output
34	GND	Signal Ground
35	OUT*	Peripheral Write Strobe Output
36	GND	Signal Ground
37	RESET*	System Reset
38	GND	Signal Ground
39	IOBUSINT*	Interrupt Input
40	GND	Signal Ground
41	IOBUSWAIT*	I/O Bus Wait
42	GND	Signal Ground
43	EXTIOSEL*	I/O Bus Select
44	GND	Signal ground
45	N.C.	No Connection
46	GND	Signal Ground
47	XMI*	Standard Z-80 Signal
48	GND	Signal Ground
49	IORQ*	Input/Output Request
50	GND	Signal Ground

NOTE: "*" means logical "0" true input or output

TABLE 2-3 Functional description of pins on the Model III I/O bus connector.

device being interfaced pull the EXTIOSEL* line (pin #43) down to a 0 during the execution of an IN command. This tells the bidirectional data bus to strobe data into the accumulator. To access the I/0 port, it is necessary to first perform the BASIC statement, OUT 236,16 or the assembly language command, OUT (0ECH), 10H, After this, whenever you perform the BASIC statement, INP (N) or the assembly language command, IN A, (N), the following happens:

1. The device address (N) is strobed onto the address bus.
2. The contents of the data bus are strobed into the accumulator (A).