PRACTICAL INTERFACING TECHNIQUES FOR MICROPROCESSOR SYSTEMS

James W. Coffron

William E. Long

PRACTICAL INTERFACING TECHNIQUES FOR MICROPROCESSOR SYSTEMS

James W. Coffron

William E. Long

PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging in Publication Data

COFFRON, JAMES.
Practical interfacing techniques for microprocessor systems.

(Prentice-Hall series in microprocessor technology)
Includes index.
1. Computer interfaces. 2. Microprocessors.
I. Long, William E. II. Title.
III. Series.
TK7887.5.C6 1983 001.64'404 82-21460
ISBM 0-13-691394-6

Editorial production/interior design: Margaret McAbee
Cover design: Zeppi Long
Manufacturing buyer: Anthony Caruso

ISBN 0-13-691394-6
Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-691394-6

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*
EDITORA PRENTICE-HALL DE BRASIL LTDA., *Rio de Janeiro*

PREFACE

This is a book about interfacing equipment to enhance and enlarge the performance capacity of existing mini, micro, or home computer systems. The focus of the discussion is on the computer system whose utilization has reached its original design limits, or which lacks certain capabilities that the user would like it to have. It is written for the engineer, technician, or serious hobbyist who may be exploring interfacing techniques for the first time, or who is seeking practical suggestions and additional ideas that will be useful in this work.

Computer users often find that larger memory is needed to accomplish desired tasks because the needs have outgrown the equipment. Or, the computer may be needed for a new application that involves the addition of I/O ports to achieve. Whether the objective is larger memory, increased flexibility, or a different way of communicating with the outside world, the problem is one of interfacing. In writing this book, the attempt has been to select and give priority to the topics that offer the greatest practical value for solving frequently encountered interfacing problems.

The topics covered here include interfacing to ROM, to static RAM, to dynamic RAM, to various I/O port architectures, and the ECL and CMOS logic families, controlling a scanned 7-segment display, A-to-D and D-to-A conversion, the use of a programmable I/O chip, the S-100 bus, serial/parallel data transmission, a video keyboard terminal, and more. A complete chapter is devoted to interfacing to the TRS-80, a popular home com-

puter. To gain a better grasp of what is covered in the book the reader is asked to examine the table of contents—the topics are all there.

A word now about what may be the most important aspect of all—the treatment of the material.

First, this book is hardware-oriented, it is about circuits, devices, the way they work and how to use them. Every effort has been made to present practical, usable information that includes circuit connections, devices, and pin numbers, and shows *how* to go about things as well as *why*. Each topic concentrates on the kind of information one must have in order to build and test the circuit successfully.

Second, a major effort has been made to present all of the information in clear and understandable language. How each circuit works is carefully explained, and many figures illustrate and illuminate significant points. One last item: potential trouble areas and practices to avoid are discussed at appropriate places to help readers avert many frustrating difficulties.

The computer is no mystery today. Complex, yes, when confronted as a large system. But much of the intimidation that early computers brought to most of us has vanished as better ways of looking at the organization, the sequences of operation, and circuit details have emerged. Hopefully, this book will help you to pull back the curtain of mystery even further.

Acknowledgements

We wish to acknowledge and thank all who have assisted and contributed to the development of this manuscript. First, to our families we owe a special debt for their indulgence, for encouraging us, and for protecting and guarding our time. We also wish to give special thanks to Margaret McAbee, Production Editor, for her sensitive and thoughtful work and for the extraordinary care and effort she expended in our behalf; to Judy Winthrop, Art Editor, for her keen sense of aesthetics and her unfailing aura of good cheer which made working with her such a pleasure; to Zeppi Long, Graphic Artist, for interpreting the nature and spirit of the manuscript in a way that is distinctive and harmonious with the Prentice-Hall Series on Microprocessor Technology.

James W. Coffron

William E. Long

CONTENTS

1

HARDWARE ARCHITECTURE FOR INTERFACING (3-BUS ARCHITECTURE)

1–1: Introduction

In this chapter we introduce and describe a microprocessor system architecture that is applicable to resolving interfacing problems which may be encountered when developing or expanding a particular system. This architecture is valid for most 8- and 16-bit microprocessor systems and is equally valid for most home computer systems. A clear understanding of this architecture can reduce interfacing problems to their simplest forms.

Many examples of interfacing to microprocessor and home computer systems are presented in this text, all of which rely on the architecture presented in this chapter. Fortunately, this understanding of architecture also applies to the effective troubleshooting of microprocessor systems. An understanding of this material will minimize any difficulty you may have when approaching a new system or new system application.

The architecture described in this text is called 3-bus system architecture and is described in detail in the text, "*Practical Hardware Details for the 8080, 8085, Z80, and 8600.*"* Although not all manufacturers use the name 3-bus architecture, this organization is common to most microprocessor systems.

To describe the architecture here, we use two different microprocessors—the 8085 and the Z80. From the detailed discussion of these

*James W. Coffron, Prentice-Hall, 1981.

microprocessors you should be able to relate these details to any other 8- or 16-bit microprocessor. We then design these microprocessors into the 3-bus system architecture to illustrate our point and next show how any system hardware operation may be analyzed or thought of in this way.

Let us begin with a definition of a system bus. For this text, a system bus is defined as a collection of electronic signals and signal lines or paths that are grouped according to function. In a block diagram, each bus has the same point of origin and the same point of destination in the system. Stated in another way, all signals in a given bus have a common function. A simple example of a system bus is the power bus. This is a bus with only one signal in it. The point of origin is the system power supply. The point of destination for this signal is the system components.

1–2: The 3-Bus System Architecture

The three major busses used to describe the digital action in a microprocessor system or home computer are the

(1) address bus,

(2) the data bus, and

(3) the control bus.

We include home computers because these systems have the same attributes as other microprocessor controlled systems. A home computer or personal computer is a sub-set of all the microprocessor systems used in the world today.

Every hardware action that takes place in a microprocessor controlled system can be performed using the 3-bus approach. Notice that 3-bus system architecture is not a simplified structure used to describe a complex action—rather, it is an accurate model that presents the complex action of a microprocessor system in a different, easier-to-understand way.

As stated earlier, the 3-bus model can accurately describe the seven hardware actions that occur in microprocessor controlled systems. These hardware actions are:

1. write data to memory *from* the CPU (control processing unit or microprocessor)
2. read data from memory *to* the CPU
3. write data to an output port *from* the CPU
4. read data from an input port *to* the CPU
5. interrupt actions *by* the CPU

6. access the memory directly (DMA), controlled *by* the CPU
7. manipulate internal registers *by* the CPU.

Note that these hardware actions take place as a result of software instructions.

Each hardware activity in a microprocessor system falls into one of these seven operation categories. However, microprocessor systems do not have to use all seven hardware operations to be of value; many very useful systems are designed to employ only three or four of the seven possible actions. But no matter how complex the operation or how long a controlling software program is, every system is executing only the seven hardware operations listed.

Figure 1–1 is a block diagram for a typical microprocessor system. Notice the three major system busses that are used. It will help to refer to this diagram while we go through the first four hardware actions, in which the use of the 3-bus architecture will be clearly visible. Later we will show

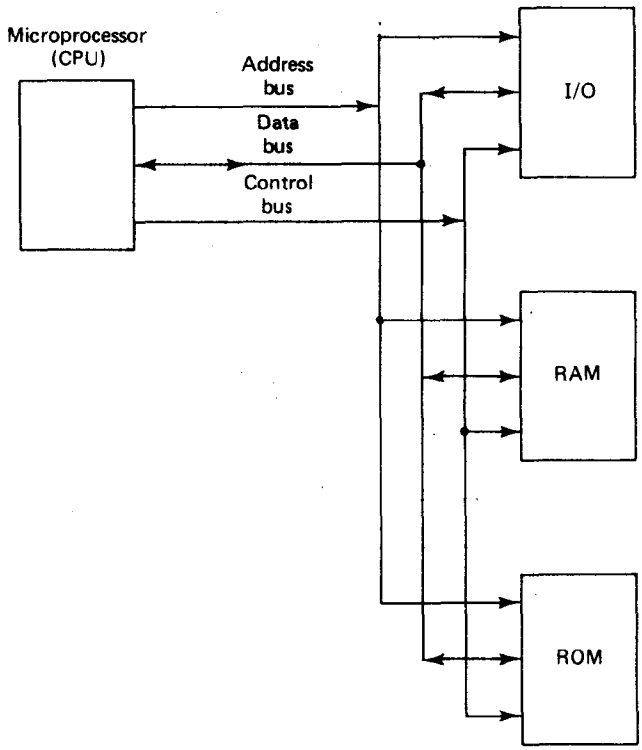

Figure 1–1 Block diagram of 3-bus architecture for a typical microprocessor system

exactly how 3-bus architecture can be used with a real microprocessor system—the 8085. Most 8-bit microprocessors can be designed into this type of system. When interfacing to a home computer or any microprocessor system, one should view the hardware in this way for it will make the overall operation of such systems much easier to understand.

1–3: Writing Data to Memory from the CPU

Figure 1–2 shows a timing diagram of a typical write cycle for a semiconductor memory device. When a mircoprocessor writes data to memory the timing sequence required by the memory must be followed. Any microprocessor must do this. Note that the timing diagram shown does not mention a particular microprocessor. It simply indicates how a memory device must be electrically communicated with to perform a successful write operation, no matter what microprocessor initiates the communication.

For a successful write operation, each system bus must perform its function or job, and each bus may be thought of as being independent of the others. For a write operation, the address bus will provide the address in memory, which the CPU will write data to. The address bus does not actually write the data; the address bus has no electrical knowledge of the type of communication or when the communication will take place; its only function is to supply the address and enable the correct path for the communication to occur.

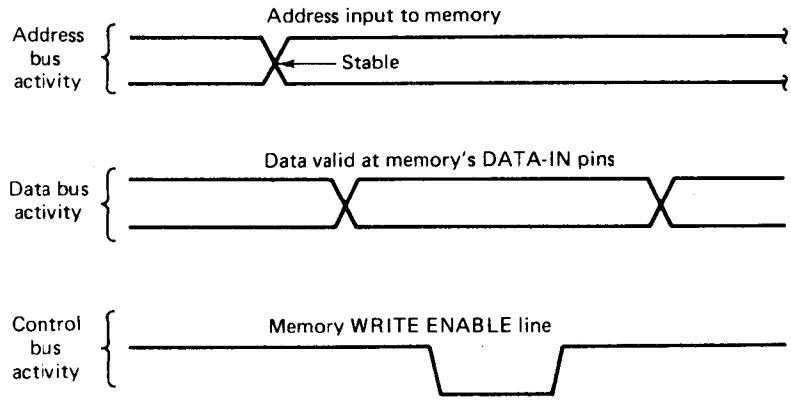

Figure 1–2 Timing diagram of a WRITE cycle in a microprocessor system. Note the activity of each bus in this operation.

The data bus has the job of supplying the physical means for transfer of data between the enabled device and the CPU. During this particular hardware operation, the data bus will physically connect the memory data input lines to the microprocessor's data output lines. Again, the data bus may be thought of as completely independent of the address bus or control bus. The data bus has no electrical indication of where the data is coming from. Its only job is to enable the electrical path for data that is to transfer between the CPU and the address selected by the address bus.

The data bus has two possible directions for the electrical data to flow during any system communication:

1. from the CPU to the system (write operation)
2. from the system to the CPU (read operation).

In each case, the data bus lines are controlled by—that is, carry data from—a different source. The destination is also different in each case. During a read operation, the data bus has its origin somewhere in the system other than the microprocessor, and the destination for the data is the microprocessor. During a write operation, the data has its origin in the microprocessor, and the destination for the data is somewhere in the system other than the CPU.

The final action in a memory write sequence occurs when the system control bus asserts the memory write enable line. This action strobes the data into the memory. One function of the control bus is to define the type of hardware operation that is occurring in the system at any given time. In the particular system we are now discussing four hardware operations are possible:

1. Memory read (MEMR)
2. Memory write (MEMW)
3. Input read (IOR)
4. Output write (IOW)

For each different hardware operation, there is a unique control bus line. Since there are four possible hardware events, there are four system control lines.

A second function of the control bus is to provide the start and stop pulses for the data transfer. During the memory write operation, the control bus does not enable, or assert, the MEMW control line until the microprocessor has valid data on the system data bus. See Figure 1–3. After a fixed length of time the control bus unasserts the MEMW control line. The hardware of the system uses the MEMW control line to assert the write

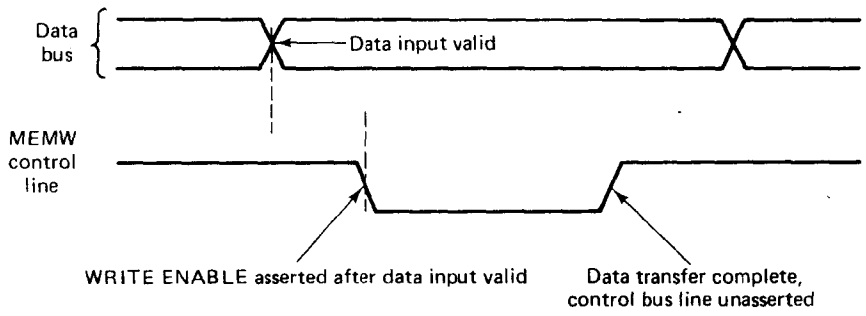

Figure 1–3 Timing diagram showing how the system control bus asserts the WRITE ENABLE line at the time the CPU has valid data placed on the system data bus

enable pulse to the memory address designated by the data from the address bus.

Figure 1–4 shows how the 3-bus architecture conforms exactly to the timing pattern that is needed for electrical communication with memory. The information on the address bus and on the data bus can be thought of as static signals, which means that these two types of signals stay useful, or valid, for the entire time required to accomplish data transfer. However the control bus signals are timed signals—these signals are valid only part of time during data transfer.

From this example of writing data to system memory it can be seen that each bus in the system has a different function. Each bus is independent of the other but all three must operate, for if any one of the three busses fails to perform its function, the data transfer is unsuccessful.

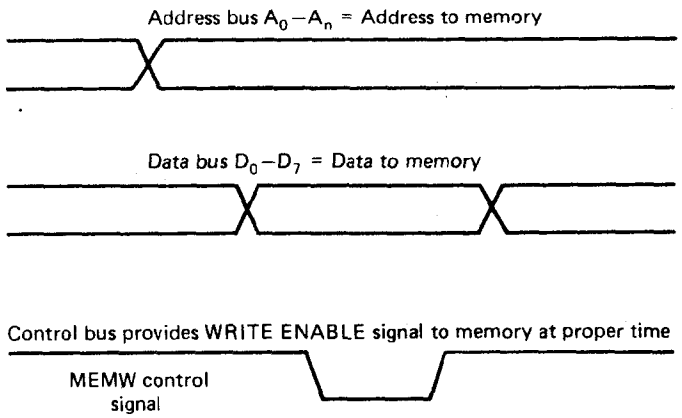

Figure 1–4 Timing diagram showing the job of each system bus during a memory write operation. If any bus fails to perform correctly, the operation will not be electrically valid.

1–4: Reading Data from Memory

Figure 1–5 shows the timing required for any digital hardware to electrically read data from system memory. Notice that once again no mention of a particular microprocessor is made. If any microprocessor is to read data electrically from a semiconductor memory, it must observe this sequence. Let us show how the 3-bus architecture realizes this timing.

The address bus has the same function as that described in the memory write sequence—to provide the correct system address of the memory location from which to read data. The data bus lines provide the physical means for the data to transfer. In this hardware transfer operation, the data originates at the memory's data output lines and its destination is the microprocessor data input lines.

Finally, the control bus asserts a unique control line, (MEMR), to start the data transfer, at which time the data from the system memory becomes valid on the system data bus and the microprocessor strobes binary infor-

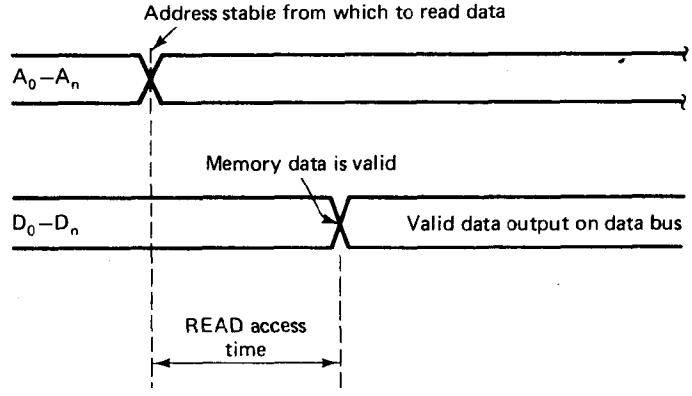

Figure 1–5 Diagram showing the system timing required for digital device to read data from a semiconductor memory

mation on the data bus into an internal register. After a fixed time, the MEMR control line becomes unasserted, which terminates the data transfer.

Figure 1–6 shows how the 3-bus architecture realizes the timing for the memory read operation shown in Figure 1–5.

1–5 Writing Data to an Output Device

An output device in a microprocessor system may be defined as any destination other than memory for data from the CPU. Examples of output devices are CRT display, floppy disc, or cassette tape. The only difference