AD-A044 607 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO F/G 9/2
HARDWARE DESIGN AND CONSTRUCTION OF A MICROPROCESSOR LABORATORY--ETC(U)
1976 J H HILL
UNCLASSIFIED AFIT-CI-77-9 NL

1 OF 2
ADA
044607

HARDWARE DESIGN AND CONSTRUCTION OF A MICROPROCESSOR LABORATORY.

By

JAMES HAROLD HILL, JR.

A THESIS PRESENTED TO THE GRADUATE COUNCIL OF
THE UNIVERSITY OF FLORIDA
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF ENGINEERING

UNIVERSITY OF FLORIDA

1976

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: CI 77-9	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Hardware Design and Construction of a Microprocessor Laboratory	5. TYPE OF REPORT & PERIOD COVERED: Thesis
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): JAMES H. HILL, JR. MAJOR, USAF	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: AFIT Student at University of Florida, Gainesville, Florida	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: AFIT/CI Wright-Patterson AFB OH 45433	12. REPORT DATE: 1976
	13. NUMBER OF PAGES: 103 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE AFR 190-17.

JERRAL F. GUESS, Captain, USAF
Director of Information, AFIT

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Attached

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ACKNOWLEDGEMENTS

I am deeply grateful to the United States Air Force for the opportunity of pursuing graduate studies under the Air Force Institute of Technology (AFIT) program.

I especially want to express my sincere appreciation to my thesis advisor, Dr. K. L. Doty, for his advice, encouragement and patience.

Special thanks also go to fellow students V. Moore, J. Bush, M. De Martinis and S. Ackerman, who provided valuable suggestions during various stages of the project, and to T. Deakin who constructed much of the hardware.

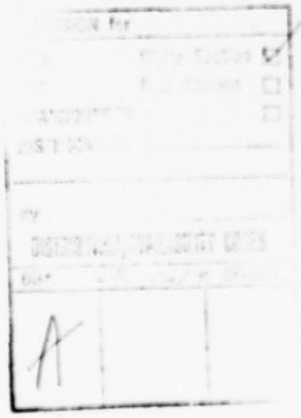


TABLE OF CONTENTS

		Page
ACKNOWLEDGMENTS		ii
ABSTRACT		v
CHAPTER		
I	INTRODUCTION	1
	1.0 General	1
	1.1 Microprocessor Background	1
	1.2 Microprocessor Characteristics	2
	1.3 Microprocessor Education	4
	1.4 Project Initiation	5
	1.5 Microprocessor Laboratory	6
	1.6 Chapter Summary	8
II	DIGITAL COMMUNICATIONS CIRCUITS	9
	2.0 General	9
	2.1 Concept of the Link	9
	2.2 Transmission Line Considerations	10
	2.3 Line Driver/Receiver Considerations	11
	2.4 Standardized Interfaces	11
	2.5 Special Interfaces	12
	2.6 Balanced Differential Circuit	14
	2.7 Modes of Operation	14
	2.8 Multiplex Considerations	15
	2.9 Line Matching	20
	2.10 Long Transmission Line Effects	25
	2.11 Selecting Line Drivers and Receivers	26
	2.12 Chapter Summary	28
III	DATA TRANSMISSION PROTOCOLS	29
	3.0 General	29
	3.1 SDLC	29
	3.2 HDLC	37
	3.3 CAMAC	41
	3.4 GPIB	46
	3.5 Chapter Summary	50

TABLE OF CONTENTS (continued)

		Page
IV	MICROPROCESSOR LABORATORY BUS	51
	4.0 General	51
	4.1 Serial Versus Parallel Transmission	51
	4.2 Laboratory Bus Structure	52
	4.3 Bus Protocols	54
	4.4 Polling	58
	4.5 Clocking	60
	4.6 Clock and Poll Circuit Operation	61
	4.7 Chapter Summary	63
V	LABORATORY COMPUTER SYSTEM	64
	5.0 General	64
	5.1 MC 6800 Microprocessor	64
	5.2 Initial Considerations	65
	5.3 System Selection	65
	5.4 Keyboard and Encoder	66
	5.5 CT-1024 Terminal	68
	5.6 MC 6800 Microcomputer	71
	5.7 Peripheral Interface Adapter (PIA)	81
	5.8 Chapter Summary	84
VI	M6800 INTERFACE AND DATA TRANSFER OPERATIONS . . .	85
	6.0 General	85
	6.1 Station Decode Circuitry	85
	6.2 Gaining Bus Control	87
	6.3 Interrupt Prioritizing	88
	6.4 Interrupt Handling	89
	6.5 Master to Slave Data Transfer	90
	6.6 Slave to Master Data Transfer	94
	6.7 Port Convention	96
	6.8 Chapter Summary	97
VII	CONCLUSIONS AND RECOMMENDATIONS	98
	7.0 Conclusions	98
	7.1 Recommendations	98
APPENDIX		100
BIBLIOGRAPHY		101
BIOGRAPHICAL SKETCH		103

Abstract of Thesis Presented to the Graduate Council
of the University of Florida in Partial Fulfillment
of the Requirements for the Degree of
Master of Engineering

HARDWARE DESIGN AND CONSTRUCTION OF A MICROPROCESSOR LABORATORY

By

James H. Hill, Jr.

June, 1976

Chairman: Keith L. Doty
Major Department: Electrical Engineering

This thesis describes the design and realization of a microprocessor laboratory that provides a capability to train students in computer interfacing, digital design using microprocessors, and software development. Four data transmission interface standards (SDLC, HDLC, CAMAC, GPIB) are discussed followed by a description of the unified bus structure chosen for the University of Florida microprocessor laboratory. Also included is a description of the M6800 microcomputer system used in the laboratory prototype along with details on bus data transfer operations.

Keith L. Doty
Chairman

Chapter I

Introduction

1.0 General

Included in this chapter is some background information on microprocessors to illustrate the importance of this relatively new device and its many possible applications. Also included is the rationale behind the need for adequate training facilities to enable the computer engineering student to obtain meaningful experience using microprocessors. Finally, the concept of the University of Florida's microprocessor laboratory is presented.

1.1 Microprocessor Background

Not since the advent of the transistor and the integrated circuit has there been a development that has created as much excitement in digital design as the microprocessor. First available commercially about five years ago, the microprocessor now finds itself in the enviable position of being the logical choice for many digital systems applications, replacing hard-wired logic.

The microprocessor is an outgrowth of large-scale-integration (LSI) technology which has revolutionized the electronics industry. As this technology gradually evolved, it became possible to provide greater gate density per chip area with acceptable production yields. This development paved the way for the "computer on a chip" - the microprocessor.

Applications of the microprocessor are many indeed. Initially it was devoted primarily to such tasks as instrumentation, data acquisition, communications, process control, etc. Now their applications range all the way from children's toys and games to business and scientific data processing. About the only thing that rivals their broad range of usefulness is the proliferation of companies that manufacture them and the intensive competition in marketing strategies.

Electronic Design News, in its survey of the microprocessor market revealed a total of 29 different microprocessors on the market in November 1975 with numerous others in the offing (2). IBM's microprocessors did not appear in this list since their production of these devices is used in-house. It should be noted, however, that IBM is perhaps the largest U. S. manufacturer of microprocessors.

As the advantages of the microprocessor become more apparent to digital designers, these devices will find increasing use in digital applications. It has been estimated that the microprocessor market will grow from 73 thousand units made in the U. S. for world consumption in 1974 to three million units in 1978 (8).

1.2 Microprocessor Characteristics

The microprocessor is generally characterized by the use of LSI circuitry for the central processing unit (CPU) and the ability to process both arithmetical and logical data in a bit parallel fashion under program control. To some the word microprocessor has another common meaning - that of a microprogrammable processor. Those readers interested in an excellent tutorial paper on microprocessors should

consult Reyling's paper (21). Madnick (16) provides some interesting observations on microprocessors and their role in future computer systems.

The advantages of the microprocessor derive from the following attributes which determine microprocessor usefulness:

1) Weight - Ranging upward from a few pounds, most microprocessor based systems are quite light, even with supporting chips and power supply.

2) Space - Present space requirements are less than one half of corresponding minicomputers. The space attribute alone opens new applications areas where space is a major constraint. Truly portable systems have become practical.

3) Cost - Cost of microprocessor based systems is generally lower than conventional digital designs for two reasons. First, the cost of the microprocessor itself is quite low due to substantial savings in volume production. The other reason is due to simplification of the product cycle (design, production, and maintenance) and the shortening of the product development time. This is an extremely important point in a dynamic market area such as computers, peripherals, etc. (5).

4) Reliability - Microprocessor faults are more difficult to test and isolate than standard LSI chips, but the basic reliability is much better than hard-wired logic due to far fewer connections outside the chip.

5) Power Consumption and Dissipation - Power consumption is lower in most microprocessor based systems. For example, a traffic

controller using the INTEL 8008 microprocessor used 50 watts compared to 200 watts for a previous system using hard-wired IC technology (8).

6) Re-utilization- When a given microprocessor system becomes obsolete, it is not necessary to scrap the entire system. Rather, the microprocessor can be used in a different application by writing a new software package for it.

Microprocessors are not without some disadvantages. Depending on the technology involved, some require a variety of power supplies while others require a single +5 volt supply. Some require an assortment of support chips (clock generator, interface adapters, bus controllers, ets.). All require some amount of programming which can be considered a disadvantage because of the generally high costs associated with customized applications software. One method of attacking this latter problem is to make use of generalized applications packages wherever possible. As the size of microprocessor systems grow and additional memory capacity is available at lower cost through technological breakthroughs, microprocessor users will be able to take advantage of high level languages and problem oriented languages which should help to reduce both the cost and time required for generating applications programs (16).

Despite its few disadvantages, the microprocessor remains an interesting, viable alternative in digital system design.

1.3 Microprocessor Education

Before the microprocessor can be used effectively as a design tool, the design engineer must thoroughly know its capabilities,

limitations, support requirements (both hardware and software) and interface requirements. It is precisely this knowledge that is sorely needed in industry today.

A recognized deficiency in the educational program of most digital design engineers is a lack of understanding the interface problem. Since an interface refers to the electrical and logical communication facility between two systems, interface design is a problem routinely encountered in the connection of peripheral devices to any computer system.

Despite the many advances made in computer technology over the past 25 years, the input/output (I/O) problem remains a major stumbling block to faster and cheaper systems. This is particularly true for microprocessor systems which are relatively cheap and fast when compared to expensive and slow peripheral devices.

1.4 Project Initiation

In recognition of the microprocessor's impact on digital design, a proposal was originated in December 1974 by Drs. K. L. Doty and J. K. Watson to create a digital systems and microprocessor laboratory at the University of Florida. Approved and funded by the National Science Foundation, this project allowed the Department of Electrical Engineering to establish a capability to train future engineers on the latest developments in digital systems design and microprocessor technology.

Specifically, the microprocessor laboratory has the following major objectives:

1) Provide students with meaningful design experience using microprocessors.

2) Provide a means of gaining practical experience in computer-peripheral interfacing.

3) Provide a means to develop software supporting microprocessor applications.

1.5 Microprocessor Laboratory

Figure 1-1 depicts the concept of the University of Florida's microprocessor laboratory in block diagram form.

Each microprocessor station, of which there could be as many as 15, will have the capability of operating in a stand alone mode or a bus transfer mode. In the stand alone mode, each station is totally independent and self-sufficient providing a student with the capability of writing, editing, and debugging his programs or exercising various peripheral devices under program control. In the bus transfer mode, any station interfaced to the laboratory bus may request and gain control of the bus to initiate a data transfer operation to any other station on the bus, including the Data General NOVA 800 minicomputer. The NOVA, with its 48K bytes of main storage and eight megabytes of disk store, will be used to cross-assemble microprocessor code and provide debugging/diagnostic aids for the microprocessor stations. In addition, the NOVA computer will be used to satisfy any mass storage and file requirements peculiar to the microprocessor laboratory. Finally, any requirement for hard copy printout of work done in the microprocessor laboratory will be routed through the NOVA to the Northeast Regional Data Center (NERDC)

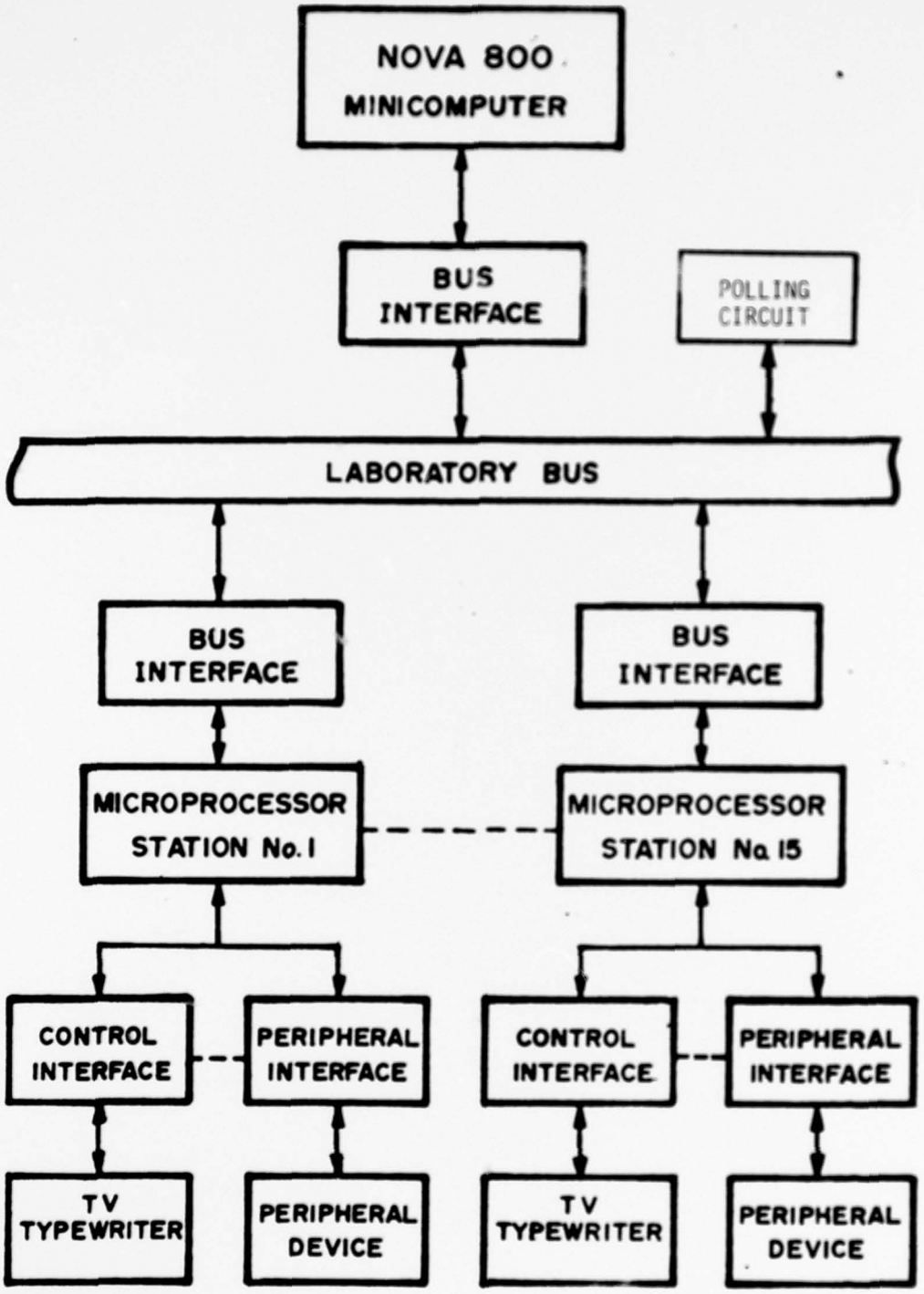


FIGURE 1-1
MICROPROCESSOR LABORATORY BLOCK DIAGRAM

IBM 370 computer which will then route the job to a remote printer.

It should be noted that a resident editor/assembler package can be added (via tape cassette) at a later date to eliminate having to use the NOVA for cross-assembly functions, freeing it for more productive work.

1.6 Chapter Summary

This chapter has introduced the microprocessor and the need to develop adequate training facilities to permit meaningful microprocessor hardware and software design experience. The reader will now be provided with information on digital data communications circuits pertinent to the microprocessor laboratory design.

Chapter II

Digital Communications Circuits

2.0 General

This chapter, intended primarily as a tutorial paper, provides sufficient background information on digital data communications circuits to permit the reader to understand the rationale behind various design decisions that are discussed in subsequent chapters.

Readers who have an understanding of digital communications circuits may wish to skip this chapter.

2.1 Concept of the Link

The concept of the link provides a convenient method of understanding the bus. A link according to Bell and Newell (1), provides a means of transmitting data (information unit) from one point in space to another point in space at the same point in time (disregarding the propagation time). There is no memory capability provided thus any information present on the link must be captured and used immediately or stored in a memory device else it will be lost.

Initiation of the transmission may be fixed at one end or the other, or be from either end depending on the design of the link. A simple link permits transmission in one direction only (from input component to output component); this is normally called a simplex link. A compound link is made up of several links, but such that no switching occurs. A half-duplex link permits information to flow

from either end to the other, but transmission is possible in only one direction at a time. A full-duplex link permits simultaneous transmission in both directions. Broadcast links permit transmission to many receivers. Although links may be configured in many different ways, the basic idea of the link is simply a wire that presents at the output end what is present at the input end.

If a single link is thought of as a scalar quantity, then a collection of links might be considered a vector quantity. This set of links, referred to as a bus, then provides a path for data transfer, control and communication between modules or components within a system.

2.2 Transmission Line Considerations

For most purposes, any link longer than two feet should be treated as a transmission line (6). It then becomes necessary to consider line drivers and line receivers. These interface circuits are devices which convert TTL signals into signals adapted for transmission lines (line drivers), and devices to convert transmission line signals back into TTL signals (line receivers). Regular TTL devices have only limited drive capability, and line driver circuits are needed to provide the transmission line interfacing.

TTL can drive ordinary interconnections (lines) as long as the interconnect time delays are much less than the TTL rise and fall time. This will reduce the current required to produce a given voltage on the line, and generally cause the transmission line to act more like a simple interconnection. This tradeoff of line delay versus driver rise/fall time will allow TTL parts to use lower power

output stages. This tradeoff will restrict normal TTL to a maximum interconnect distance of approximately two feet, which is adequate for device-to-device connections on a printed circuit card or wire-wrap board. Interconnections much longer than this will result in more pronounced transmission line effects, and generally will degrade system noise immunity and performance.

If long lines, large amounts of external noise, and a bus organized circuit are all required or present simultaneously, then the lack of drive and relatively small noise margins of regular TTL is severely felt. Interconnections longer than two feet, therefore, should be treated as a transmission line and line drivers/line receivers should be used.

2.3 Line Driver/Receiver Considerations

There are a number of line drivers/receivers available which vary considerably in configuration, function, complexity and cost.

Selecting the proper interface device for the application requires three basic decisions - standardized or special interface, single ended or balanced differential form, simplex or multiplex mode.

2.4 Standardized Interfaces

A standardized interface includes the following:

a) EIA RS 232-C

b) MIL STD 188C

c) IBM 360 I/O

A complete discussion of each of these is beyond the scope of this paper. Readers interested in information on the Electronic

Industries Association (EIA) RS232C interface standard should consult reference 4. Information on MILITARY STANDARD (MIL STD) 188C is contained in reference 23. IBM's 360 interface standard is contained in reference 11.

2.5 Special Interfaces

The first decision necessary for a non-standardizes interface involves form of operation. There are two basic forms of data communications circuits, single-ended and balanced differential.

The advantage of the single-ended system is simplicity with only one signal wire required per circuit as represented in Figure 2-1. The disadvantage is susceptibility to induced noise Vn and ground shift noise Vgs. Induced noise is caused by magnetic and capacitive coupling from adjacent signal lines or other noise generators. Ground shift noise is the result of the voltage potential developed across a finite resistance and inductance due to flow of current through the ground. The net effect of Vn and Vgs alters the voltage at the receiver input. Vn and Vgs are added to any signal produced by the driver, and a receiver cannot discriminate between a legitimate signal and a signal which is the sum of the noise in the circuit and the actual signal produced by the driver. The noise immunity of single-ended circuits, however, can be improved with three methods:

1) Use shielded cable and reduce ground impedance.

2) Increase the induced driver output voltage levels. This swamps out noise but increases power consumption.

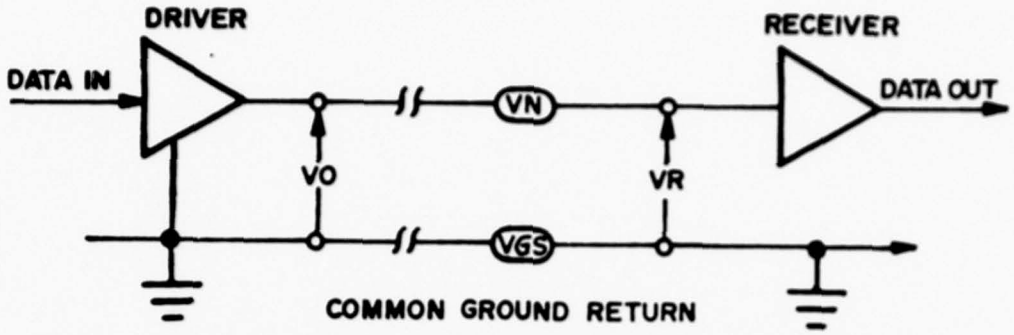


FIGURE 2-1
SINGLE-ENDED CIRCUIT

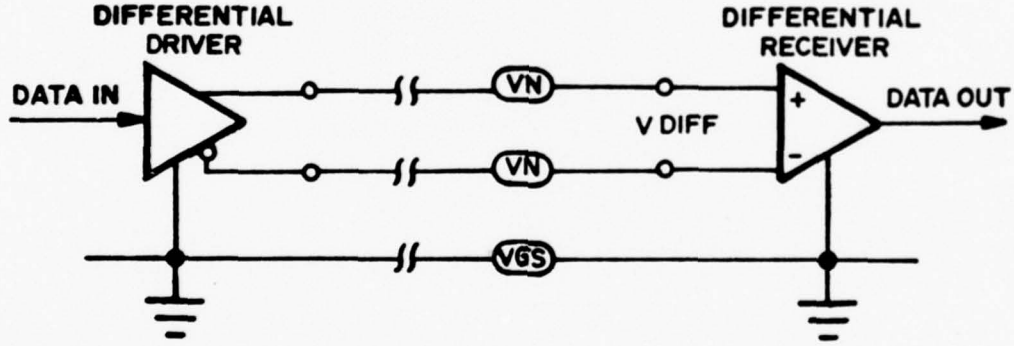


FIGURE 2-2
BALANCED DIFFERENTIAL CIRCUIT

3) Add hysteresis in the receiver which increases the d.c. noise margin but may introduce time distortion.

2.6 Balanced Differential Circuit

As illustrated in Figure 2-2, the balanced differential circuit uses a twisted pair of wires as a transmission line, a differential driver, and a differential receiver.

The twisted pair transmission line cancels magnetically induced currents in the line because of adjacent twists of the line. Electrostatically coupled noise equally affects both lines of the twisted pair. Thus, it is transformed into a common mode signal at the receiver. The ground shift noise also appears to the receiver as a common mode voltage. Because the signal voltage Vdiff and the noise voltage Vn + Vgs appear to the receiver as differential and common mode voltages, respectively, they can be separated by a receiver with high common mode rejection. In this manner, information can be transferred through environments which would otherwise cause errors in a single-ended system (6).

2.7 Modes of Operation

There are two basic modes of operation for data communications circuits - simplex and multiplex. A simplex circuit allows one way, nonreversible data flow. A multiplex circuit allows bidirectional (half-duplex) or multidirectional (data bus or party line), non-simultaneous data flow. A general multiplex circuit has two or more pairs of drivers and receivers on the same line, but only one driver may be transmitting at any one time. A special multiplex

mode is a distribution system that has one driver and two or more receivers placed at various locations on the line.

Some basic multiplex modes of operation for data communication circuits are illustrated in Figures 2-3, 2-4, and 2-5.

Operating mode selection depends mainly on the system requirements. As a general rule, simplex systems are easier to implement because timing problems are minimal. Multiplex systems conserve overall system wire costs but are more difficult to design.

2.8 Multiplex Considerations

Some of the multiplex mode (data bus) operational methods and problems that must be considered are as follows:

1) The protocol or "handshaking" required for a particular port on the bus to send data must be designed. The protocol sequence must usually provide the following operations:

a) The port must signal a desire to use the bus (interrupt).

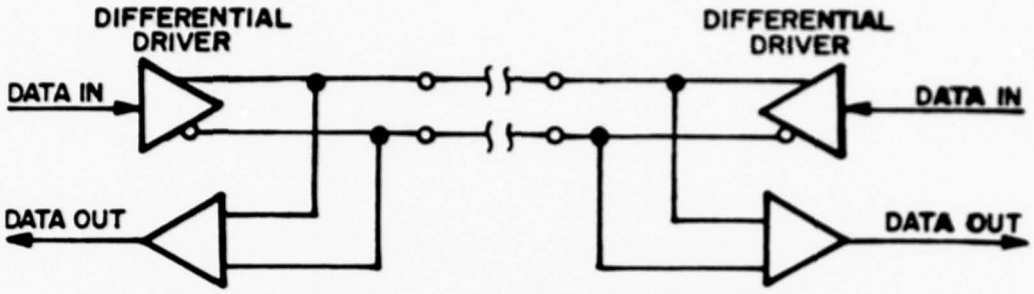
b) Bus controller must acknowledge interrupt and send "go ahead" command.

c) Port assumes control of bus and sends data, perhaps preceded by the code to indicate the recipient(s) of the data.

d) Receiving port(s) must acknowledge receipt of data.

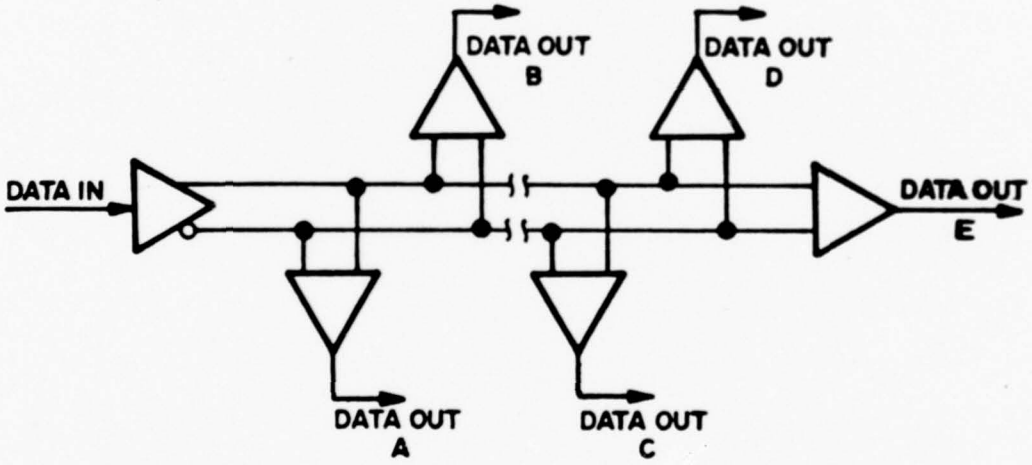
e) Transmitting port receives the acknowledgement and releases control of the bus so that other ports may pass their data.

The overall bus operation is either polled or asynchronous. In polled operation, a central bus controller addresses each port in turn to ask if any data is waiting to be sent. If the addressed



HALF DUPLEX MODE BALANCED DIFFERENTIAL

FIGURE 2-3



BALANCED DIFFERENTIAL DISTRIBUTION BUS

FIGURE 2-4

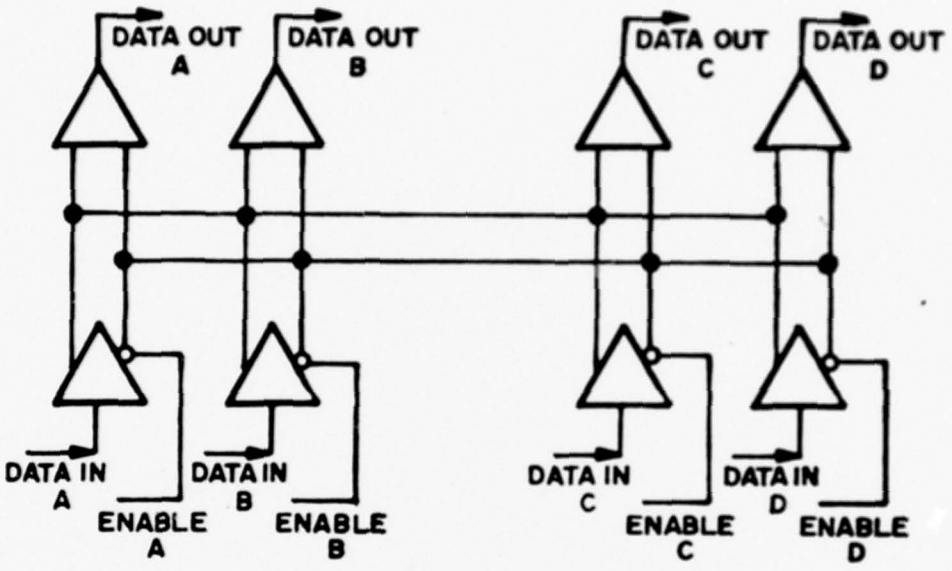


FIGURE 2-5
BALANCED DIFFERENTIAL DATA BUS

port has no traffic, it signals "no data" and the controller inquires at the next port. If the port has some data, then the controller gives the "go ahead" to the port; data is sent, and the controller then inquires at the next port.

In asynchronous operation, any port having data essentially "holds up its hand" and waits for a "go ahead" signal to ripple down the series enabling logic. The priority for a particular port is determined by the port's proximity to the master control port which sends a "go ahead" down the enable logic chain at regular time intervals. This scheme is well suited to bus-organized minicomputers. The Digital Equipment Corporation's UNIBUS and OMNIBUS architectures are excellent examples (3).

2) The effect of powered-down drivers and receivers to normal bus operation must also be considered. Integrated circuit drivers and receivers contain parasitic diodes that are normally reverse-biased when the power supply is on. Unless special design techniques are used, these diodes can become forward-biased when the unit is powered-down, thus causing the bus to malfunction.

3) The protocol timing must include sufficient time delays to allow for the different port-to-port signal propagation delays.

4) Both physical ends of the transmission line must be terminated to prevent spurious signal levels due to reflections. More will be said about this in section 2.9 on line matching.

5) Stubs or taps from the main transmission line should be kept to a minimum length. If stubs must be used, then to cause the least perturbations on the line, the stub length should be

controlled such that the propagation delay of the stub is less than 1/4 of the signal rise or fall time at the stub to line connection point (6).

6) If a 3-state driver system is used (logic zero, logic one and "off" or high impedance state), some means must be provided to detect the difference between a driver sending data and all drivers off condition. In the 2-state bus system, this problem does not occur because a logic zero (usually a high) indicates either a logic zero or that no port is currently sending. A logic one (usually a low) on a 2-state system then indicates a port is transmitting a logic one and the receiver should interpret it as such.

7) The data format (parallel or serial) must also be considered. Parallel operation is fast but expensive, since it requires one transmission line and the associated interface for every bit of the word that is transmitted in parallel. Serial operation is slower, but requires only one transmission line and interface per port. This saving, may however, be offset by the need for a parallel-to-serial converter at the transmitting end, and a serial-to-parallel converter at the receiving end.

The parallel structure is commonly used for rapid exchange of data over short distances; e.g., within a computer or between computer and peripherals. The serial structure is used for communications over long distances, as between a terminal and its controller.

8) A final consideration concerns polled operation of a multiplex system. The amount of time necessary to address and receive acknowledgement from a port must be weighed against the volume of

data the ports normally send, and the total number of ports on the bus. If there is a large number of ports on the bus, most of the time might be used by the polling operation with very little time devoted to actual exchange of information. A large number of ports combined with a high relative volume of traffic expected per port can lead to data backing up at each port waiting to be sent, and an overall reduction in data throughput. In a "real time" system where fast response is essential, serious consideration must be given to splitting up a single large bus into several "satellite" busses, each with its own polling controller and protocol with respect to the central bus. Queing theory can be used to estimate the throughput on a bus structure when many variables including the number of ports, the mean transaction length, and the number of transactions per port per unit time are known.

2.9 Line Matching

The purpose of line matching is to reduce or eliminate errors caused by transmission line reflections in data communications circuits. When system operation is not affected, these reflections can be ignored, similar to ignoring contact bounce in a mechanical switch. When data bit duration is long relative to line propagation delay, reflection effects die away in a relatively short time. However, in most transmission line circuits, reflections do affect the system and thus steps must be taken to minimize reflection effects so as to reduce or eliminate errors in data transmission. There are three basic methods of achieving this objective (6).

2.9.1 Controlling Rise Time/Line Delay

Controlling the ratio of signal rise time to line delay is one method of minimizing line reflection effects. Since most integrated circuit logic has very low output impedance and high input impedance relative to the characteristic impedance of normal lines used (50-200 ohms), this combination will likely produce large signal overshoot and undershoot at the receiver end of the line assuming that the signal rise times produced by the driver are short relative to the time delay of the line (6). If, however, the rise time of the signal is greater than twice the one way time delay of the line, a different result is obtained with the ringing amplitude at the receiver less than that predicted by classical transmission line theory. When the driver output signal rise time is four times as great as the time delay of the line, the ringing is considerably reduced and the line input and output waveforms are almost identical, with only a few small perturbations.

To make an unterminated line driven by a voltage source type driver behave properly, one of two things can be done:

1) Reduce the maximum allowable length of the line so its time delay is less than one quarter the rise or fall time of the driver.

2) Control the slew rate of the driver so the rise time at the line input is greater than four times the time delay of the line.

Both line length limiting and explicit driver slew rate control are sometimes applied by such standard interfaces as

MIL STD 188C or EIA RS 232-C.

Controlling the ratio of signal rise time to line propagation delay has a serious drawback in that delay times through a circuit are always greater than those obtained using a terminated line with a fast rise and fall time signal. This is due to the delay of the signal to rise above the receiver threshold level.

2.9.2 Parallel Termination

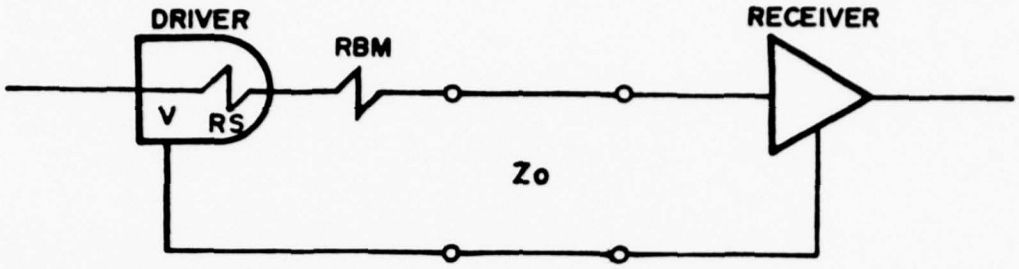
Parallel termination is another method used to reduce transmission line reflection effects. With the parallel termination method, multiple receivers can be used on the line forming a distribution bus (one driver, many receivers). After a delay time for the signal to propagate from the driver end to the receiver end, the signal is absorbed by the termination on the line. From classical transmission line theory, we know that any line terminated in its characteristic impedance will exhibit no reflections on the line. Thus a signal arriving at the end of a properly terminated line is totally absorbed and the line assumes steady-state conditions. The primary disadvantage of the parallel termination method is the power dissipated in the termination resistor since it provides a d.c. load on the driver.

The choice and placement of the parallel termination network depends on the type of driver (voltage or current source) and whether the system is simplex or multiplex. For simplex mode with voltage source drivers, a single resistor is placed across the end of the line most distant from the driver. This applies to both single-ended and balanced differential lines. Current source

drivers require that each output have a d.c. path to the common signal return, or ground, so that a split termination is used. If simplex mode differential form is used, the parallel termination network must be two resistors, each equal to half the characteristic impedance of the line connected from each side of the line to the signal common return. For multiplex operation, both ends of the line must have parallel termination networks. This prevents false signals caused by reflections from an unterminated end from occurring when a driver along the line is enabled. Again, voltage source drivers use a single termination resistor; current source drivers require split terminations for balanced differential operation.

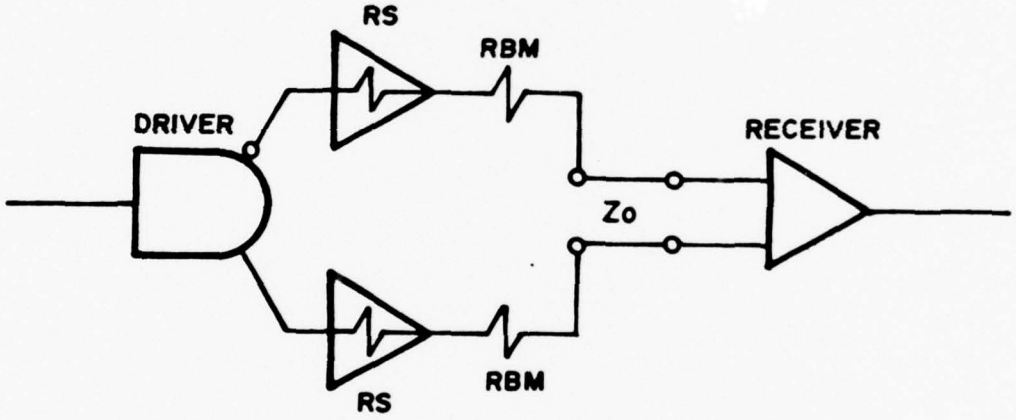
2.9.3 Back Matching

A final method of minimizing reflection effects is the series termination or back matching method. This method is useful for either single-ended or a balanced differential simplex system where only one driver and one receiver is required. A matched source is constructed by inserting a resistor Rbm in series with the voltage source driver output of a value such that the sum of the driver output resistance plus Rbm equals the characteristic impedance of the line. For differential back matching with a differential voltage source driver, two resistors are used, one in series with each output. The value of Rbm is such that the driver output impedance, plus Rbm, equals half the characteristic impedance of the line. Figure 2-6 provides a depiction of simplex back matched driver systems.



RS + RBM = Zo

SINGLE ENDED



RS + RBM = Zo/2

BALANCED DIFFERENTIAL

FIGURE 2-6

SIMPLEX BACKMATCHED DRIVER SYSTEM

The back matched driver method is based on two points of transmission line theory.

1) An unterminated line approximately doubles the signal level at the receiver site.

2) When driver source impedance equals the line impedance, the line appears terminated with respect to signals returning to the source.

It should be pointed out that the back matched driver method is limited to simplex operations only and only one receiver located at the end of the line opposite the driver end of the line. This limitation results from the fact that any additional receivers placed along the line do not receive a full signal swing until the voltage wave from the load end returns to their bridging points. As a result, these additional receivers have severely reduced noise immunity and may falsely indicate received data.

2.10 Long Transmission Line Effects

A fast signal transition on a long transmission line becomes more rounded and exponential as the signal propagates down the line. Additionally, the resistance of the wires comprising the line causes an overall reduction in signal amplitude. These two effects combine to limit the data rate that can be carried by a given type and length of line. Of these two effects, the change in signal wave-shape is the most limiting to the maximum data rate. If a new data bit is sent and arrives at the line end before the line has completed its response to the previous data bit, then the signal representing the new data bit can cross the receiver's threshold

earlier than normal. This causes the recovered data from the line receiver to exhibit pattern dependent transition displacement, or time jitter. In the limit, as a shorter and shorter pulse is sent, the signal at the line end may not even have time to cross the receiver's threshold, and the data bit might be missed completely. This phenomenon is called intersymbol interference, and is due to previous data bit(s) causing a time shift of the threshold crossing of the present data bit (6). Intersymbol interference starts to occur any time the minimum time duration of a bit is less than the signal rise (or fall) time at the end of the line. Figure 2-7 shows this effect.

2.11 Selecting Line Drivers and Receivers

The tradeoffs between external noise and noise immunity, signal quality and line length, serial or parallel structure, and cost versus performance all affect the selection of which particular line drivers and receivers to use. A few additional factors follow.

1) The most limiting factor to data rate with long lines (>50 feet) is usually the rise and fall time of the cable.

2) Balanced differential forms are preferable to single ended forms where high noise environments are present.

3) The total number of ports on a multiplex system should be restricted so that the parallel combination of the input impedances of receivers and the output impedances of the disabled drivers is greater than the characteristic impedance of the line.

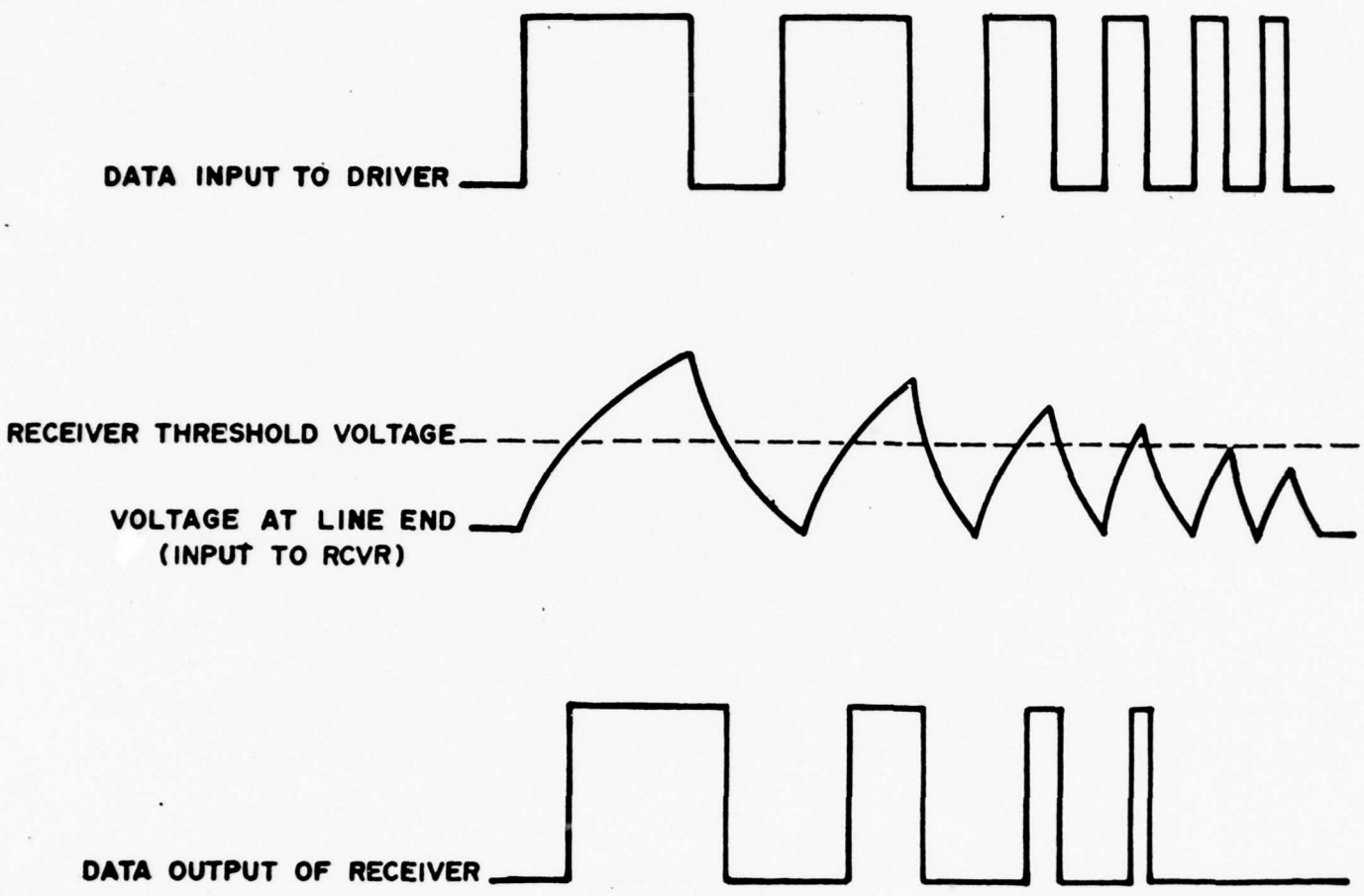


FIGURE 2-7
DISTORTION EFFECT ON DATA BANDWIDTH RESOLUTION

4) Ground returns are necessary for proper operation of integrated circuit line drivers and receivers. This may be accomplished by connecting the shield of the line to the ground pins of integrated circuits connected to the line.

5) Data rates above 10 Megabits will usually require emitter coupled logic (ECL) to be used instead of TTL types for the drivers and receivers because of the 20 to 50 nanosecond propagation delays in the TTL devices.

2.12 Chapter Summary

This chapter has presented sufficient information on digital data communications circuits to enable the reader to understand the rationale behind major design decisions discussed in subsequent chapters.

The reader is now prepared to examine several interfacing standards which are included in the next chapter.

Chapter III

Data Transmission Protocols

3.0 General

This chapter provides information on four data transmission protocols which are either approved standards or proposals. This information will provide insight into the problem of determining an acceptable protocol for the microprocessor laboratory bus which is discussed in the next chapter.

3.1 SDLC

Synchronous Data Link Control (SDLC) is an IBM protocol for bit serial synchronous transmission between buffered stations on a data transmission link using centralized control (10). Some major features of SDLC are that it is independent of code structure. It is bit oriented and does not use control characters (which sometimes have multiple meanings). It is unambiguous as to data link control functions to be performed and to the direction of transmission. It uses one standard frame format for information transfer as well as for link supervision and control. This standard is designed to achieve greater utilization of resources - terminals, communications processors, transmission links - than has been possible with previous data link controls.

SDLC employs centralized control where one station is designated the primary station. It retains control of the link at all

times and exercises control through a prescribed set of commands and responses. The primary station is also responsible for initiating error recovery procedures when it detects errors.

All other stations on the link connected to the primary station are designated as secondary stations. They receive the commands and information sent by the primary station and then react with suitable responses and their own information (if any) to be sent to the primary station.

All SDLC transmitted frames conform to a standard format. Furthermore, multiple functions are handled within a single frame. For example, an information frame sent by the primary station to a secondary station is used to transfer information. The same information frame can be used to acknowledge to the secondary station, using a receive count, that one or more frames previously sent have been received and accepted as correct. On receiving a valid error-free acknowledgement, the secondary station releases (clears) an appropriate number of frame buffers, readying the buffers for new inputs. Meanwhile, the same information frame from the primary station can poll the secondary station to find out if it has any frames to transmit (13).

As mentioned, frames can contain counts of received frames and transmitted frames that are independent of each other. Independent frame sequence numbering compensates for a variety of information flow conditions that can occur during data link operation. For example, SDLC can accommodate unbalanced information flow, unequal information frame length, and variability in the amount of information to be transmitted (13).

Unbalanced information flow means that more information (volume) flows in one direction than in the other, e.g., remote job entry (RJE) applications. Unequal information frame length means there will be short frames in one direction and long frames in the other direction, e.g., inquiry/response systems. Variability in the amount of information can mean that there is wide variation in the number of frames in quque at different secondary stations, e.g., data requirements vary with working hours.

As shown in Figure 3-1, a frame starts with an 8-bit flag field followed in position by an address field, a control field, an information field (if present), and ending with another flag field.

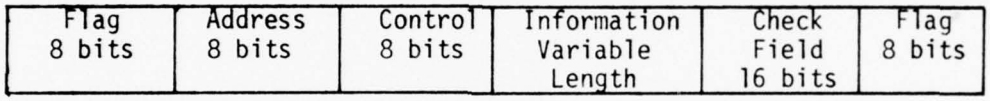


Figure 3-1

SDLC Frame Format

The purpose of the flag field is to denote the start and end of a frame. The flag acts as a delimiter between frames. A frame starts and ends with an 8-bit flag sequence 01111110.

The address field designates which secondary station is to receive the balance of the transmitted frame. When a secondary station transmits, the address tells the primary station which secondary station originated the frame. Also, the primary station will accept a frame only when it contains the address of a secondary station that has been given permission to transmit. For data integrity reasons, the address appears in each frame.

The 8-bit control field is the kernel of SDLC and will be described in more detail following a brief discussion of the information and frame check fields.

The information field may vary in length. This field may or may not be present depending on whether data or control information is being transmitted in a given frame. If data is being transmitted, it may be configured in any code structure, e.g., straight binary, BCD, packed decimal, EBCDIC, ASCII, or Baudot. It should be noted that peripheral device control characters, such as CARRIAGE RETURN, will actually be part of the information field, while the code being used (ASCII, EBCDIC, etc.) may be implied in the address of a specific terminal designed for a specific code.

The frame check sequence (FCS) field is included in all SDLC frames to detect errors which may occur during transmission. This field is 16 bits long and immediately precedes the beginning of the end-of-frame flag. The contents of the FCS field, based on a cyclic redundancy check, is intended to check all data transmitted between the start flag and the end flag. More details on this check can be found in reference 10.

The purpose of the control field is to tell (command) the addressed secondary station what operation it is to perform. The secondary station uses the control field to react (respond) to the primary station.

The control field takes on any one of three formats depending on whether the field is to indicate:

1) information transfer (see Figure 3-2)

2) supervisory commands/responses (see Figure 3-3)

3) nonsequenced commands/responses (see Figure 3-4)

The control field in its information format is used by primary and secondary stations to transfer an information field. A 0 in the 9th bit after the end of the start flag identifies the control field as being in the information transfer format. A transmitting station, whether primary or secondary, uses the Ns subfield to indicate the count sequence, or number, of the frame being sent. Thus Ns means "I am now sending you frame number N," where N can go from 0 to 7. Similarly, Nr means "I am acknowledging error-free receipt of sequences up to Nr-1 that you previously sent and I am now looking for frame number N." The use of independent frame-sequence counts provides for the detection of missed, erroneous, or duplicated frames in either direction. At the completion of a frame, the transmitting station updates its send-sequence count. The station receiving the frame compares the contents of its receive-sequence-count field (Nr) and, if they are equal, accepts the frame provided no error is detected by the frame check sequence. The receiving station's receive count (Nr) is then updated. Thus the contents of the Nr field in the received frame acknowledges to the receiving station that frames through sequence number Nr-1 have been accepted and corresponding buffers may be cleared. A primary station inserts a 1 in the 13th bit (P/F) position to inform (POLL) the secondary station to initiate transmission. The secondary station inserts a 1 in this position to respond to the primary station to indicate when a frame is the FINAL one in a transmission. Otherwise, the 13th bit stays 0.

INFORMATION FORMAT

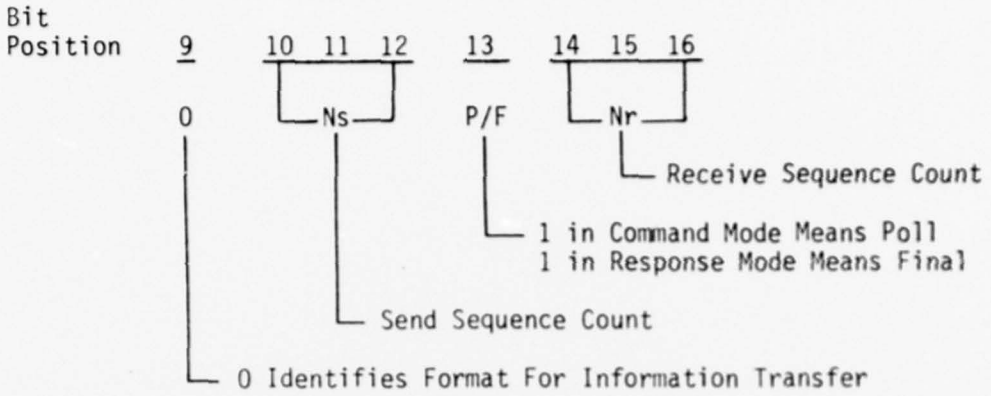


Figure 3-2

Information Format Of SDLC Control Field

SUPERVISORY FORMAT

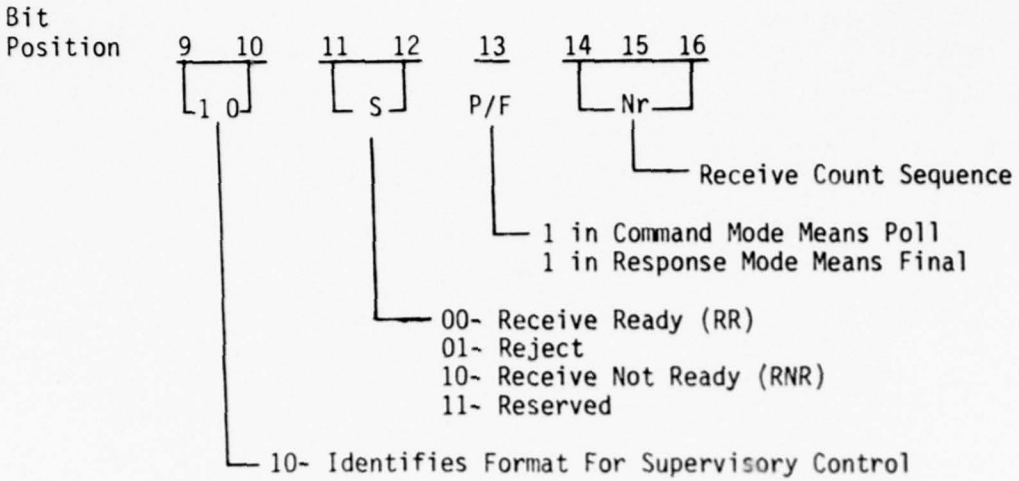


Figure 3-3

Supervisory Format Of SDLC Control Field

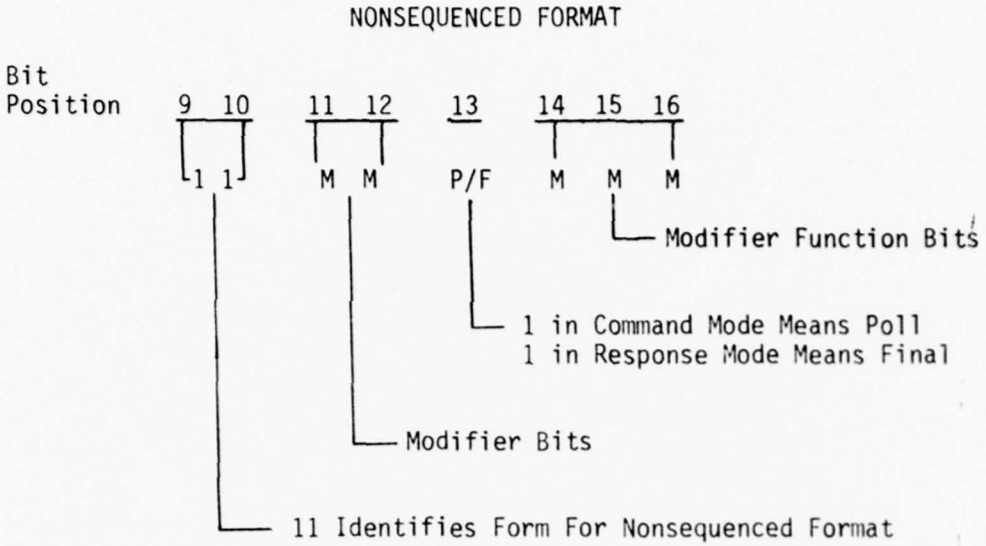


Figure 3-4

Nonsequenced Format Of SDLC Control Field

The control field in its supervisory format, Figure 3-3, is used for acknowledging frames and requesting retransmission. It is identified in the control field by a binary 10 pattern in the 9th and 10th bits after the start FLAG. The supervisory functions accomplished depend upon the supervisory (S) subfield bit patterns as indicated in Figure 3-3. A frame containing a supervisory field never contains an information field. The supervisory control format regulates the flow of information frames by invoking such controls as RECEIVE READY and RECEIVE NOT READY.

The control field in its nonsequenced format, Figure 3-4, can provide up to 32 functions without changing frame count sequence. Examples of nonsequenced control functions are the polling of secondary stations without affecting sequence numbers, broadcasting, exchanging terminal identification information between primary and secondary stations, and commanding a secondary station to go "on hook" (primarily on dial-up links). Typical nonsequenced response functions are to reject invalid commands and to perform initialization and diagnostic procedures pertaining to data link control.

3.2 HDLC

High Level Data Link Control (HDLC) is very similar to SDLC and is an interface standard being proposed by the American National Standards Institute (ANSI) to the International Organization for Standards. This protocol defines in detail the structure for communication systems using bit-oriented high-level data link procedures (12).

In HDLC procedures, data communication between two data stations consists of the transfer of frames containing information from the data source to the data sink, and acknowledged by a frame in the opposite direction. Until the Data Terminal Equipment (DTE) comprising the data source receives the reply, it must hold the original information in memory in case the need should arise for retransmission.

A data link involves two or more participating stations. For control purposes, one station on the link must assume responsibility for the organization of data flow and for link level error recovery operations. The station assuming these responsibilities is known as the primary and the frames it transmits are referred to as command frames. The other stations on the link are known as secondaries and frames they transmit are referred to as response frames (12).

Figure 3-5 illustrates the frame format specified in the standard.

For the transfer of data, the following two cases of data link control are illustrated in Figure 3-6 and discussed below.

In the first case, the DTE comprising the data source performs a primary data link control function and controls the DTE comprising the data sink that is associated with a secondary data link control function by select commands.

In the second case, the DTE comprising the data sink performs a primary data link control function and controls the DTE comprising the data source that is associated with a secondary data link control function by poll commands.

The control of traffic between the data source and the data sink is effected by means of a numbering scheme which is cyclic

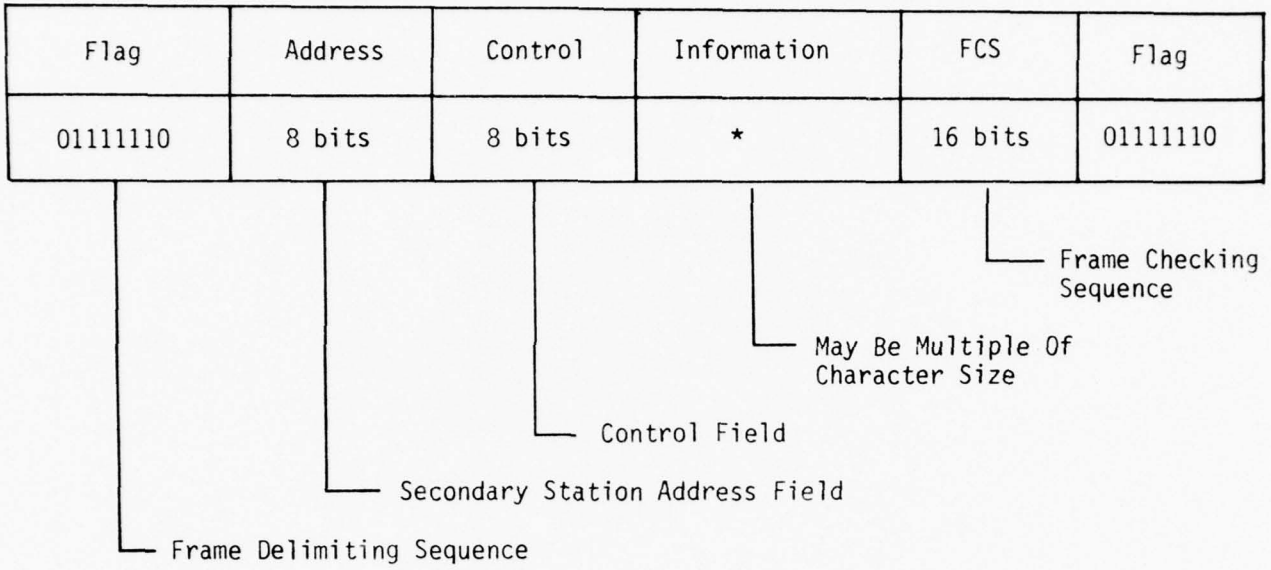


Figure 3-5

HDLC Frame Structure

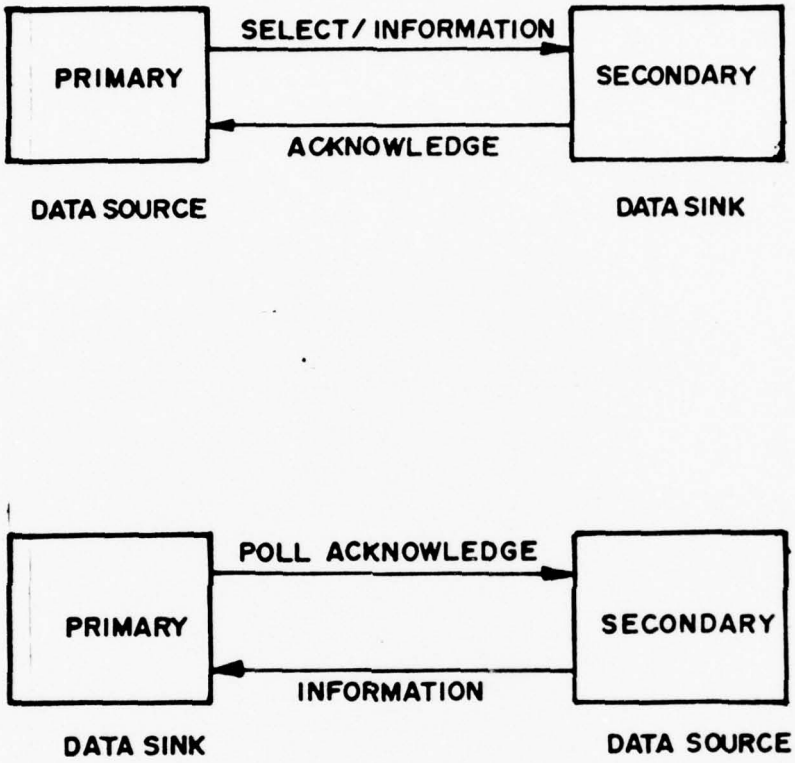


FIGURE 3-6

DATA LINK CONTROL

within a modulus specified in the standard. An independent numbering scheme is used for each data source/sink combination on the link (12).

The acknowledgement function is accomplished by the data sink informing the data source of the next expected sequence number. This can be done in a separate frame not containing information or within the control field of a frame containing information (12).

Only the fundamentals of the HDLC protocol have been covered here. Included in the standard is information regarding operational modes, control field formats and parameters, commands and responses, and functions of the poll/final bit. For details on this proposed standard the interested reader should consult reference .

3.3 CAMAC

Computer Automated Measurement and Control (CAMAC) has emerged as a modular instrumentation and digital interfacing standard (9). CAMAC systems or components are being used in the laboratories of the Energy Research and Development Administration (ERDA), formerly the Atomic Energy Commission. Although the majority of the present CAMAC systems are used almost exclusively for data gathering, there is now wide-spread interest in employing CAMAC for process control as well as computer interfacing.

Figure 3-7 reflects the standardized CAMAC bus structure allowing the use of computer-independent modules to interface to the real world.

The basis of the CAMAC system is the "crate" that houses the interfacing modules. At the rear of the crate is a dataway that

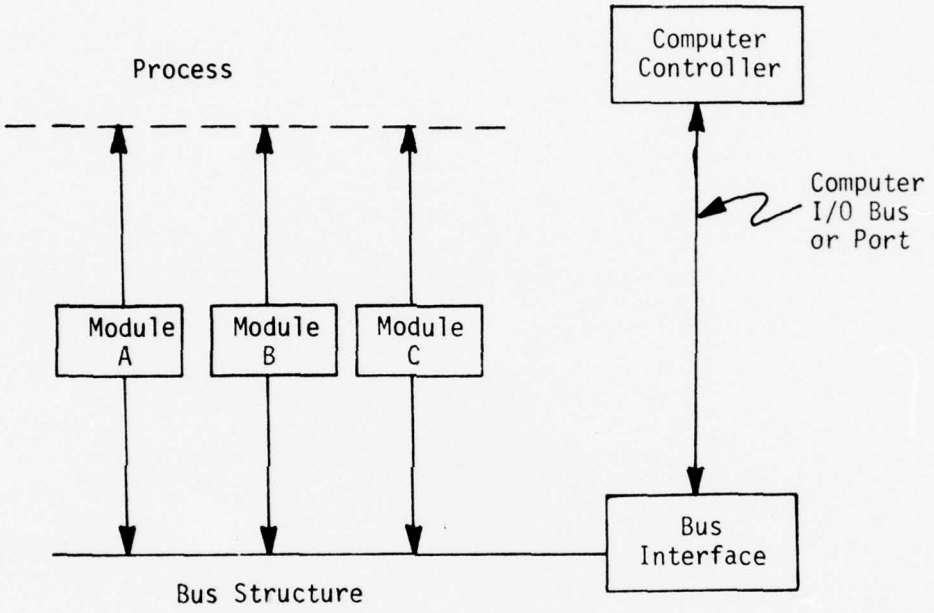


Figure 3-7

CAMAC Standardized Bus Structure

provides a digital pathway between the modules. Each crate can receive up to 25 modules. What is inside the modules is not specified in any way; only the interface to the dataway is standardized. The single interface to the external computer (or other crates) is made via a module inserted at the right side of the crate. This module is called the crate controller. If a microprocessor is placed in the crate controller position, then there is no need for an interface through the controller since the microprocessor provides the intelligence in the crate controller itself (9). IEEE Standard 583-1975 provides the mechanical and electrical standards for the CAMAC system (22). Also, IEEE Standard 596 specifies how up to seven crates can be interconnected for high speed application (a parallel "highway") and IEEE Standard 595 (serial highway) provides the standards for slower-rate applications requiring large-distance bit or byte serial data transfers between up to 62 crates.

Figure 3-8 shows the basic dataway interface to a CAMAC module. The command lines include five function code and four subaddress lines. The five function lines are coded into 32 function codes, of which about half are defined to achieve operational compatibility between modules. The four subaddress lines are used to subdivide the module into 16 entities, referred to as "registers."

Commands are performed only by those modules that are addressed by the N line. There are also unaddressed commands which apply to all modules, e.g., initialize (Z), clear (C), inhibit (I) and busy (B). Busy indicates that a dataway cycle is in progress or an unaddressed command is being sent.

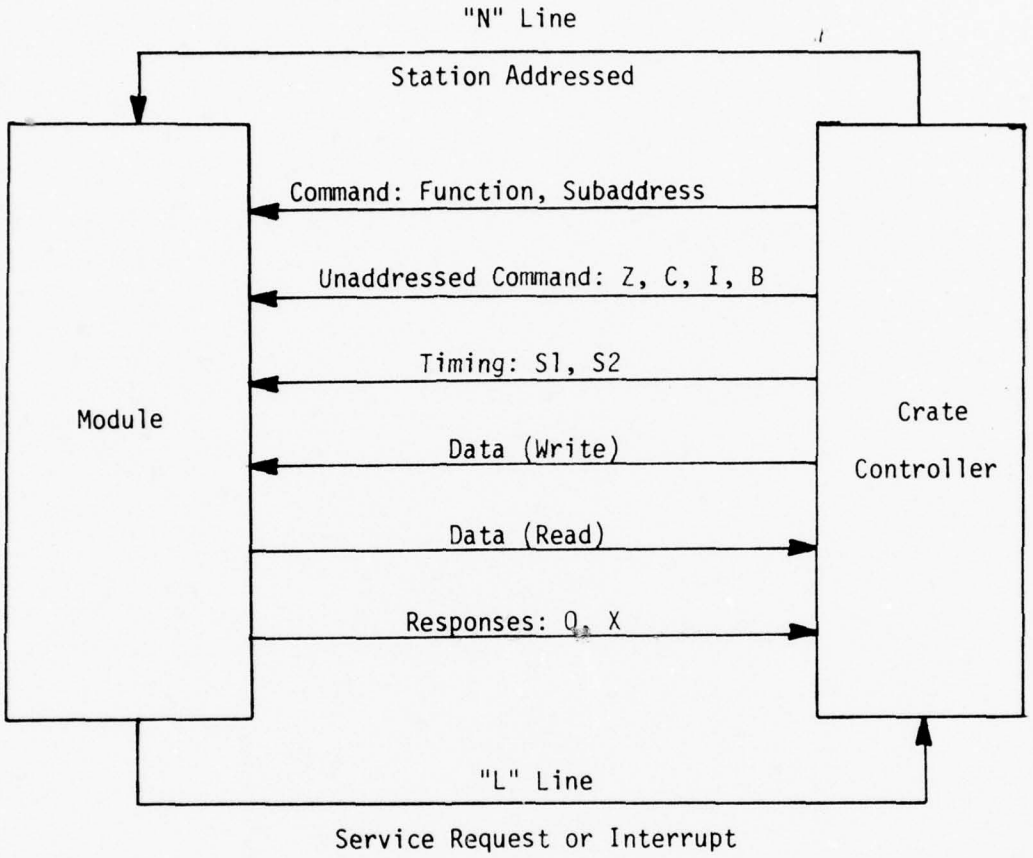


Figure 3-8

CAMAC Dataway Interface

A dataway cycle includes two timing signals, S1 and S2, usually generated by the crate controller. S1 is used as a strobe to accept data from the dataway; S2 is used for initiating actions that might change the state of the dataway read or write lines. Thus the CAMAC cycle is a synchronous two-phase timing sequence where data are read or loaded (transferred) on S1 and changed (cleared or sequenced) on S2. There is no standard cycle time, but a minimum of 1 microsecond is specified and all modules must meet this specification.

All data are transferred within a crate in parallel format with 24 read lines and 24 write lines. Thus data transfers of up to 24 bits wide may be handled.

The terms read and write only imply direction of transfer and the data may actually be control information. For example, F(16) (Write Group 1 Register) may be used to send 24 bits of individual on-off information to a register module, from which the signals are sent to the process under control (22).

There are two response bits that are returned from the module to the crate controller. The X response line indicates that its particular function code and subaddress are present, i.e., that a module is present in the designated station. The Q response bit is a generalized response signal to be used as the designer sees fit. For example, the Q bit can be used as a control line for block transfers.

The L line is the source of an interrupt request from a module. Usually many different interrupt requests may be generated within a module and all are OR tied onto the single L line. The

standard provides guidelines for the control and rapid identification of these multiple interrupts.

The N line, when active, indicates to the respective module that the command on the command bus pertains to that module. Since the modules are gated independently by the N lines, the entire crate acts as a multiplexer when data is transferred from the module to the dataway. With data transfer in the opposite direction (dataway to module) the crate acts as a distributor.

3.4 GPIB

The General Purpose Interface Bus, proposed by Hewlett-Packard, is an interface standard designed primarily for instrumentation applications in much the same manner as the CAMAC standard.

Figure 3-9 depicts the GPIB interface standard.

The basic interface structure consists of 16 lines connecting a number of instruments. At any given time, any particular instrument connected to the bus may be either idle or functioning as a talker, listener or controller. As a talker, it sends data over the bus to a listener or listeners. As a listener, it receives such data. As a controller, it directs the flow of data on the bus by designating which instruments are to send data and which are to receive data (14).

The bus itself is grouped functionally into three component buses. The data bus (8 lines) is used to transfer data in bit parallel, byte serial form from talkers to listeners. The transfer bus (three lines) is used for the handshaking process by which a talker or controller can synchronize its readiness to transmit data

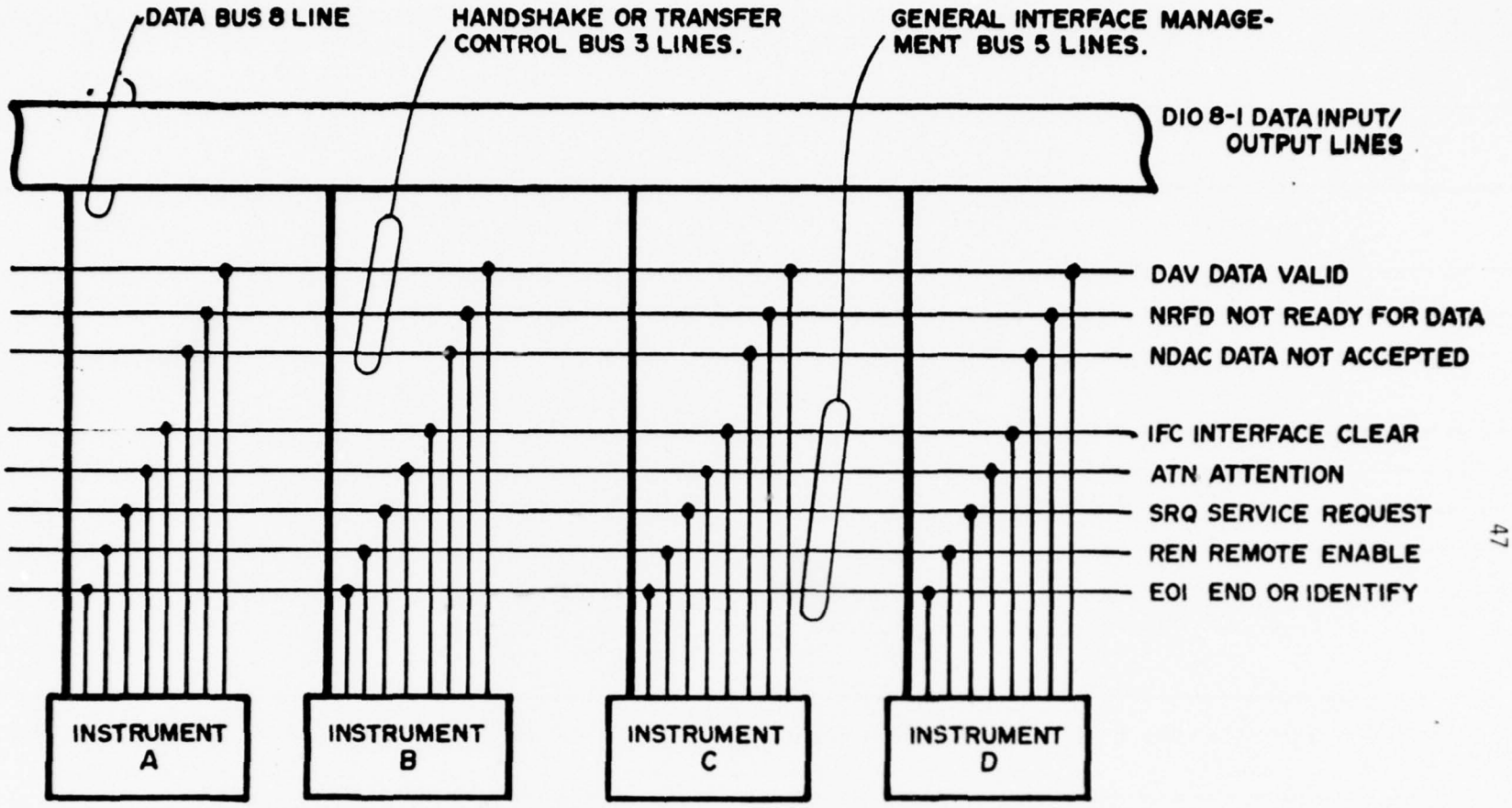


FIGURE 3-9
GPIB INTERFACE STANDARD

with the listener's readiness to receive data. The general interface management bus is principally used by the controller. It should be noted that the bus is entirely passive. All of the active circuitry that allows an instrument to talk or listen or control is contained within the instrument. Circuitry and bus together make up the interface.

The controller governs the operation of the bus. It uses a group of commands, referred to as interface messages, to direct the other instruments on the bus in carrying out their functions.

The commands serve several different purposes and are shown in Table 3-1 along with their code form and meaning.

Table 3-1

Command and Address Form

Code Form	Meaning
X 0 0 A5 A4 A3 A2 A1	Universal Commands
X 0 1 A5 A4 A3 A2 A1	Listen Address
X 0 1 1 1 1 1 1	Unlisten Command
X 1 0 A5 A4 A3 A2 A1	Talk Address
X 1 0 1 1 1 1 1	Untalk Command
X 1 1 A5 A4 A3 A2 A1	Secondary Commands
X 1 1 1 1 1 1 1	Ignored

The controller has two ways of sending interface messages. Multiline messages are sent over the data lines and the three transfer-bus lines. Uniline messages are transferred over the five individual lines of the management bus.

The commands serve several different purposes. Addresses or talk and listen commands select the instruments that will transmit and accept data. Universal commands cause every instrument equipped to do so to perform a specific interface operation. They include multiline messages and three uniline commands - interface clear (FC), remote enable (REN), and attention (ATN). Addressed commands affect only those devices that are addressed and are all multiline commands. The "un" commands disable an instrument from sending or receiving data (14).

During the configuration of a system, each device must be assigned one or more addresses unique to it. However, two listeners may have the same address if they are always to receive the same data. An instrument is assigned its address by some convenient means such as switches on the rear panel or jumper wires on a printed-circuit board. The particular value is the system designer's choice.

There are four phases to the data transfer cycle - the talker generates a new data byte, the data on the data bus's signal lines settle, the listener accepts the data, and the listener becomes ready for the next byte.

The data rate at which the bus system will operate is determined by the talker, listener and transmission system. The time taken to generate data is determined by the characteristics of the talker, and the times taken to accept data and to become ready for more depend on the listener. The data settling time is determined by the characteristics of the transmission system (drivers, receivers and terminations). By careful design and configuration, a rate of one megabyte per second can be achieved with this protocol (14).

3.5 Chapter Summary

Information contained in this chapter provides the reader with insight into the importance of bus protocols in data transmission and the variety of protocols available. It then becomes a matter of determining which of these protocols is best suited for the application or if an entirely different protocol is required. These considerations will be evaluated in the next chapter.

Chapter IV

Microprocessor Laboratory Bus

4.0 General

This chapter provides the rationale behind the various design decisions for the laboratory bus. Each of the major decisions will be discussed in detail.

4.1 Serial Versus Parallel Transmission

Perhaps the most important decision affecting overall performance of the microprocessor laboratory bus is the data format employed for data transmission - either bit serial or bit parallel. Although bit serial could be selected with resultant decrease in costs and degradation in rate of data transmission, the parallel method was chosen for several reasons as outlined below.

Microprocessors are inherently parallel devices and to take maximum advantage of microprocessor speed of operation and minimize interface requirements with input/output ports, a parallel data bus appeared more desirable from an overall performance viewpoint. This method of data transmission becomes even more essential and more realistic when considered in light of current interest in parallel processing, fault tolerant processing, and computer networking. There is a need for increased research in these areas and thus the parallel bus structure was chosen to provide the maximum capability to support this effort.

Although the parallel bus structure provides maximum performance capability for data transfer operations, the parallel method involves more circuitry and is thus more expensive to implement. Fortunately, the length of the laboratory bus did not create an excessive cost situation with respect to bus cabling requirements. Had the length of the bus been much greater, then the parallel versus serial option would require re-evaluation.

The parallel structure also requires additional line drivers/receivers for each line, but this additional expense is not as great as imagined since a serial bus would require special support circuits for parallel to serial and serial to parallel conversion.

4.2 Laboratory Bus Structure

A unified bi-directional data bus structure was chosen for the microprocessor laboratory. This organization is very similar to the Digital Equipment Corporation's UNIBUS. It is a high-speed data transmission facility. The actual bus is a matched and terminated transmission line which must be driven and received with devices designed for this application. The following sections describe bus transmission, bus signal levels and bus transmitter and receiver circuits.

4.2.1 Bus Transmission

Since the overall length of the bus approximated 50 feet, either twisted pair cable or flat ribbon cable could be used to implement the bus. Coaxial cable was ruled out completely because of its cost and the fact that the bandwidth/data rate capabilities it

provides were not required in this application.

Alpha type 6012 shielded twisted pair cable was chosen to provide maximum noise immunity, minimum crosstalk and impedance compatibility with the bus drivers selected (discussed in next section). This is a high quality data cable consisting of six pairs of #22 gauge stranded conductors, with each pair having a drain wire and aluminum foil shield.

For the laboratory bus, the twisted pairs could be used in a single ended fashion and still satisfy bus requirements since the length of the bus did not make balanced differential lines mandatory.

4.2.2 Bus Drivers/Receivers

The 8838 quad unified bus transceiver was selected for use with the laboratory bi-directional bus. This particular chip has similar specifications to that required by DEC for UNIBUS applications (3).

Each quad transceiver contains four totally separate driver/receiver pairs per package. Typical receiver input hysteresis is 1 volt with a guaranteed minimum bus noise immunity of 1.3 volts. Receiver hysteresis is independent of receiver output load. The 8838 chip also provides open collector driver outputs allowing wire-OR connections on the bus. One 2-input NOR TTL gate, provided in the 8838 chip, disables all drivers in a package simultaneously. This is a desirable feature which can be used to prevent inadvertent actions on the bus.

This driver/receiver package is designed for use in bus organized data transmission systems interconnected by 120 ohm impedance

lines. For proper termination, a 180 ohm resistor from the bus to the +5 volt logic supply together with a 390 ohm resistor from the bus to ground is required. Low bus pin currents allow up to 27 driver/receiver pairs to utilize a common bus. Also, bus loading is unchanged when power is removed. Reference 19 contains complete technical specifications on this integrated circuit.

It should be noted that the 8838 driver is logically an AND gate. Even though there is voltage inversion from input to output, there is no logical inversion. For example, if both inputs to a driver are logic 1 active high, the driver output will be logic 1 active low. In a bus configuration, any driver whose output goes active low logic 1 will cause the bus to which it is connected to be active low logic 1.

Logically, the receiver has no effect on the signal, but there is voltage inversion thereby cancelling the effect of the first voltage inversion caused by the driver. The output of the receiver is TTL level compatible

4.2.3 Bus Signal Levels

Bus signal levels are as follows:

logic 1 = 0 volts

logic 0 = +4 volts

4.3 Bus Protocols

Of those data transmission protocols discussed in Chapter 3, none ideally suited the laboratory bus application although the General Purpose Interface Bus came closest. Should there ever be a

need to connect instrumentation modules to the laboratory bus, then the overall bus philosophy should be re-evaluated with careful consideration being given to adopting GPIB as the bus standard.

The only real disadvantage in using GPIB as the laboratory bus standard is that a bus controller is required. This can be provided in one of several ways. A dedicated controller, either hard-wired or microprocessor based, may be used. Such a controller would be designated system controller and automatically assume the role of controller upon application of power. Subsequently, it can transfer this role to another device. Another approach might have one of the modules connected to the bus to assume the controller role in a dedicated manner.

A bus protocol similar to GPIB was chosen for the microprocessor laboratory as discussed below.

A 24 line bus was chosen as indicated in Table 4-1. The bus consists of eight data, four master, four slave, three handshake and one ground line. Each of their functions will be described.

Table 4-1

Laboratory Bus Organization

Pin No.	Designation	Function
1	$\overline{\text{DATA 0}}$	Data Bit 0 (LSB)
2	$\overline{\text{DATA 1}}$	Data Bit 1
3	$\overline{\text{DATA 2}}$	Data Bit 2
4	$\overline{\text{DATA 3}}$	Data Bit 3
5	$\overline{\text{DATA 4}}$	Data Bit 4
6	$\overline{\text{DATA 5}}$	Data Bit 5

Table 4-1 - continued

Pin No.	Designation	Function
7	$\overline{\text{DATA 6}}$	Data Bit 6
8	$\overline{\text{DATA 7}}$	Data Bit 7 (MSB)
9	$\overline{\text{MASTER 0}}$	Master station address bit 0 (LSB)
10	$\overline{\text{MASTER 1}}$	Master station address bit 1
11	$\overline{\text{MASTER 2}}$	Master station address bit 2
12	$\overline{\text{MASTER 3}}$	Master station address bit 3 (MSB)
13	$\overline{\text{SLAVE 0}}$	Slave station address bit 0 (LSB)
14	$\overline{\text{SLAVE 1}}$	Slave station address bit 1
15	$\overline{\text{SLAVE 2}}$	Slave station address bit 2
16	$\overline{\text{SLAVE 3}}$	Slave station address bit 3 (MSB)
17	$\overline{\text{DATA RDY}}$	Data ready
18	$\overline{\text{DATA REC}}$	Data received
19	$\overline{\text{POLL}}$	Poll
20	$\overline{\text{CLK}}$	Clock
21	$\overline{\text{BUS GRANT}}$	Bus grant to master station
22	$\overline{\text{BUS REL}}$	Bus release (polling resumes)
23	$\overline{\text{SLAVE PRES}}$	Slave station present
24	GND	Bus ground, signal common

4.3.1 DATA - These eight bi-directional data lines transfer a single byte of data from the master station to the slave station.

4.3.2 MASTER - Appearing on these four lines is the station code of the station having control of the bus.

4.3.3 SLAVE - The code of the station designated by the master as slave will appear on these four lines.

4.3.4 Handshake - There are three handshake lines used to complete a data transfer.

1) Data Ready ($\overline{\text{DATA RDY}}$) - Asserted by the master station when the byte on the DATA lines is valid and ready to be transmitted to the slave station.

2) Data Received ($\overline{\text{DATA REC}}$) - Asserted by the SLAVE station when the data byte has been received.

3) Slave Present ($\overline{\text{SLAVE PRES}}$) - Asserted by a device whenever its code appears on the SLAVE lines. This allows the MASTER station to verify that the SLAVE station is available before initiating a data transfer operation.

4.3.5 Control - There are four control lines used to determine the state of the polling logic.

1) $\overline{\text{POLL}}$ - Asserted low to indicate that the bus control logic is in the polling mode trying to find a station that wants control of the bus.

2) $\overline{\text{CLK}}$ - A clock signal appearing on the bus to indicate that the code on the MASTER lines is valid for the time the clock signal is active low.

3) $\overline{\text{GRANT}}$ - Asserted low when a station is granted bus control. The $\overline{\text{GRANT}}$ signal causes the control logic to cease the polling function.

4) $\overline{\text{REL}}$ - Asserted by the station currently in control of the bus when it has completed data transfer operations and it wishes to

release control to permit polling to be resumed.

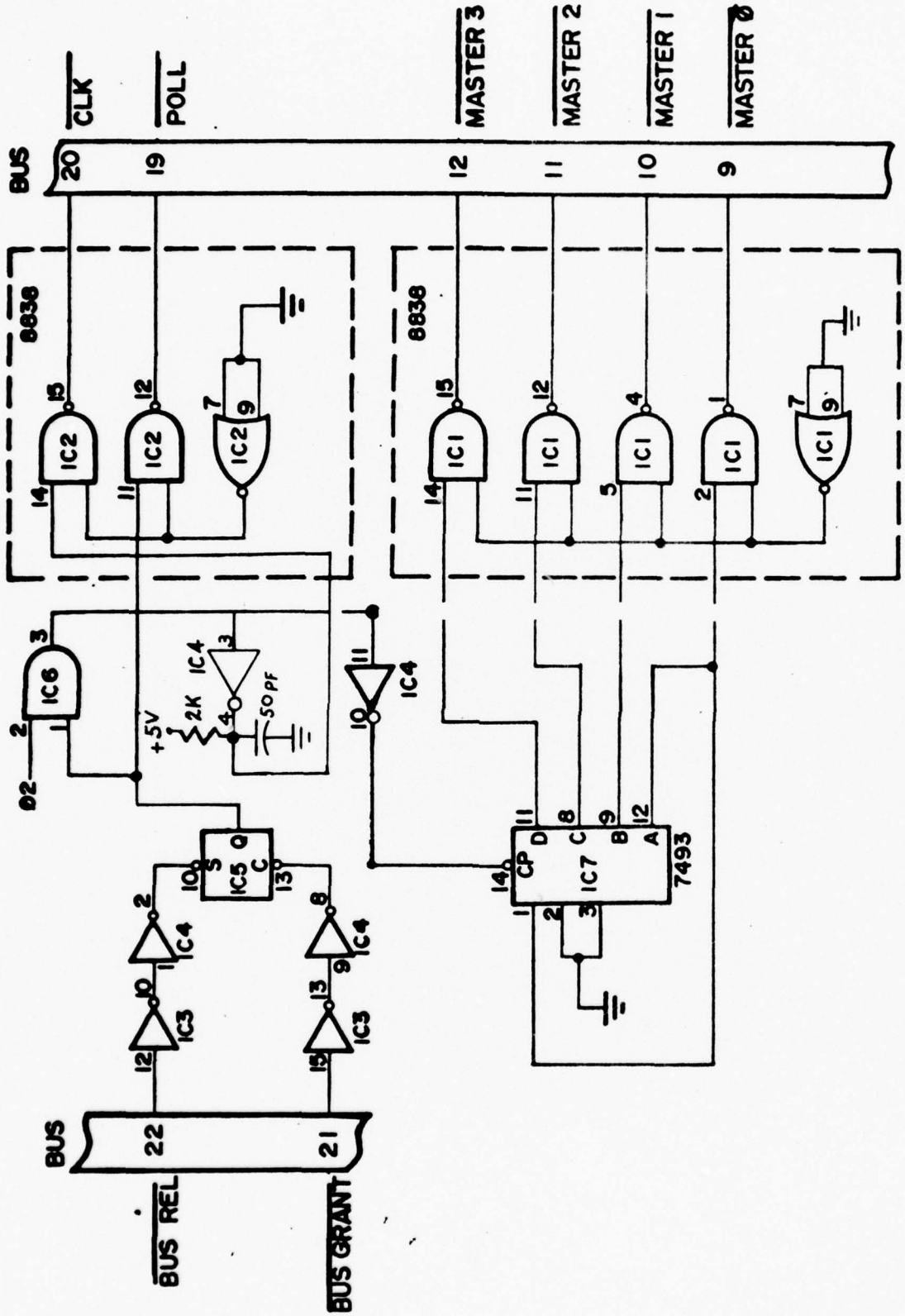
4.4 Polling

Polled operation of the bus was selected over asynchronous operation primarily because it could be realized easily and the number of stations envisioned for the bus ($\leq$16) would not involve an excessively long wait for any one station, provided polling occurred at a sufficiently fast rate. The polling circuitry is designed to insure that no station can regain bus control until all other stations are afforded an opportunity to access the bus. This prevents what is commonly called "hogging the bus."

Figure 4-1 contains the schematic diagram for the clocking and polling circuitry used for the laboratory bus.

A clock signal is used to increment a modulo 16 counter whose output is used to drive the four bus lines corresponding to the address of the master station. This address appears at each station on the bus, but since each station has its own unique four bit binary code, only the station whose code corresponds to the address on the master lines can respond. The only time a given station is permitted to gain control of the bus is when its particular code appears on the master lines and that particular station has a bus request pending. With these conditions satisfied, a station is granted bus control which causes the polling operation to be terminated.

Having been granted bus control, that station becomes the master station and may then initiate data transfer operations with a slave station which it designates. All data transfers are accomplished via handshaking between the master station and the slave station.



CLOCK AND POLL CIRCUIT FIGURE 4-1

A more detailed explanation of the data transfer operation is contained in Chapter VI.

Once a station gains control of the bus, it retains control until it initiates a bus release command which causes the polling operation to be resumed.

4.5 Clocking

Initially, the 555 timer integrated circuit, operating in the astable mode, was examined as a possible source of the clocking pulse for the counter. This proved to be impractical since its upper frequency limit in this mode is approximately 120 KHz (as determined from a breadboard experiment). This was considered to be too slow for satisfactory access to the bus. In general, a clock rate of approximately 1 MHz was desired, thus offering bus access to a new station every microsecond. Since no more than 16 stations were envisioned for the laboratory, no station would have to wait longer than 16 microseconds for access to the bus (provided the bus is not already busy). While in the poll mode, access to the bus would typically average eight microseconds. Considering the microprocessor's speed of operation, this was considered to be quite acceptable.

The 555 timer idea was quickly abandoned in favor of using the phase two (Ø2) clock signal driving the microprocessor itself. This signal, fully buffered, is conveniently available at each input/output slot on the mother board. This signal, an almost symmetrical square wave, had a period of 1.15 microseconds, thus providing the required clock pulse at a frequency of 870 KHz. This signal was thus chosen as the clock pulse for the counter and provides each

station access to the bus every 18.4 microseconds.

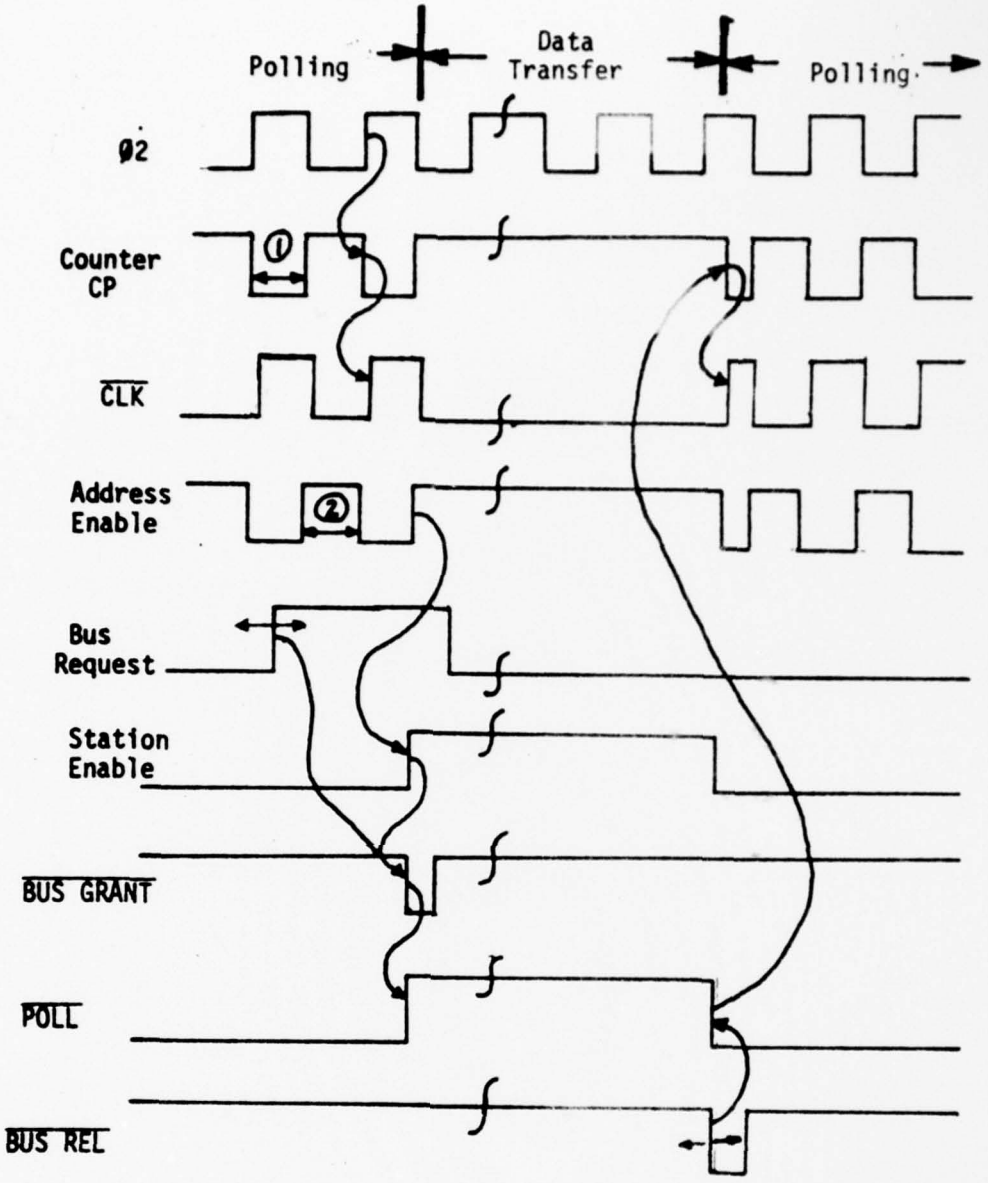
4.6 Clock and Poll Circuit Operation

The current operating mode of the clock and poll circuitry is determined by flip flop IC5. When it is set, the poll mode will be active indicated by $\overline{\text{POLL}}$ on the bus and the Ø2 clock signal is gated to the four bit binary counter causing it to be incremented on negative half cycles. The Ø2 signal is inverted before reaching the clock input of the counter to insure correct operation of other circuitry. This is better illustrated by the timing diagram of Figure 4-2.

The four bit binary output of the counter is applied to the bus drivers corresponding to the master address lines. The RC time delay circuit is used to delay $\overline{\text{CLK}}$ approximately 200 nanoseconds. This signal is used to indicate a valid master address. This prevents false reading of the master address lines.

Since all stations have their respective bus lines tied together as discussed in section 4.2, any station that causes an active transition on the $\overline{\text{BUS REL}}$ and $\overline{\text{BUS GRANT}}$ lines will affect the status of the mode flip flop. An active low on $\overline{\text{BUS REL}}$ will cause the flip flop to set, which places the flip flop in the polling mode, allowing the counter to place sequential addresses on the master address lines. If this flip flop is already in the polling mode, an active low on $\overline{\text{BUS GRANT}}$ causes the flip flop to clear, thus terminating the polling mode and initiating the data transfer mode.

For the laboratory prototype, the clock and poll logic was placed on the same Vector card as the microprocessor interface. This was done simply for convenience and to take advantage of available space and power.



Note: 1 - Counter changes and stabilizes
2 - Valid master address

Figure 4-2

TIMING DIAGRAM FOR GAINING BUS CONTROL

It may be desirable in the final installation to place the clock and poll circuitry external to all stations and have it contained in a small module available to the laboratory instructor. It may also be desirable to modify the design slightly to provide a capability for the laboratory instructor to initiate a bus release command should any station gain bus control and fail to release it after a reasonable period of time.

4.7 Chapter Summary

This chapter has presented a description of the laboratory bus. Information on data format, transmission line selection, line driver/receiver selection, signal levels and bus protocols has been provided to enable the reader to understand the organization of the bus structure.

Chapter V

Laboratory Computer System

5.0 General

This chapter will describe the microcomputer system chosen for the laboratory prototype station.

5.1 MC 6800 Microprocessor

The Motorola MC 6800 Microprocessing Unit (MPU) was selected for the microprocessor laboratory application because of several attractive features. First of all, it is well supported by a complete family of compatible chips - MC 6810 128 X 8 RAM, MC 6820 peripheral interface adapter (PIA), MC 6830 ROM, and MC 6850 asynchronous communications interface adapter (ACIA). This entire family of chips operates on a single +5 volt d.c. supply which greatly simplifies power requirements. All chips follow the same address and data bus organization which simplifies bus and interface requirements. The MC 6800 is a self-contained, 8 bit, microprocessor that is capable of operating with virtually any MOS or standard TTL device. The only signal requirement external to the MPU chip is a clock signal which is easily generated via one of three options.

Although the MC 6830 provides a micro-operating system (known by its trademark MIKBUG) for the 6800 MPU, there is practically no applications software available. This is not considered to be a severe constraint since most of the software requirements will be

generated by the laboratory users.

Another plus factor was the availability of the MC 6800 Microprocessor Applications Manual (17) which provides approximately 700 pages of valuable, detailed information on both hardware and software requirements and system considerations. Documentation of this kind is essential if a system is to be designed, constructed and placed "on the air" with a minimum of frustration.

5.2 Initial Considerations

In order to minimize costs, the initial plan was to purchase component parts and build a microcomputer prototype around the MC 6800 MPU and support chips. It quickly became obvious that this was not the best approach for several reasons. Purchasing in small quantities with resultant price inflation, long lead times and difficulty in locating sources of some needed parts all compounded the problem. Additionally, much labor would be required to perform hardware wiring and debugging of the system, a penalty which could not be paid in either time or money. Finally, the difficulty in reproducing additional copies of the prototype so as to achieve standardization among them was another factor entering the decision to abandon this approach.

5.3 System Selection

In July 1975, the Southwest Technical Products Corporation of San Antonio, Texas, announced a kit version of the MC 6800 with many available options to provide as much capability as the user required. It was decided to purchase one of these 6800 computer system kits

with 8K of RAM memory to support the microprocessor laboratory project. Construction of the kit required approximately 100 man-hours with an additional 20 man-hours required for system debugging and check out. The kit proved to be well designed and neatly packaged in an attractive cabinet.

Purchase of the SWTPC 6800 system was considered appropriate in that it was reasonably priced and was engineered to be directly compatible with a TV typewriter terminal (CT 1024), manufactured by the same company, which was available for use on this project.

Figure 5-1 provides a block diagram of the SWTPC 6800 computer system with TV typewriter terminal.

A brief description of each of the major system components will now be provided to describe overall system operation.

5.4 Keyboard and Encoder

The keyboard, an alphanumeric typewriter style, employs 53 single-contact, normally open keys arranged in a matrix. When any key is pressed, its corresponding ASCII code (see Appendix A) is generated by a diode matrix and a key-pressed output (active low $\overline{KP}$) is provided. Only seven bits of the standard eight bit ASCII code are generated which is adequate for most applications.

The keyboard and encoder was first printed in the February 1975 issue of Popular Electronics Magazine as a hobby project. Subsequently, it was made available in kit form by Southwest Technical Products Corporation. At $49.95, the kit is not only economical but simple and easy to build. One criticism of the kit is that it does not have a professional quality key touch and the keys have

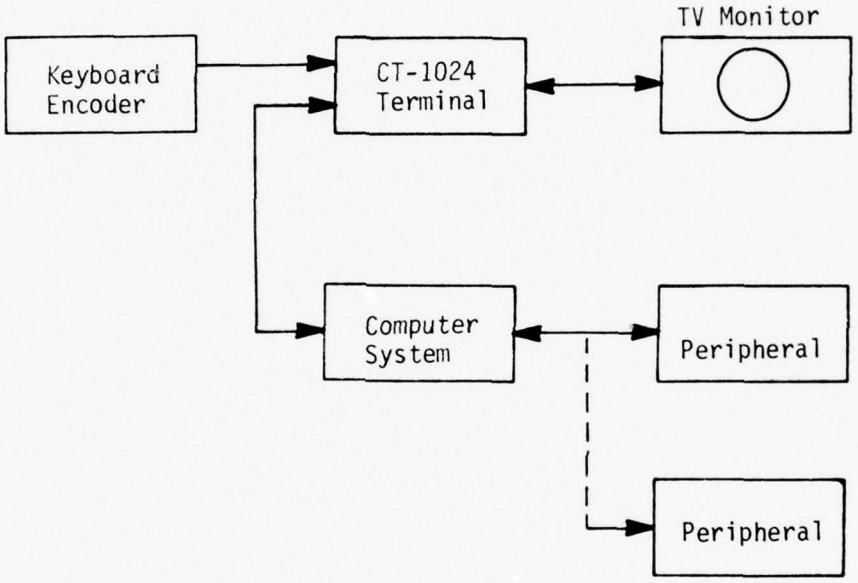


Figure 5-1

Prototype System Block Diagram

a habit of sometimes sticking, especially during damp weather.

5.5 CT-1024 Terminal

The CT-1024 terminal, also available from Southwest Technical Products Corp. in kit form, is designed to store and display two individual pages (each page containing 16 lines of 32 characters each) on a modified television or video monitor. Data input to the terminal comes from either the keyboard/encoder unit described earlier or from the computer system. With the computer off, the terminal's input comes directly from the keyboard/encoder. With the computer on any data from the keyboard first goes to the computer system and subsequently "echoed" to the terminal for display on the CRT monitor.

The entire screen of the video display has been arranged for 16 lines of 32 characters each. Although the second page of memory allows twice as many characters to be stored in memory, only one page can be displayed at a time. Each character displayed is actually an array of 35 dots arranged so there are five horizontal and seven vertical dots. A 2513 character generator IC chip decodes the ASCII data (provided at its input terminals from memory) into the correct dot patterns for the character to be displayed. The dots are selected and used one character row at a time since the horizontal trace of the television (or monitor) sweeps one video line at a time. Horizontal spacing between characters is provided by displaying a blank dot column between each displayed character, and vertical spacing is provided by sweeping three blank video lines between each set of seven "character dot video"

lines. This combination of keyboard/terminal has become known as the "TV typewriter."

Incorporated into the TV typewriter design are provisions for a cursor (indicating position of next character), and cursor control (up, down, left, right, home up). Line feed serves to move the cursor downward. Also provided is the capability to select memory pages as well as to erase memory either from cursor position to end of line (EOL) or from cursor position to end of frame (EOF).

To use a standard television receiver as a monitor, it is necessary to modify the TV slightly. Although the modification will vary from set to set, the idea is to connect the TV typewriter's video output to the input of the television's video amplifier, which is located between the last video IF stage and the video output circuit. It may or may not be necessary to provide d.c. bias restoration to the video output circuit depending on the make and model of television receiver used. Don Lancaster, in an article in Byte (15), has provided extensive information on ways and means of accomplishing this television interface. For additional information on the subject, the reader might wish to refer to Lancaster's forthcoming book, TV Typewriter Cookbook.

In order for the TV typewriter to communicate with other devices, all input and output data is converted from parallel ASCII into sequential "one bit at a time" form by the serial interface. The main component of the serial interface is the Univeral Asynchronous Receiver/Transmitter (UART) chip which provides for this conversion from the parallel form into a series of properly timed

ones and zeros including not only the serial data but the start, stop, and parity (if used) bits as well. The reverse is true in the receive mode. The standard baud rate (speed at which serial data is transmitted or received) provided is 110 baud. However, this rate was found to be too slow and the baud rate was increased to 300 baud by adding additional components. At 300 baud, the system can accept data at a rate which is faster than an operator would normally enter data thus eliminating the problem of "data overlap"at 110 baud. Also, the 300 baud rate capability will reduce the time required for data input/output by a factor of three.

In the transmit mode, the serial data leaves the UART chip through the transmitter serial output. This TTL level data is then converted to RS-232C format by additional circuitry.

In the receive mode, the incoming RS-232C formatted serial data is converted into TTL compatible levels and fed into the UART's serial input. When the UART chip sees the stop bits of the character being received, it raises its "output data available" line. When the character in the terminal's register is loaded, the "character accepted" line pulses low clearing the "output data available" line and generates a negative going "keypress strobe" to load a new character into the terminal's data registers.

Since the input/output connections are RS-232 compatible, they may be attached directly to most couplers and data sets. However, to record on or playback from magnetic tape, it will be necessary to employ an FSK encoder/decoder to get the digital data on or off the tape. The author understands that Southwest Technical Products

will have a tape cassette interface available soon. In the meantime, interested users of this system might want to investigate the possibility of using the "Kansas City" standard described in Byte (20).

5.6 MC 6800 Microcomputer

The 6800 microprocessor chip itself is a 40 pin eight bit parallel processor with 16 memory/peripheral address lines and an eight bit bi-directional data bus. There is a full complement of 72 basic instructions with five possible addressing modes (direct, relative, immediate, indexed, extended). There are six internal registers (program counter, stack pointer, index register, accumulator A, accumulator B and condition code register). The processor has both maskable and non-maskable interrupts which are executed as jumps to specific memory locations (vectored interrupts). A pushdown stack, located within user memory, is easily accessible and space limited only by the programmer and the amount of RAM memory available. A micro-operating system, the MC 6830 ROM MIKBUG, provides the ability to:

1) Load user programs or data into memory from either the keyboard or tape (where applicable).

2) Execute user programs.

3) List user programs or data within specified memory locations.

4) Print the contents of internal MPU registers.

5) Change the contents of specified memory locations or internal MPU registers.

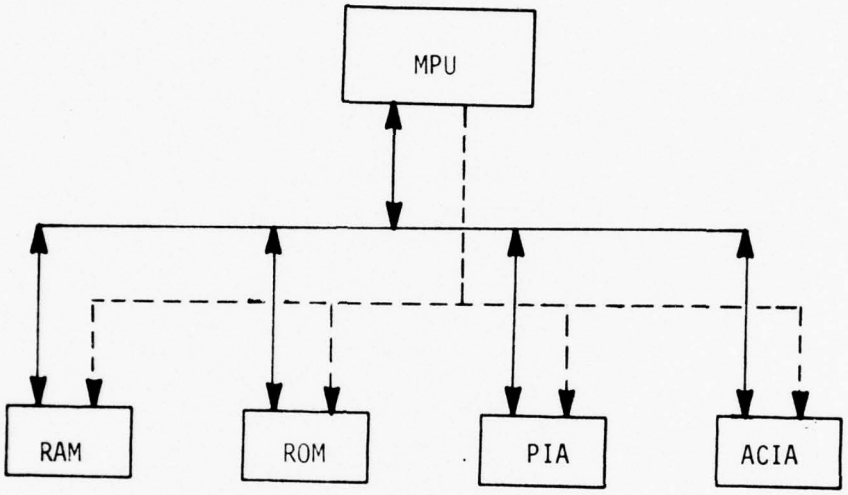


Figure 5-2

M6800 System Block Diagram

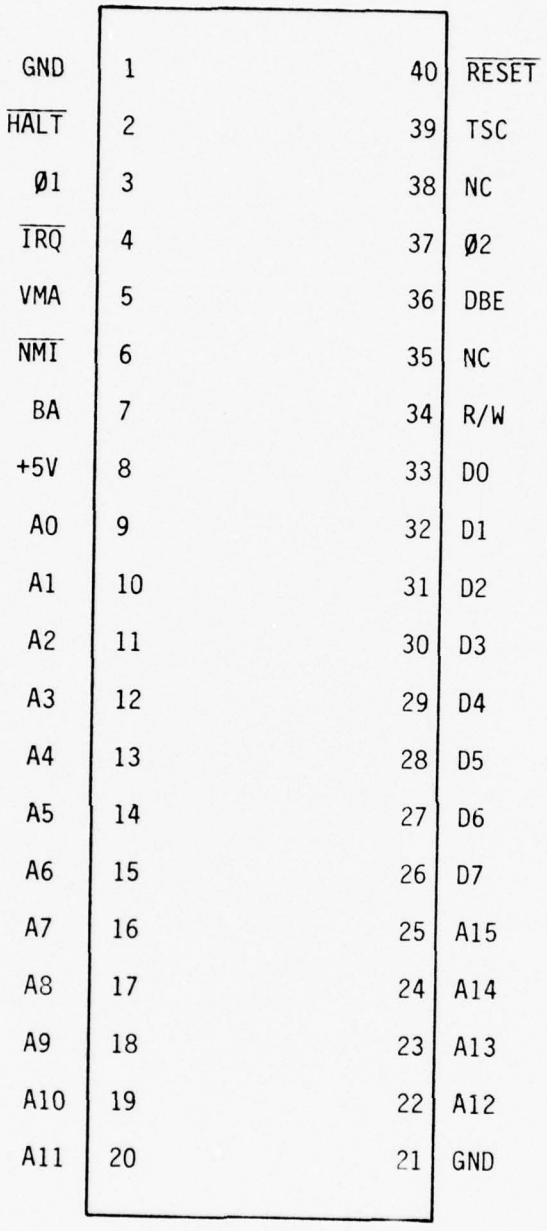


Figure 5-3

M6800 MPU Pin Out

For complete details on programming the MC 6800, the reader is referred to reference 18 of the bibliography.

Figure 5-2 shows a simplified depiction of the MC 6800 system.

The 6800 MPU is contained in a 40 pin package as shown in Figure 5-3.

5.6.1 M6800 MPU Internal Registers

5.6.1.1 Index Register

The index register (X) is a 16 bit (2 byte) register which is primarily used to store a memory address in the indexed mode of addressing. The index register may be incremented, decremented and stored.

5.6.1.2 Program Counter

The program counter (PC) is a 16 bit register containing the address of the next byte to be fetched from memory. When the current value of the program counter is placed on the address bus, the program counter will be incremented automatically.

5.6.1.3 Stack Pointer

The stack pointer (SP) is a 16 bit register that contains a beginning address in RAM where the status of the MPU registers may be stored when the MPU has other functions to perform, such as during an interrupt or during a branch to subroutine. The address in the stack pointer is the starting address of sequential memory locations in RAM where MPU status registers will be stored as follows:

Stack Pointer Address: contents of PC (low byte)

Stack Pointer Address - 1: contents of PC (high byte)

Stack Pointer Address - 2: contents of IX (low byte)

Stack Pointer Address - 3: contents of IX (high byte)

Stack Pointer Address - 4: contents of ACCA

Stack Pointer Address - 5: contents of ACCB

Stack Pointer Address - 6: contents of CC

As the status of each register is stored on the stack, the stack pointer will be decremented automatically. When the stack is unloaded (status of registers restored), the status of the last register on the stack will be the first register that is restored. The stack pointer always points to the next memory location available for stack operations.

5.6.1.4 Condition Code Register

The condition code register is an 8 bit register. Each individual bit may be set or cleared by the execution of an instruction. To see how each instruction affects the condition code register, refer to the M6800 programming manual (18). The primary use of this register is in the execution of the conditional branch instruction. The function of each bit in the register is shown in Table 5-1.

Table 5-1

Condition Code Register

7	6	5	4	3	2	1	0
1	1	H	I	N	Z	V	C

Table 5-1 - continued

Bit No.	Function
0	C (carry-borrow test)
1	V (overflow test)
2	Z (zero test)
3	N (negative test)
4	I (interrupt mask)
5	H (half carry test)
6	always logic 1
7	always logic 1

5.6.2 Summary of MPU Operation

The MPU requires a two phase symmetrical, TTL compatible, non-overlapping clock signal. During phase 1 (Ø1), an address is placed on the address bus by the MPU. During the second phase (Ø2) of the clock, the bi-directional data bus will be active. The first byte of an instruction enters the MPU and is transferred into an internal instruction register and decoded by the MPU. The MPU will then have the information needed to read in an additional one or two bytes as necessary to form the effective address of the operand. Once the entire instruction is read into the MPU (one, two or three bytes) the instruction is then executed. The MPU then reads in the next sequential byte in the program and places it again in the instruction register. The program will be executed sequentially in this manner unless a branch or jump instruction changes the value of the program counter. If this occurs, the next instruction to be executed is determined by the new program counter value.

5.6.3 M6800 Interrupt Handling

If an interrupt or reset occurs during program execution, the program counter will be changed according to Table 5-2 with the program counter receiving the new value indicated in order to provide a vectored jump to the proper service routine.

When an interrupt occurs, the stack pointer will be used to store the contents of the internal registers. This permits a return to the proper location of the interrupted program when an RTI (return from interrupt) instruction is executed. In a similar manner the program counter is changed when an RTS (return from subroutine) instruction is executed.

Table 5-2

Interrupt Vector Memory Assignments

Interrupt	Address
$\overline{\text{RESET}}$ (Low Byte)	FFFF
$\overline{\text{RESET}}$ (High Byte)	FFFE
$\overline{\text{NMI}}$ (Low Byte)	FFFD
$\overline{\text{NMI}}$ (High Byte)	FFFC
SWI (Low Byte)	FFFB
SWI (High Byte)	FFFA
$\overline{\text{IRQ}}$ (Low Byte)	FFF9
$\overline{\text{IRQ}}$ (High Byte)	FFF8

5.6.4 M6800 Memory

The SWTPC M6800 computer system uses the 2102-1 1024 X 1 bit static random access memory with an access time of 500 nanoseconds.

A total of 8K RAM memory was purchased with the M6800 system to support software development.

The memory chips are configured in a matrix fashion as indicated in Table 5-3. Each printed circuit memory board is capable of accommodating 4K of memory and a total of eight memory boards can be placed in operation giving a total capability of 32K. As shown in Table 5-3, each memory quadrant contains 1K of memory capacity with eight 2102-1 chips being used in each quadrant, each chip providing one bit of an eight bit word.

To fully understand how memory addressing is accomplished, the reader is referred to Figure 5-4. As shown, address line A15 is used as an enable line to a one of eight decoder. When A15 is low (as it will be when addressing RAM), Ø2 is low and VMA is high, the decoder will be enabled thus allowing the contents of address lines A14, A13 and A12 to determine which decoder output will be active low. Each memory board is connected to a different decoder output line such that only one board is enabled at a time. This connection is made automatically depending on which slot the board is plugged into. Next, the contents of address lines A11 and A10 determine which one of four outputs are active low. Each output is connected to the chip enable line ($\overline{CE}$) of all memory chips in the appropriate quadrant. Having selected a memory board and quadrant, a particular word in memory is selected by applying address lines A9 through A0 to the corresponding address inputs on the 2102 memory chips. These 10 address lines (A0 - A9) as well as the R/W line of all memory IC's are parallelled.

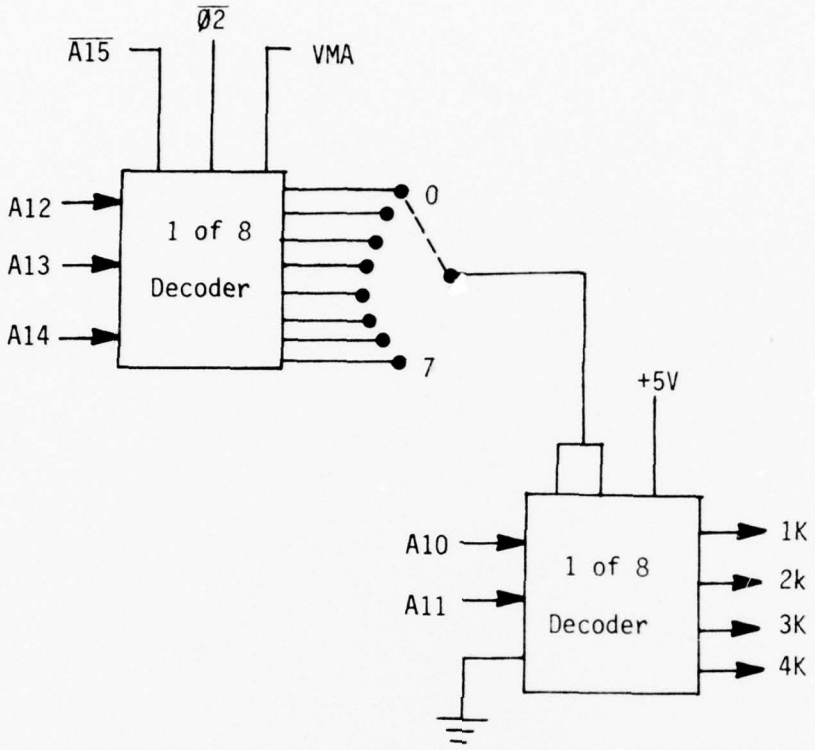


Figure 5-4

Memory Addressing Scheme

Table 5-3

M6800 Memory Address Assignment Table

Board #	Quadrant	Starting Address	Ending Address
0	1	0000	03FF
0	2	0400	07FF
0	3	0800	0BFF
0	4	0C00	0FFF
1	1	1000	13FF
1	2	1400	17FF
1	3	1800	1BFF
1	4	1C00	1FFF
2	1	2000	23FF
2	2	2400	27FF
2	3	2800	2BFF
2	4	2C00	2FFF
3	1	3000	33FF
3	2	3400	37FF
3	3	3800	3BFF
3	4	3C00	3FFF
4	1	4000	43FF
4	2	4400	47FF
4	3	4800	4BFF
4	4	4C00	4FFF
5	1	5000	53FF
5	2	5400	57FF
5	3	5800	5BFF

Table 5-3 - continued

Board #	Quadrant	Starting Address	Ending Address
5	4	5C00	5FFF
6	1	6000	63FF
6	2	6400	67FF
6	3	6800	6BFF
6	4	6C00	6FFF
7	1	7000	73FF
7	2	7400	77FF
7	3	7800	7BFF
7	4	7C00	7FFF

5.7 Peripheral Interface Adapter (PIA)

The PIA, another 40 pin chip, provides versatility and flexibility in interfacing the 6800 system with peripheral devices.

The PIA communicates with the MPU via an eight bit bi-directional data bus, three chip selects, two register selects, two interrupt request lines, one read/write line, an enable line, and a reset line. These are shown in Figure 5-5 and will be discussed later.

Each PIA has two eight bit bi-directional peripheral data buses for interfacing with peripheral equipment. Each peripheral data line may be programmed to act as an input or an output. In addition to the two eight bit peripheral data buses, peripheral control lines CA2 and CB2 may be programmed to act as a peripheral input or output control lines. Control lines CA1 and CA2 are fixed to carry control information from the peripheral devices to the PIA.

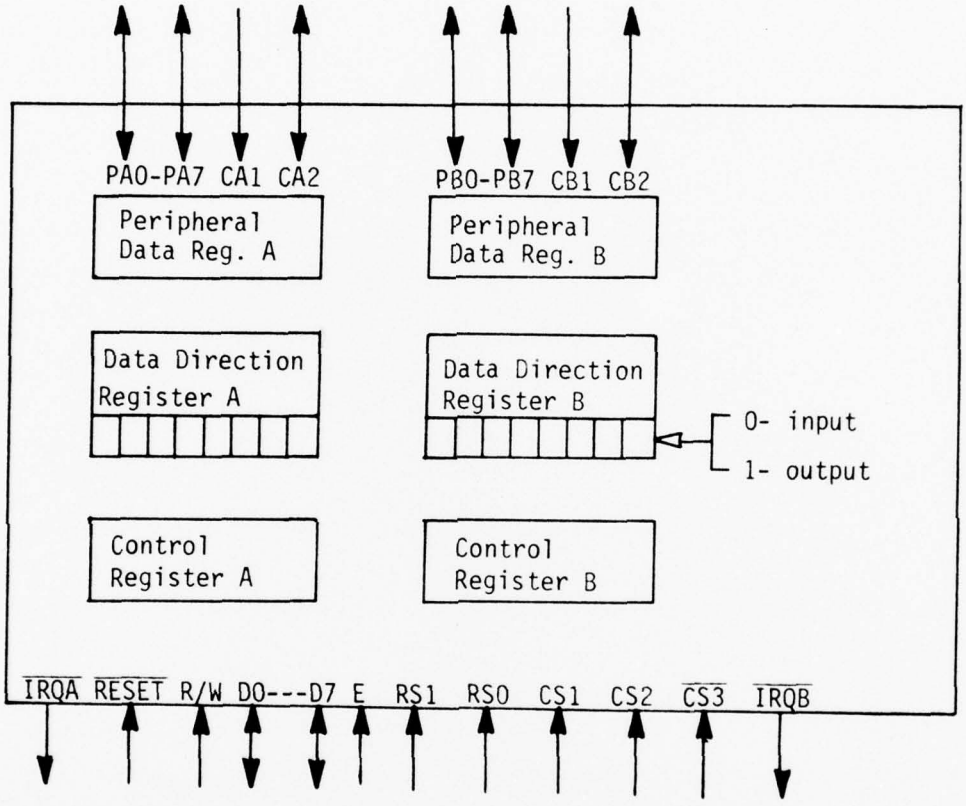


Figure 5-5

Peripheral Interface Adapter

Each PIA consists of two control registers, two data direction registers, and two peripheral interface registers (peripheral data). The control registers and the data direction registers are used to control the data in and out of the PIA. This is accomplished by setting a "1" in the corresponding bit in the Data Direction Register (DDR) if the line is to be an output or a "0" in the DDR if the line is to be an input. When the data in the peripheral data lines are read into the MPU by a load instruction, those lines that have been designated as input lines (0 in DDR) will be gated directly to the data bus and into the register selected in the MPU.

When an output (write) instruction is executed to send data to a peripheral, data will be transferred from the MPU via the data bus to the peripheral data register and hence to the peripheral device depending upon whether the respective data lines have been programmed as outputs by the DDR. For additional information on programming the PIA to achieve its full capability, the reader is referred to reference

In summary, some of the important features of the PIA are:

1) 8-bit bi-directional data bus for communications with MPU.

2) Two bi-directional 8-bit buses for interface with peripherals.

3) Two programmable control registers.

4) Two programmable data direction registers.

5) Four individually controlled interrupt input lines; two useable as peripheral control outputs.

6) Handshake control logic for input/output peripheral operation.

5.8 Chapter Summary

This chapter has described the Motorola M6800 microcomputer system chosen for the University of Florida's microprocessor laboratory. Reasons for its selection have been presented along with information on a compatible terminal. Memory addressing information was included along with information on using the peripheral interface adapter as a convenient method of interfacing the MC6800 with the laboratory bus or other peripherals.

Chapter VI

M6800 Interface and Data Transfer Operations

6.0 General

This chapter describes the interface between the M6800 computer system and the laboratory bus. Included is a detailed account of bus data transfer operations.

6.1 Station Decode Circuitry

As stated in section 4.4, each station on the laboratory bus must have its own unique station code to insure that only one station responds to a given code. Figure 6-1 shows the station decode logic as well as the other logic necessary to interface the M6800 microprocessor station(s) with the laboratory bus.

No action can take place on the bus until a station gains control of the bus. This action will be described in detail.

While in the polling mode, the four master address lines of the bus will contain the address (station code) of the station being offered bus control. Thus, bus control is offered to only one station at a time provided no two stations have the same code. Note from the clock and poll logic diagram of Figure 4-1 that the master address is enabled onto the bus after the output of the binary counter is stabilized. This prevents false master addresses from appearing on the bus.

As shown in Figure 6-1, receivers (IC3) at each station take the master address from the bus converting it into TTL compatible signals.

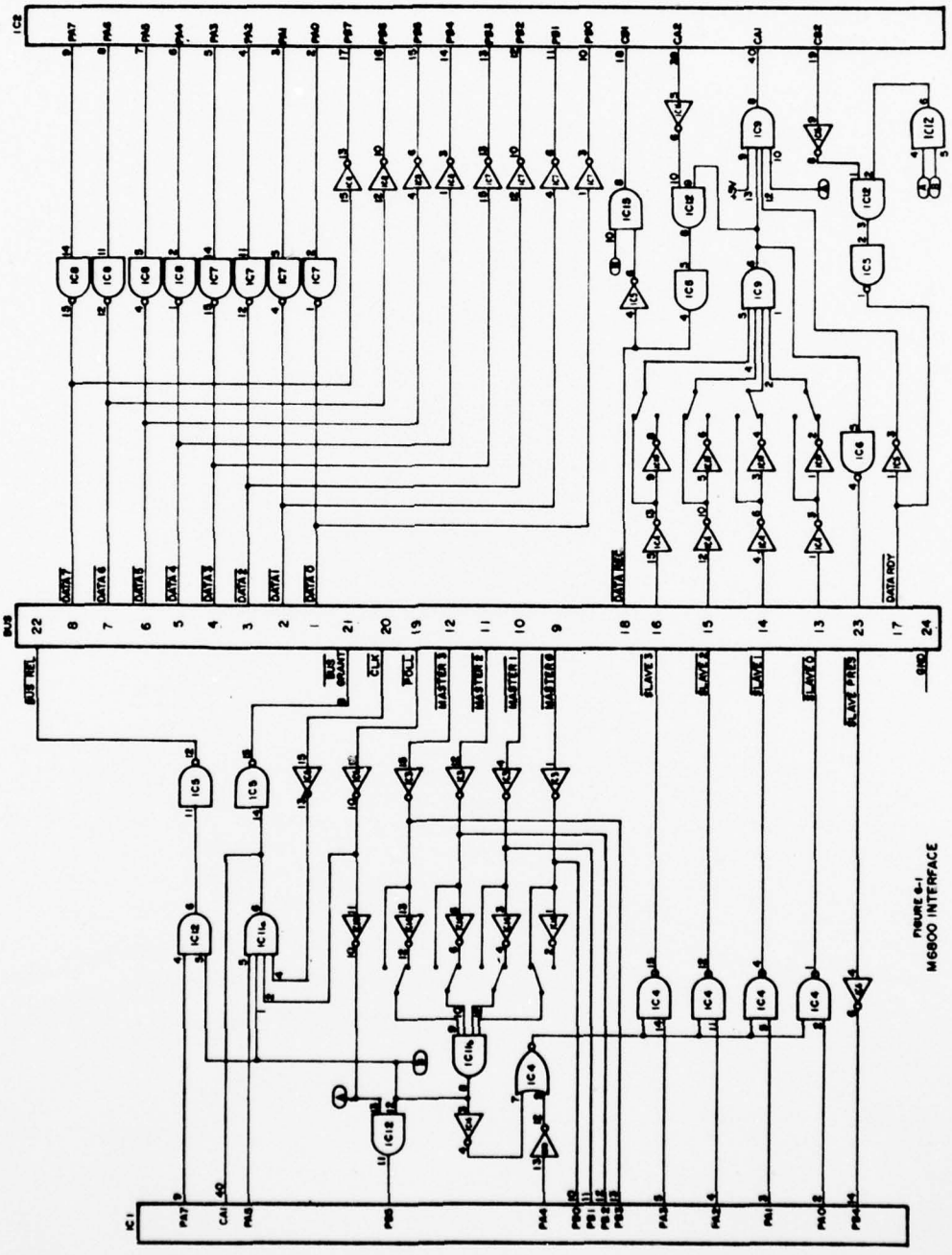


FIGURE 6-1
M6800 INTERFACE

This master address is then applied to the station decode circuit (IC14). The inputs to the master enable gate (IC11b) are jumper selected so as to insure that all four inputs to IC11 are high based on the unique code for each station. The output of IC11b, master enable, will then go high when that station's address appears on the master address lines.

There are two notes of particular importance at this point. Figure 6-1 reflects the station code for station one only. Other stations on the bus would have the proper configuration of the station decode circuitry corresponding to their own unique address. Also, no microprocessor station can have station code zero as this address is reserved solely for the NOVA computer.

The master enable signal is a key factor in initiating any further action on the bus. As shown in Figure 6-1, it is used to enable a valid bus request from the station whose address currently appears on the master address lites.

6.2 Gaining Bus Control

A station can gain control of the bus in the following manner. It must first initiate an active high bus request via peripheral interface adapter IC1 PA5. Also the clock signal on the bus must be active low (indicated by $\overline{\text{CLK}}$), and polling must be in effect (indicated by $\overline{\text{POLL}}$). With these conditions satisfied, master select (output of IC11a) will go high momentarily when that station's code appears on the bus. When master select goes high it initiates an interrupt request via IC1 CA1 while simultaneously causing the bus grant line to go active low ($\overline{\text{BUS GRANT}}$) which clears the mode flip flop causing the

poll mode to be terminated and the data transfer mode to be entered (see Figure 4-1). This change in mode stops the Ø2 clock signal to the counter, freezing its output to the address of the station with bus control. This process of gaining bus control is shown in the timing diagram of Figure 4-2.

Master enable is also used to enable the drivers associated with the slave address lines. This insures that the slave address is specified only by the station with bus control.

Master enable is also used to gate a valid bus release command onto the bus. As shown in Figure 6-1, master enable is applied as one input to AND gate IC12. The other input, an active high bus release command, is initiated under software control and comes from IC1 PA7. This insures that only the master station has the capability of causing an active low $\overline{\text{BUS REL}}$ condition on the bus. This prevents another station from inadvertently causing a bus release command to be executed while the master station is performing a data transfer operation.

Additionally, the master enable signal is applied as an input to IC1 PB5 to allow the software to determine that a bus grant has been made and the station is now in control of the bus. This determination may also be accomplished by performing a read operation on the control register of the appropriate PIA as explained in section 6.4 of this chapter.

6.3 Interrupt Prioritizing

In the present design, a "round robin" priority scheme is used on the bus. This insures that each microprocessor station

has an equal opportunity for gaining bus control. It also insures that no station can regain control of the bus until all other stations are afforded an opportunity to do so. Thus no station can jump in line ahead of another station. This helps to prevent any one station from "hogging the bus".

Only two types of interrupts may occur in this application. When a station is granted bus control and becomes the master station, an interrupt signal will appear at IC1 CA1. The other case occurs when a station has its code appear on the slave address lines and data is ready to be transferred. In this latter case the interrupt signal will appear at IC2 CA1. No external interrupt prioritizing circuitry is needed to satisfy this application since the $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$ lines are OR-tied and fed to the interrupt request ($\overline{\text{IRQ}}$) line of the MPU. Determination as to which interrupt occurred is then accomplished under software control as explained in the next section.

6.4 Interrupt Handling

When the $\overline{\text{IRQ}}$ line of the MPU goes active low, the MPU will finish execution of its current instruction, stack its registers, set its interrupt mask and fetch the starting address of the $\overline{\text{IRQ}}$ interrupt service routine by vectoring to memory locations FFF8 and FFF9. The service routine then determines which kind of interrupt has occurred by performing a read operation on the contents of the A and B control registers. If bit 7 of IC1 control register A is set, then the station must perform as a master station. If bit 7 of IC2 control register A is set, then the station must perform as a slave station. Appropriate software is required to accomplish each of these functions.

See Figures 6-3 and 6-4 for guides to these programming requirements.

It should be noted at this point that the interrupt input lines were not chosen arbitrarily. They were selected as indicated above to take advantage of the handshaking capability that exists inherently in each peripheral interface adapter. The interested reader will find complete details on handshaking contained in section 3-4.1 of reference

6.5 Master to Slave Data Transfer

With the interface in the data transfer mode, the master station must now initiate those actions necessary to accomplish data transfer to the slave station. Figure 6-3 is a flow chart depicting generally how the data transfer operation is performed by the master station and serves as a model for the software that must be written to accomplish this operation.

Having initiated a bus request command as described previously the software must cause the station to wait for confirmation that bus control has been granted. This can be verified by checking bit 7 of IC1 control register A via a read operation to see if the flag associated with CA1 interrupt input line is set. If set, this means that the station had requested and was granted bus control as the master station.

The next step is to verify that the data transfer mode of operation is in effect. This is accomplished by checking IC1 PB5 for a logic one or high state.

The master station must next specify the address of the slave station. This is done via IC1 PA0-PA3 which becomes a valid slave

AD-A044 607 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO F/G 9/2
HARDWARE DESIGN AND CONSTRUCTION OF A MICROPROCESSOR LABORATORY--ETC(U)
1976 J H HILL

UNCLASSIFIED AFIT-CI-77-9 NL

2 OF 2
ADA
044607

END
DATE
FILMED
10-77
DDC

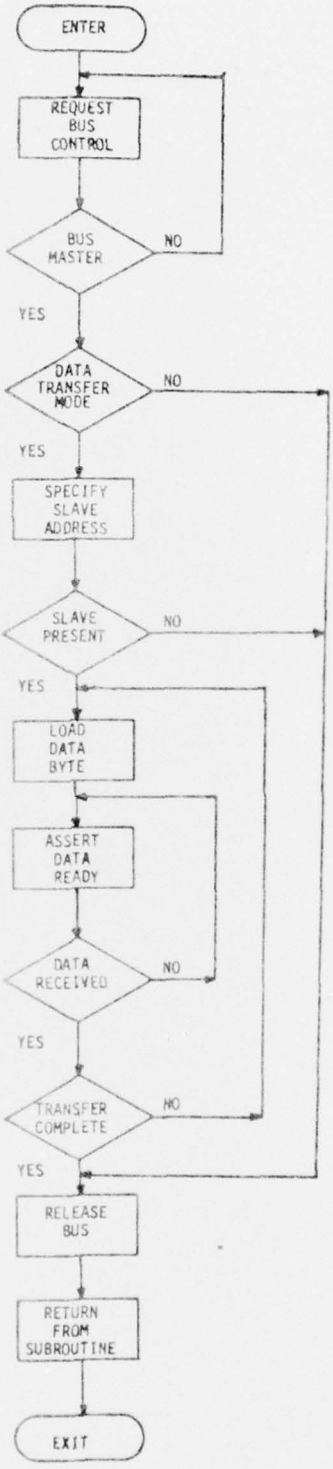


Figure 6-3

Master Station Flow Chart

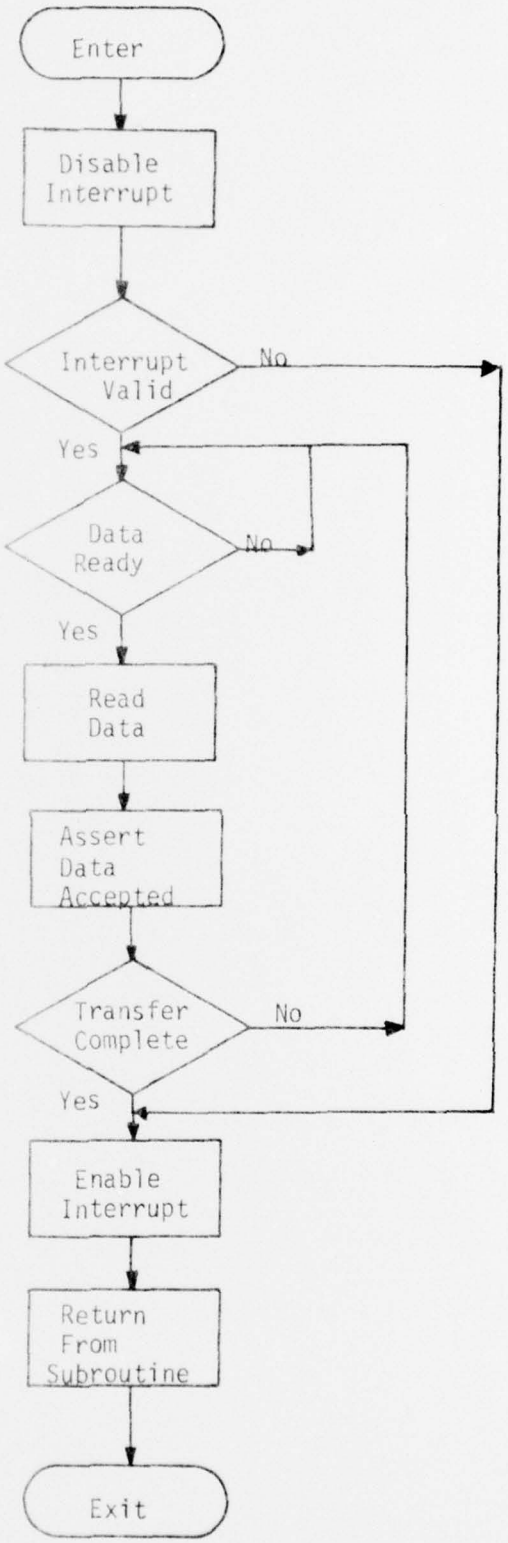


Figure 6-4 Slave Station Flow Chart

address when IC1 PA4 is asserted high to indicate "address valid now." This address then appears on bus lines $\overline{\text{SLAVE 0}}$ - $\overline{\text{SLAVE 3}}$.

If the station specified as the slave is present, it will cause the $\overline{\text{SLAVE PRES}}$ line to go active low indicating its presence allowing the master station to verify this condition by examining IC1 PB4 for a high level input. If the test for this condition proves false, then the desired slave station is not present on the bus and thus the desired data transfer cannot take place. Should this be the case, the master station would exit its routine by initiating a bus release command and executing an RTS instruction.

If the slave station is present, the master station can proceed with the data transfer routine by loading the first byte of data into IC2 PA0-PA7 and asserting data ready at IC2 CB2 (an active low transition). CB2 must have been programmed previously as an output by setting bit 5 of IC2 control register B equal one (high) and bit 4 of control register B equal zero (low). Also bit 3 of control register B must be zero (low) to establish the handshake mode for the data transfer operation.

Due to the fact that the PIA port convention (A - output, B - input) was established by Southwest Technical Products Corp. as contrary to that recommended by Motorola, it should be noted that an A side Read or B side Write operation will not generate the strobe necessary for handshaking. This limitation is conveniently overcome by including "dummy" instructions in the program. For example, an A side Write instruction must be followed immediately by an A side dummy Read to generate the handshake strobe. Similarly, a B side

Read can be followed by a dummy Write instruction. Use of the dummy instruction is explained in reference

By causing an active low signal at IC2 CB2, the master station can then inform the selected slave station that a byte of valid data is available and ready to be received. The data ready command is placed on the bus and will cause an interrupt to the designated slave station via IC2 CA1. The slave station then examines the flag bits of IC2 control register A and B to determine the source of the interrupt. If bit 7 of IC2 control register A is set, then this confirms that valid data is available from the master station. The slave station then performs a READ output register B followed by a dummy READ output register A operation in order to generate the strobe necessary for the handshake operation. During handshaking, this strobe acknowledging receipt of data is generated automatically at IC2 CA2.

The data received command from the slave station will cause the master station's data ready line (IC2 CB2) to go inactive, automatically permitting a new data byte to be loaded before asserting the data ready line again. This handshake process will continue until all data is transferred.

6.6 Slave to Master Data Transfer

The reader should note that although a master station can send data to a slave station, it cannot ask the slave station to send data without reversing roles. This can be accomplished under software control in the following manner.

Any station desiring to receive data from another station must first gain bus control as the master station. It then designates the station from which it desires data by specifying its address on the slave address lines. It now begins a data transfer operation with the slave station using a header message.

The first byte of the header might indicate the address of the current master station, whether the master station is sending data or wishes to receive data, and the priority of the transmission or request. For example, bit 7 (MSB) of the first byte might be used to indicate whether data is being sent or whether data is being requested. Bits 6-3 might be used to indicate the address of the current master station. Bits 2-0 might be used to indicate the priority. The second and third bytes of the header message might be used as follows. As a data sender, the second and third bytes might specify the number of bytes of data being sent by the master station. As a requester of data, the second and third bytes might be used to identify the specific data being requested.

After receipt of the header message, the current slave station would examine the header to determine if it is to continue receiving data in the slave role or whether it must change roles in order to transmit data. Tables 6-1 and 6-2 contain sample header formats.

This scheme would be implemented in software. What has been suggested here is only a first approximation of how this capability could be provided. A detailed study of the software requirements is beyond the scope of this paper.

Table 6-1

Header Format for Data Sender

	FORMAT							
Byte	B7	B6	B5	B4	B3	B2	B1	B0
first	1	A3	A2	A1	A0	P2	P1	P0
second	Number of bytes of							
third	data being sent							
	A3 - A0: Address of sender							
	P2 - P0: Priority of data being sent							

Table 6-2

Header Format for Data Requester

Byte	B7	B6	B5	B4	B3	B2	B1	B0
first	0	A3	A2	A1	A0	P2	P1	P0
second	Indicates specific kind							
third	of data requested							
	A3 - A0: Address of requester							
	P2 - P0: Priority of request							

6.7 Port Convention

It should be noted that the input/output port convention used in the interface design was dictated by the Southwest Technical Products Corporation's (SWTPC) MP-L parallel interface board. As mentioned earlier, the A port is wired as an output port while the B port is wired as an input port. This particular choice of convention by SWTPC is contrary to that recommended in the Motorola

Applications Handbook (17). Why this convention was selected in the SWTPC MP-L design is unknown to the author. He can only speculate that the choice was an arbitrary one on the part of the SWTPC designers.

Although this convention will work quite well as long as loading specifications are followed as indicated in the Motorola Handbook, it is unfortunate that this convention was selected as it unnecessarily complicates the handshaking operation by requiring dummy instructions and fails to take full advantage of the inherent capabilities of the peripheral interface adapter.

Consideration was given to changing the MP-L board and adopting the Motorola convention. However, in order to use any future software packages (resident editor/assembler, etc.) available from SWTPC it was decided to stick with the SWTPC convention.

6.8 Chapter Summary

This chapter has described the interface between the M6800 microcomputer system and the laboratory bus using the inherent handshaking capabilities provided by the peripheral interface adapter. The procedures necessary for gaining control of the bus were described as well as a detailed account of the bus data transfer operations.

Chapter VII

Conclusions And Recommendations

7.0 Conclusions

1) Microprocessors are presenting a new dimension in digital design, providing a literal explosion in new applications.

2) It is essential that an adequate microprocessor training facility be provided at the University of Florida to permit meaningful microprocessor hardware and software experience for computer engineering students.

3) The microprocessor laboratory described in this paper is a first step in providing a capability for extended research into important areas of multiprocessor and parallel processor systems, distributed networks, fault tolerant processing, load leveling, array or vector processing, etc.

7.1 Recommendations

1) The microprocessor laboratory at the University of Florida should be expanded to include microprocessor types other than the Motorola M6800 system. This would provide a real basis of comparison, especially in such areas as architecture and benchmark performance.

2) System software needs to be written providing the control

and support programs required for maximum utilization of the hardware capabilities.

3) Research should be initiated into the integration of multiple functional units into a multiprocessing or parallel processing system to examine various hierarchical and distributed networks.

4) During formal course work, increased emphasis needs to be placed on the specification of the rules governing the flow of data between functional units and the possibility of allowing multiple simultaneous paths. One suggestion is to require the class (or each student) to design a bus protocol ideally suited to such organizations and structures.

5) Directly related to recommendation 4 is the need for a better understanding of the input/output problem with its concomitant interface problem. This is an area requiring additional research.

6) Suggest that students be provided maximum access to the microprocessor laboratory and encouraged to develop software that supports microprocessor applications.

7) Suggest students, faculty, and hobby groups provide suggestions as to how the microprocessor laboratory could be utilized to its fullest capability to provide useful services to the campus community while not actively serving as a laboratory facility.

Appendix A

Below is a table listing the American Standard Code for Information Interchange (ASCII). Only 7 bits of the code are listed; the eighth bit is often a 1 all the time or used as a parity bit for error testing.

b7	b6	b5	b4	b3	b2	b1	Column / Row	0	1	2	3	4	5	6	7
								0	0	0	0	1	1	1	1
								0	0	1	1	0	0	1	1
								0	1	0	1	0	1	0	1
	0	0	0	0		0		NUL	DLE	SP	0	@	P	\	p
	0	0	0	1		1		SOH	DC1	!	1	A	Q	a	q
	0	0	1	0		2		STX	DC2	"	2	B	R	b	r
	0	0	1	1		3		ETX	DC3	#	3	C	S	c	s
	0	1	0	0		4		EOT	DC4	$	4	D	T	d	t
	0	1	0	1		5		ENQ	NAK	%	5	E	U	e	u
	0	1	1	0		6		ACK	SYN	&	6	F	V	f	v
	0	1	1	1		7		BEL	ETB	'	7	G	W	g	w
	1	0	0	0		8		BS	CAN	(	8	H	X	h	x
	1	0	0	1		9		HT	EM	)	9	I	Y	i	y
	1	0	1	0		10		LF	SUB	*	:	J	Z	j	z
	1	0	1	1		11		VT	ESC	+	;	K	[	k	{
	1	1	0	0		12		FF	FS	,	<	L	\	l	|
	1	1	0	1		13		CR	GS	-	=	M	]	m	}
	1	1	1	0		14		SO	RS	.	>	N	^	n	~
	1	1	1	1		15		SI	US	/	?	O	-	o	DEL

BIBLIOGRAPHY

1. Bell, C.G. and Newell, A. Computer Structures: Readings and Examples. McGraw-Hill. New York, N.Y. 1971.

2. Cushman, Robert H. "EDN's Second Annual Microprocessor Directory". Electronic Design News. November 20, 1975.

3. Digital Equipment Corporation. "PDP-11 Peripherals Handbook". Maynard, Massachusetts. 1975.

4. Electronic Industries Association. "Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange". EIA Standard RS-232-C. Washington, D.C. August 1969.

5. Electronics. "Diverse Industry Users Clamber Aboard the Microprocessor Bandwagon". July 11, 1974.

6. Fairchild Semiconductor Corporation. "The TTL Applications Handbook". Mountain View, California. August 1973.

7. General Electric Company. "Introductions to Data Communications". Document Number CPB-322A. July 1964.

8. Gilb, T. "Microcomputers - Present Properties and Probable Applications". Computers and People. February 1975.

9. Horelick, D. and Larsen, R.S. "CAMAC: A Modular Standard". IEEE Spectrum. April 1976.

10. International Business Machines Corporation. "IBM Synchronous Data Link Control General Information". Document Number GA 27-3093-0. Research Triangle Park, North Carolina. 1974.

11. International Business Machines Corporation. "IBM System 360 I/O Interface Channel to Control Unit". Document Number A22-6843. Poughkeepsie, New York. 1973.

12. International Organization For Standards. "Data Communication - High-level Data Link Control Procedures". Draft of ISO/D15 3309.2 proposed standard. June 1975.

13. Kersey, J.R. "Synchronous Data Link Control". Data Communications. May/June 1974.

14. Knoblock, D.E. et al. "Insight Into Interfacing". IEEE Spectrum. May 1975.

15. Lancaster, Don. "Television Interfacing". Byte. October 1975.

16. Madnick, Stuart E. "The Future of Computers". Technology Review. July/August 1973.

17. Motorola Semiconductor Products Inc. "M6800 Microprocessor Applications Manual". Phoenix, Arizona. 1975.

18. Motorola Semiconductor Products Inc. "M6800 Microprocessor Programming Manual". Phoenix, Arizona. 1975.

19. National Semiconductor Corporation. "Interface Integrated Circuits". Santa Clara, California. April 1974.

20. Peschke, M. "Byte's Audio Cassette Standards Symposium". Byte. February 1976.

21. Reyling, George Jr. "Microprocessors - The Next Generation in Digital Design". National Semiconductor Corporation. 1974.

22. The Institute of Electrical and Electronics Engineers. "IEEE Standard Modular Instrumentation and Digital Interface System (CAMAC)." IEEE Standard 583-1975. New York, N.Y. 1975.

23. United States Department of Defense. "Common Long Haul and Tactical Communication Technical Standards - MIL STD 188-100". Washington, D.C. November 1972.

BIOGRAPHICAL SKETCH

James Harold Hill, Jr. was born in Laurens, South Carolina on March 1, 1938. Graduating from Laurens High School in June 1956, he entered Clemson University where he received the degree of Bachelor of Science in Electrical Engineering in August 1960. He worked briefly for the Federal Aviation Administration in Washington, D. C. prior to entering active duty with the United States Air Force in April 1961. He is a Regular Air Force officer currently serving in the rank of Major. He was selected for graduate studies under the Air Force Institute of Technology (AFIT) program and entered the University of Florida in September 1974 where he has pursued studies leading to the degree of Master of Engineering.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Engineering.

Keith L. Doty, Chairman
Associate Professor of
Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Engineering.

Stanley Y. Su
Associate Professor of
Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Engineering.

James K. Watson
Associate Professor of
Electrical Engineering

This thesis was submitted to the Graduate Faculty of the College of Engineering and to the Graduate Council, and was accepted as partial fulfillment of the requirements for the degree of Master of Engineering.

June, 1976

Dean, College of Engineering

Dean, Graduate School