intel
McGraw-Hill

intel McGraw-Hill

COMMUNICATION SYSTEMS

PRACTICAL GUIDE TO INTEL'S CONNECTIVITY DESIGNS

NATHAN GUREWICH

COMMUNICATION SYSTEMS GUREWICH

McGraw Hill

Communication Systems

Intel/McGraw-Hill Series

Published

BABBAR/STECIAK • *The Official Intel 386SL Portable Computer Book*

BUNZEL/MORRIS • *Multimedia Applications Development: Using DVI Technology*

EDELHART • *Intel's Official Guide to 386 Computing*

INTEL • *Intel386 Family Binary Compatibility Specification 2*

INTEL • *i486 Microprocessor Programming Reference Manual*

LUTHER • *Digital Video in the PC Environment*

MARGULIS • *i860 Microprocessor Architecture*

RAGSDALE • *Parallel Programming*

Communication Systems

Practical Guide to Intel's Connectivity Designs

Nathan Gurewich

McGraw-Hill, Inc.

New York St. Louis San Francisco Auckland Bogotá
Caracas Lisbon London Madrid Mexico Milan
Montreal New Delhi Paris San Juan São Paulo
Singapore Sydney Tokyo Toronto

Library of Congress Cataloging-in-Publication Data

Gurewich, Nathan
Communication systems : practical guide to Intel's connectivity designs / Nathan Gurewich.
p. cm. — (Intel/McGraw-Hill series)
Includes index.
ISBN 0-07-025234-3
1. Local area networks (Computer networks). 2. Electronic digital computers—Circuits. 3. Computer network protocols. I. Title. II. Series.
TK5105.7.G87 1992
004.6'8—dc20 91-46677
CIP

1 2 3 4 5 6 7 8 9 0 DOC/DOC 9 8 7 6 5 4 3 2

ISBN 0-07-025234-3

The sponsoring editor for this book was Daniel A. Gonneau, the editing supervisor was Martha W. Gleason, and the production supervisor was Suzanne W. Babeuf. This book was set in Century Schoolbook. It was composed by North Market Street Graphics.

Printed and bound by R. R. Donnelley & Sons Company.

Contents

Foreword ix
Preface xiii

Chapter 1. Introduction to Local Area Networks (LANs) 1

1.1 The Need for Local Area Networks 1
1.2 Space Division Multiplexing 2
1.3 Sharing the Network Cable 4
1.4 Central Controller 6
1.5 Cables 7
1.6 The Node Board 8
1.7 A Quick Review of the ISO Seven Layers Model 9
1.8 Equating the LAN Protocol to the ISO Seven Layers Model 11
1.9 The IEEE 802 LAN Protocols 14
1.10 The Various LANs in the IEEE 802.3 LAN Protocols 15
1.11 CSMA/CD Access Method 17
1.12 A Quick Review of Manchester Format 18
1.13 The Frame 19
1.14 The Address of a Node 21
1.15 Principle of Operation of an 802.3 LAN 22
1.16 Reception of Frames by the Node 22
1.17 Transmission of Frames by the Node 24
1.18 How Could a Collision Occur? 26
1.19 The Different Sections of the 802.3 Node Board 27
1.20 Summary 30

Chapter 2. The Principle of 1BASE5 (Starlan) 31

2.1 Introduction to Starlan 31
2.2 Connecting Node Boards to the Hub Board 32
2.3 Upstream and Downstream Sections of the Hub 37
2.4 The Header Hub (HHUB) 40
2.5 Principle of Operation of Starlan 40
2.6 The Telephone Wires and Starlan 42

2.7 Using Existing Telephone Wiring for Starlan 44
2.8 Summary 45

Chapter 3. The Transmitter and Receiver of a 1BASE5 (Starlan) Node 47

3.1 Introduction 47
3.2 The Transmit Control Circuit 50
3.3 The Receive and Receive Control Sections 54
3.4 Squelching the Incoming Signal 56
3.5 The Receive Control Circuit 57
3.6 Summary 60

Chapter 4. The LAN Manager Section of a Node in 1BASE5 LAN 61

4.1 The Manchester Decoder-Encoder and the Carrier Sense Sections 61
4.2 A Quick Review of the Intel 80188 Processor 63
4.3 Connecting ROM and RAM Chips to the 80188 65
4.4 A Quick Review of the On-Chip DMA Controllers of the 80188 66
4.5 The Rest of the 80188 Pins 67
4.6 The High Integration Mode and High Speed Mode of the 82588 68
4.7 The 82588-80188 interface 69
4.8 The Serial Interface Port of the 82588 71
4.9 Summary 74

Chapter 5. The Software of the 1BASE5 Node Board 75

5.1 Transmitting a Frame 75
5.2 Preparing Data for Transmission in the RAM Chips 76
5.3 Storing the Received Data in RAM Chips 77
5.4 The Two Status Bytes Attached to the Data 79
5.5 The Amount of RAM Buffer Required for Storing Incoming Frames 81
5.6 The Command Register of the 82588 84
5.7 The Programming Model of the 82588 85
5.8 The FIX POINTER and RELEASE POINTER Commands 87
5.9 Commands That Utilize the Service of the DMA Controller of the 80188 89
5.10 The Channel Bit of the Command Byte 90
5.11 Updating the Status Registers 91
5.12 The Program of the 80188 92
5.13 Summary 92

Chapter 6. 10BASE5 (Ethernet) and 10BASE2 (Cheapernet) 93

6.1 The Topology and Components of Ethernet 93
6.2 The Topology and Components of Cheapernet 95
6.3 The Transceiver Circuit in Ethernet 98
6.4 Supplying Power to the Transceiver Board 101
6.5 Transceiver Chips 102
6.6 The Transceiver Circuit in Cheapernet 103
6.7 The Node Board in Ethernet and Cheapernet 103
6.8 The Ethernet Serial Interface Chip 105

6.9 Receiving Frames from the Transceiver 107
6.10 The Clock of the 82501 108
6.11 Informing the 82586 about Collision 109
6.12 The Loop-Back Concept 110
6.13 The Shared Memory Concept 111
6.14 The Operation 113
6.15 Writing the Program for the 80186 114
6.16 Command Example 115
6.17 The High Performance 32-Bit LAN Coprocessors 119
6.18 Summary 121

Chapter 7. Twisted Pair Ethernet 123

7.1 The Need for an Additional 802.3 LAN 123
7.2 Connecting Nodes to the Multiport Repeater Board 124
7.3 The Medium Dependent Interface Connector 128
7.4 The Node Board in a Twisted Pair Ethernet 130
7.5 Design of a Twisted Pair Ethernet Node Board with an Analog Front End 130
7.6 The Analog Front End 132
7.7 The Transceiver Serial Interface 134
7.8 A Twisted Pair Ethernet Serial Supercomponent 135
7.9 No Change to the Existing Software of an Ethernet Node 136
7.10 The Multiport Repeater (MPR) 136
7.11 Implementing a Multiport Repeater Board 140
7.12 Implementing the Twisted Pair MAU 140
7.13 The 82503 141
7.14 Summary 144

Chapter 8. Serial Communication 145

8.1 Connecting LANs with WANs 145
8.2 Serial Communication Circuitry 146
8.3 Modems 147
8.4 USART 148
8.5 USART-Processor Interface, Hardware Design 149
8.6 The Data Buffer Block of the 8251A 150
8.7 The Control Logic Block of the USART 152
8.8 Protocols 155
8.9 Transmitter, Hardware Design 159
8.10 Receiver, Hardware Design 161
8.11 Synchronous Modems and Asynchronous Modems 162
8.12 The SYNDET/BRKDET Pin of the 8251A 165
8.13 Software Implementation 167
8.14 Sending the Mode Instruction to the 8251A 170
8.15 Sending Instructions to the 8251A 172
8.16 Monitoring the Operation of the 8251A 175
8.17 The Read Status Byte 177
8.18 Fetching Received Characters from the 8251A 177
8.19 Writing Characters to Be Transmitted to the 8251A 178

Appendix A 183

A.1 Choosing the Right Network 183
A.2 The Line Integrity Test Feature of the 10BASE-T 183
A.3 Minimally Configured Station 184
A.4 The File Server 184
A.5 Intelligent Repeaters 186
A.6 Combining Several Networks 187
A.7 Node Drivers 188
A.8 Interfacing to the I/O of the PC 189

Appendix B. 82586 1EEE 802.3 Ethernet LAN Coprocessor 193

Appendix C. 82596CA High-Performance 32-Bit LAN Coprocessor 231

Appendix D. 82588 High-Integration LAN Controller 305

Glossary 333
Index 343

Foreword

In the old days, when I owned a 286 microprocessor-based personal computer, I used my PC to do two things, word processing and spreadsheet number crunching. Some of my colleagues got more adventurous and ran database applications as well. I was one of the lucky ones in our department in that I had my own dedicated dot matrix printer. Do you remember the times when people used to bring you their floppy disks so that you could print out a letter or a table for them?

Today's PC

Today I have an Intel 486SX* microprocessor-based personal computer. Under Microsoft Windows† 3.0, I run numerous applications programs simultaneously. I have added a graphics presentation program to my original word processor and spreadsheet. I no longer have my own dedicated dot-matrix printer; instead I share a number of printers with other people in my office. Since I am in marketing, I generate numerous presentations. Having the capability to generate and print presentation-ready black and white or color slides right from my PC saves me a great deal of time and money. Today, I print impressive presentations via an Ethernet‡ local area network on a Hewlett-Packard laser printer, or even a Tektronix color printer.

Two years ago I was introduced to electronic mail, and have been addicted ever since. Twice a day I *read* my *inbox, send* a *reply, forward* messages, *create* reports or memos, and *copy* them to other email users around the globe. Unlike faxes, email transmission is almost instantaneous, error-free and 100 percent reliable. Since most of the people I deal with are on email, I hardly ever use a fax machine these days.

---

*Intel 486SX and Intel 386 are trademarks of Intel Corporation.

†Windows is a trademark of Microsoft Corporation.

‡Ethernet is a trademark of Xerox Corporation.

Local area networks now permeate over one-third of corporations around the world, allowing companies to share expensive peripheral devices. "Sneakernet" (walking a floppy disk from one PC to another to transport data) is a thing of the past. Electronic mail is becoming the third power application that every user wants to have, made possible by local and wide area networks.

Bob Metcalfe, founder of 3Com Corporation, co-invented Ethernet at Xerox's Parc laboratories in the early 1970s. Digital Equipment, Xerox, and Intel commercialized Ethernet in the early 1980s. Today, Ethernet is *the* local area networking standard, with over 50 percent market share. It is a multibillion dollar business, with 3 million personal computer adapter cards alone being sold in 1991. The local area networking industry continues to grow faster than the personal computer industry as a whole; after all, there are over 50 million PCs still to be networked!

A Look into the Future

In 1981, after the original IBM PC was announced, a number of smaller companies started selling expensive serial and parallel port adapters. In the middle 1980s, both serial and parallel port add-in cards became very affordable. A few years later, personal computer manufacturers started integrating both ports into the PC. Today, Intel's newest microprocessor, the high-integration Intel 386 SL microprocessor superset, contains two serial and one parallel port *on-chip*. As future transistors become smaller, cheaper, and more powerful, more I/O functions will be integrated into the CPU. By the turn of the century, it is very likely that Intel's high-integration microprocessors will integrate Ethernet directly into the CPU.

Ethernet adapter cards will constitute the lion's share of the LAN market for quite a number of years. However, as the need for networking grows, personal computer manufacturers will start integrating the Ethernet chipset directly onto the personal computer motherboard. Today, in late 1991, the first network-ready PCs are starting to appear. In 1992, many more Ethernet-ready PCs will be introduced.

Intel has sold more than 9 million Ethernet VLSI (very large scale integration) components in the last 8 years, with 2 million more being forecast for 1991 alone. During those 8 years, Intel has introduced more than a dozen Ethernet components, two new chips in 1991 alone. As one of the technological leaders in the personal computer connectivity business, we will continue to bring out leading-edge connectivity components in the future too.

Ethernet is not an easy technology. I am positive that this book will go a long way in helping design engineers build better adapter cards or PC motherboards, so that the next time around I can send you the foreword via video-mail.

ROBERT BREYER
Product Marketing Engineer
Intel Corporation

Preface

Local area network (LAN) equipment includes products that are procured by virtually all segments of industry: small, medium, and large corporations; manufacturers; law and accounting firms; schools and universities; banks; mail order businesses; and other organizations that require establishing data communication between several personal computers (PCs). The number of PCs connected via a LAN varies as well. Some networks consist of three to four PCs connected in small offices to form a tiny LAN for the purpose of sharing a laser printer, databases, and other resources. Other networks consist of hundreds of stations.

To be able to design LAN equipment, the designer must fully understand the principle of operation of the equipment. Chapter 1 introduces the terms and definitions and explains how LANs work. The chapter discusses the difference between the probabilistic and deterministic type of LANs, the ISO seven layers model, and the various IEEE 802 LAN protocols. It includes an elaborate discussion of the CSMA/CD access method, Manchester decoding-encoding, the 802.3 frame, the collision concept, and other topics that are prerequisites to understanding the material presented in subsequent chapters. Since Chap. 1 presents the theory of local area networks, it serves as the base for all subsequent chapters.

To provide the reader with a full understanding of LAN design problems and their solutions, the book presents LAN design circuitries in the same order as they occurred historically. The reason for this particular approach is not to teach the history of LAN, but rather to present the reader with the various stages that the art of LAN design experienced. By following this route, the reader will gain insight into current design challenges as well as future design challenges that are involved in LAN equipment design.

One of the earlier LANs was the Starlan. This LAN is at least 10 times slower than current LANs and is becoming obsolete. Yet, by

studying this particular LAN, the reader will gain a full understanding of the degree of complexity (hardware and software) that is required in LAN implementation: the off-the-shelf LAN controller chip, the required magnetic components, the logic "glue" chips, and the various cable considerations and LAN topology. Chapter 2 discusses the topology and cable requirements of the Starlan, Chaps. 3 and 4 discuss the hardware node design, and Chap. 5 discusses the software of the implementation of the node. Although the Starlan itself became obsolete, the reader will find that most of the information presented in these chapters is still valid for current LANs.

The next stage of LAN evolution was the introduction of Ethernet and Cheapernet. These LANs utilize different LAN topologies and different types of cables. More important, these LANs are 10 times faster than the Starlan. Chapter 6 is devoted to the implementation of these LANs by using the next generation LAN controller chip, the Intel 82586 LAN controller. The 82586 is a more complex chip that requires more programming than the 82588. However, the experience gained by studying the 82588 will enable an easy transition to the more advanced 82586 LAN controller, since at that point, the reader already understands the performance requirements and what is expected from a LAN controller. The last section of Chap. 6 includes a discussion of the most up-to-date LAN controllers. The Intel 82596 LAN controller is currently the most advanced controller available, and its advanced performances should be obvious to the reader who masters the 82586.

Chapter 7 presents the next stage of LAN evolution, the most current 802.3 LAN protocol (the newly announced 10BASE-T LAN protocol). This LAN is as fast as the Ethernet/Cheapernet, and while its main component (the LAN controller) is the same controller as the one used by the Ethernet and Cheapernet (e.g., 82586, 82596), its topology and the types of cables that it uses resemble the Starlan. Chapter 7 discusses the various off-the-shelf 10BASE-T components, including the Intel 82503. Finally, Chap. 8 discusses serial communication.

The field of designing and manufacturing LAN equipment is dynamic and depends on several factors. One factor that determines the destiny and direction of LANs is the consumers who require inexpensive, reliable equipment. The other factor is the constant improvement of integrated circuits technology. As this technology keeps progressing, the integrated circuits manufacturers are able to announce new chips which may be used as the building blocks for future LAN equipment that will be able to reliably transfer data at higher data rates. Indeed, in many cases it is the integrated circuits manufacturers who introduce and contribute ideas to new LAN protocols.

An example of the dynamic nature of the LAN market is the introduction of the 10BASE-T LAN, the latest IEEE 802.3 protocol. This

new protocol is expected to replace the popular 10BASE5/10BASE2 LANs. It contains all the "good" features of these LANs. That is, it uses the same CSMA/CD access method which proved itself to be very reliable during the last several years, and it has the same high data rate capability (10 Mbit/s). Yet this new protocol introduces an improvement: it does not require coaxial cables, but rather it uses inexpensive twisted pairs of wires.

The purpose of this book is to introduce hardware and firmware design know-how of local area network circuits to engineers, technicians, students, and anybody else who is interested in gaining such knowledge. Although the book includes a discussion of all the current IEEE 802.X LAN protocols, the book specifically concentrates on the CSMA/CD type of LANs: the IEEE 802.3 protocols: 1BASE5 (Starlan*), 10BASE5 (Ethernet†), 10BASE2 (Cheapernet), and the newly evolved 10BASE-T (Twisted Pair Ethernet). The book teaches the theory and principle of operation of these LANs and then illustrates how to implement a design of such LAN circuits (hardware and firmware) by using off-the-shelf reliable components.

NATHAN GUREWICH

*Starlan is a registered trademark of AT&T.

†Ethernet is a registered trademark of Xerox Corporation.

Communication Systems

Chapter

1

Introduction to Local Area Networks

1.1 The Need for Local Area Networks

As the number of personal computers (PCs) utilized in various industries increases, there is a growing demand for establishing data communication between PCs by connecting them together.

A *network* is the tool by which the PCs are connected. The network consists of hardware components (boards, cables, connectors, etc.) and software components.

The network enables a user of one PC in the network to transfer data to any other PC in the network. When the network cables are limited to short distances, the network is called a *local area network,* or *LAN* for short. Among other things, the LAN protocol specifies the maximum allowed cable length and the type of cables to be used.

The major reasons for demanding the ability to connect several PCs together are

- *Electronic transmission.* By networking several PCs together, there is no need to transfer data from one PC to another by physically transferring diskettes. Rather, the data can be transmitted electronically over the network cables.
- *Sharing peripherals.* By networking several PCs together, costly peripheral devices such as plotters can be shared among several users.

1.1.1 Types of LANs

There are many types of LANs, each suitable for a particular type of application. Some LANs are oriented toward noisy environments. In

these LANs, the prime concern is to be able to establish communication despite the large noise generated by the environment. Such LANs are used, for example, in manufacturing environments. In other LAN applications, the ability to transfer data as fast as possible is the prime concern. Yet in other applications, the prime concern is the cost of the LAN.

1.1.2 The need for standard protocols

As the demand for LANs increased, various industries realized the need of establishing standard protocols for LANs.

A standard protocol is a set of rules and specifications that fully defines and specifies the LAN. A standard LAN is one that is built from several components, where each component is assumed to comply with the specifications mentioned in the standard protocol.

By having a standard protocol, the end user is able to construct a LAN by purchasing different components of the LAN from different vendors based on cost, service, and product reliability. The network should not experience any problem when using components from different vendors, as long as the LAN components were designed in strict compliance with the protocol. It is therefore important to design the LAN equipment in accordance with the protocol.

Another important advantage of having standard protocols for LANs is that once a standard protocol is accepted and recognized, there is a large demand by LAN designers for chips that are capable of implementing portions of the hardware and software tasks dictated by the protocol. As the demand for these chips increases, chip manufacturers have the incentive to invest in the design, manufacturing, and marketing of these chips, since the expected sales justify the investment. The availability of inexpensive LAN chips to the LAN designers reduces development time and yields more reliable products.

In many cases, the chip manufacturers are the major contributors for drafting the specifications of the standard protocols. Of course, having the integrated circuit (IC) manufacturers contribute their ideas to the standard protocols makes sense, since they are the ones who know best what can and cannot be integrated on silicon chips.

1.2 Space Division Multiplexing

One method of connecting several PCs together to form a network is shown in Fig. 1.1. As shown, each PC has a cable connecting it to all the other PCs in the network. This method is called *space division multiplexing* since the space is divided so that each portion of the space carries a cable.

One of the major drawbacks of the space division multiplexing technique is the fact that it requires too many cables compared to other

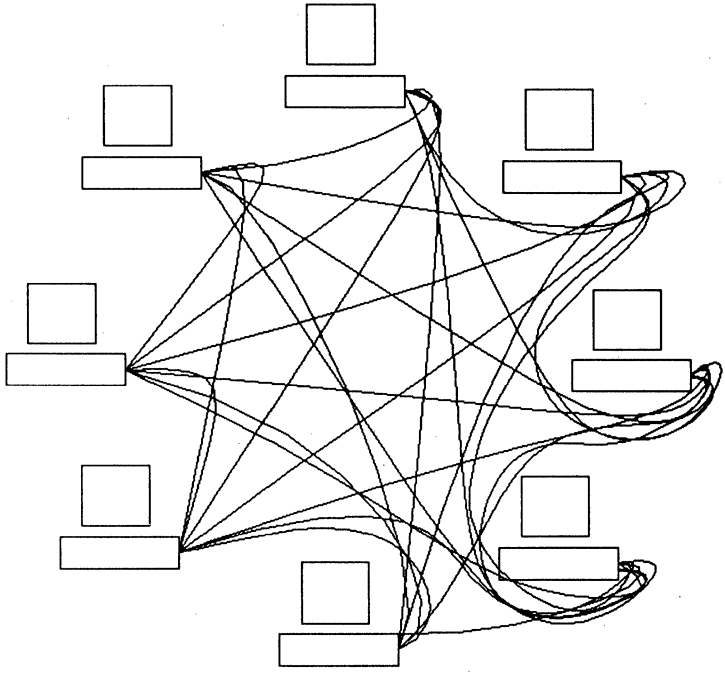


Figure 1.1 Space division multiplexing.

available multiplexing techniques. Stretching cables between rooms, floors, and buildings is expensive. Major contributors to the incurred cost are the cost of the network cables and the cost of the labor to install the cables. Another drawback of the space division multiplexing method is the lack of flexibility. Adding a new PC to the network requires the installation of cables from the new added PC to all the rest of the PCs in the network.

Example 1.1 Assume that 100 PCs are to be networked using the same technique shown in Fig. 1.1. How many cables are needed?

Solution When N is the number of PCs to be connected using the technique shown in Fig. 1.1, the total number of cables required is

$$\text{Total number of cables required} = \frac{N \times (N-1)}{2}$$

For 100 PCs,

$$\text{Number of cables required} = \frac{100 \times 99}{2} = 4950 \text{ cables}$$

Obviously, a better technique for connecting PCs is needed.

1.3 Sharing the Network Cable

A less expensive method that uses fewer cables is shown in Fig. 1.2. In this technique, a single cable is used to connect all the PCs. While in the network shown in Fig. 1.1 a few PCs may converse with each other simultaneously, in the network shown in Fig. 1.2 only one PC is permitted to transfer data at any given time. The network cable is shared by all the PCs and may carry a single transmission at any given time. As might be expected, to connect a PC to the network cable, some type of connection board is required. In Fig. 1.2, this board is denoted as the *interface board.*

Figure 1.3 is another example of a network in which only a single PC is permitted to transmit at any given time. Again, as might be expected, the cables are not just tied up at the center, but rather an electronic board called a *hub board* is required.

1.3.1 LAN topologies

Different LANs have different topologies. The LAN topology defines the way by which the PCs are connected together. Examples of some of

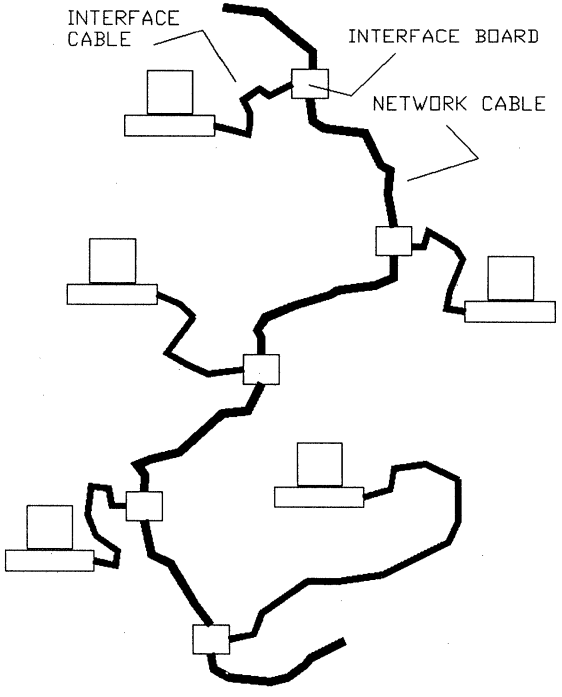


Figure 1.2 Sharing the cable, bus topology.

the common topologies are the *bus topology* shown in Fig. 1.2 and the *star topology* shown in Fig. 1.3. The LAN topology is specified in the LAN protocol.

1.3.2 Access method, deferring procedure, and collision

When the PCs in a LAN share a single cable, certain sharing rules and procedures must be established in order to avoid chaos on the cable.

When a PC wishes to access the network cable for the purpose of transferring data to another PC, it must verify first that the network cable is free. This access verification process is called the *access method.* In some LANs, each of the transmitting PCs is responsible for finding whether or not the network cable is free. In other LANs, the PC is told by a special LAN circuitry called a *central controller,* whether or not the network is free.

If the PC concludes that the network is not free, it defers its transmission for a later time. Deferring the transmission for a later time is called the *deferring procedure.*

If it happens that more than one PC is transmitting at the same time, the network is in a *collision.* In a collision, the network cable carries a signal that is a mixture of data from more than one PC. This signal is useless, and the PCs in the network should be able to recognize such a signal and ignore it.

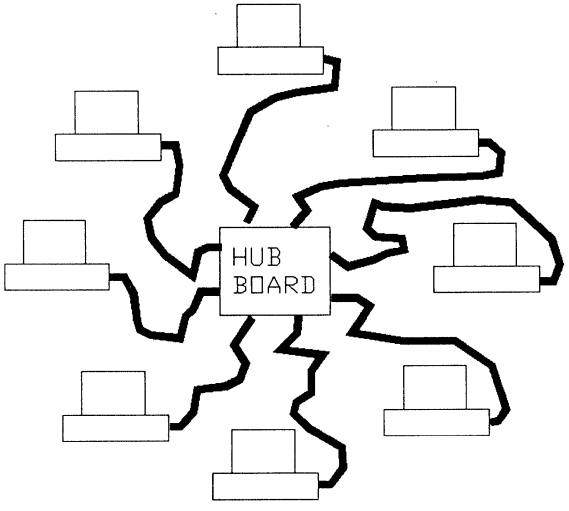


Figure 1.3 Sharing the cable, star topology.

Although the access method is equipped with enough access rules to avoid collisions, collisions do occur. The LAN protocol specifies how to detect collisions and what to do in the event a collision is detected.

1.4 Central Controller

Some LANs are designed to operate with a central controller, while other LANs are designed to operate without a central controller. A central controller is an intelligent electronic circuit that has the responsibility of managing the whole network. A PC in a LAN that utilizes a central controller is not permitted to transmit unless granted a permission to do so by the central controller. Once a PC is granted the permission to transmit, it may transmit only for the amount of time dictated by the central controller.

The advantage of using a central controller in a LAN is that the central processor can be programmed to favor a group of PCs in the network, granting these PCs more permissions to transmit than other PCs in the network.

A LAN that utilizes a central controller is categorized as a *deterministic* type, since the central controller can be programmed in such a way that it is possible to determine in advance which PC is allowed to transmit, for how long, and at what sequence.

The major disadvantages of a using central controller in a LAN are

- If the central controller circuit fails, the LAN is useless, since this type of LAN cannot operate without its central controller.
- Adding and deleting PCs from the network requires reprogramming the central controller.
- A LAN that utilizes a central controller has the additional cost of the central controller equipment.

1.4.1 Modulation

Some LANs utilize the *frequency division modulation* technique. This technique is similar in concept to the technique used in cable television, where simultaneous transmissions over a single network cable are permitted. Each transmission is assigned to a different frequency channel. This technique requires the use of both transmitter circuits that modulate the data to the required frequency channel and receiver circuits that demodulate the data back to the baseband (original) frequency of the data.

Example 1.2 In a baseband transmission the entire bandwidth of the network cable is dedicated for a single transmission.

1.5 Cables

There are various types of cables used in LANs. The most common types of cables used in LANs are

- *Coaxial cables.* These are well-shielded cables, designed to carry data at rate of 10 Mbit/s for distances ranging from 200 to 500 m without distorting the data. These cables are available as single-shielded or as double-shielded cables. High-quality coaxial cables are expensive.
- *Unshielded twisted pair.* This is the type of wire used by telephone companies to install telephone services. These wires are very inexpensive, probably the most inexpensive wires that one can purchase. These cables are purchased in *bundles;* typically each bundle is composed of 25 pairs of twisted wires.
- *Fiber optics.* Currently, these cables are expensive since there is not yet a great demand for them. As demand increases, prices are expected to decrease.

Example 1.3 Explain the principle of transmitting and receiving data by using fiber optic cables.

Solution The principle of transmitting and receiving data by using fiber-optic cables is illustrated in Fig. 1.4. The electronic switch shown in the figure is opened and closed in accordance with the data to be transmitted.

When a 0 is to be transmitted, that 0 causes the switch to open and current flows through the LED, causing it to emit light into the fiber cable.

When a 1 is to be transmitted, that 1 causes the switch to close (shorting out the LED); thus no current is flowing through the LED, and therefore no light is emitted to the fiber cable.

Thus, a flow of light through the cable represents a 0, and the absence of light represents a 1. The light is then propagated through the cable. The cable is flexible and can be routed from one place to another easily. The amazing characteristic of the fiber cable is that light is transmitted from one end of the cable to the

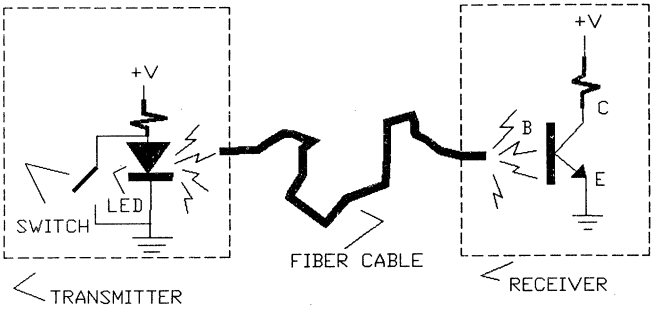


Figure 1.4 The principle of fiber optics transmission.

other end even if the cable is not positioned in a straight line. However, being made of glass or plastic, too much bending of the cable would break it.

Being a light, the transmission has the following major advantages:

- *No interference.* The transmission of the light causes no interference to nearby cables and nearby electronic instruments. Also, external electrical noise cannot distort the transmission of the light.
- *No distortion due to cable characteristics.* Since the data are transferred as light, the transmission is not affected by the resistance, capacitance, and inductance of the cable.
- *High speed of transmission.* The data are traveling at the highest speed that one could hope for, the speed of light.

On the other end of the fiber cable, there is an optical transistor. The fiber cable emits the light into the base of the transistor.

The optical transistor is turned ON when light is applied to its base, causing a low voltage at its collector, which is then interpreted as 0. When light is not applied to its base, the transistor is OFF and the collector is at high voltage, which is then interpreted as 1.

The limiting factor for the speed at which data can be transmitted is the ability of the LED and the optical transistor to switch on and off. A typical rate of transmission is 100 Mbit/s.

1.6 The Node Board

In order to connect PCs to a LAN, electronic boards called *node boards* are required. In many cases, the node boards are designed to be plugged into one of the I/O slots of the motherboard of the PC. A simplified logic block diagram of a node board is shown in Fig. 1.5. The node board is responsible for accepting data from the PC and transmitting the data to the network via its transmitter. The node board is also responsible for receiving data from the network via its receiver and transferring the received data to its PC.

The device connected to the node board is called *data terminal equipment,* or DTE for short. The DTE may be a PC (as shown in Fig. 1.5) or any other device. No matter what type of DTE is connected to the node and no matter how the DTE is connected to the node, the node always communicates with the network in exactly the same way, as prescribed in the network protocol. The node board shown in Fig. 1.6 is shown to be connected to the serial port (RS-232) of the PC. Nevertheless, the nodes of Figs. 1.5 and 1.6 are identical as far as the network is concerned. Of course, the node must contain the appropriate interface circuitry that enables the interface between the node and its DTE.

1.6.1 Connecting a node to the network

The network protocol completely defines and specifies the cables that connect the node to the network. Included within the specification are

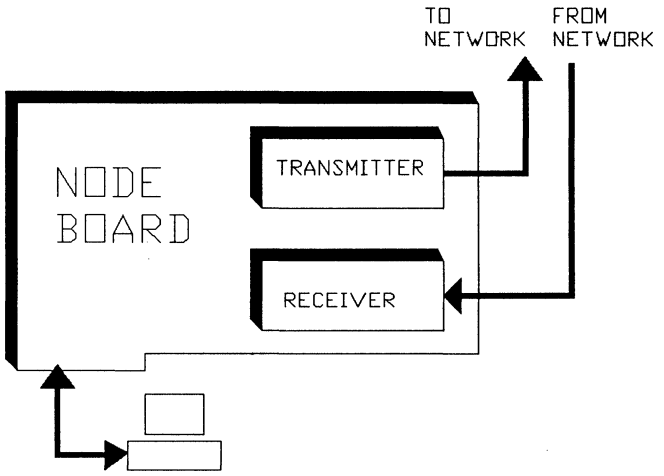


Figure 1.5 Connecting a node board to a DTE.

the number of wires in the cable, the type of wires, and the maximum length of the wires.

A DTE together with its node is called a *station.*

1.7 A Quick Review of the ISO Seven Layers Model

ISO is the abbreviation of *International Standards Organization.* This organization took upon itself the tasks of developing and publishing various standards which are accepted worldwide. The organization has representatives from most of the countries in the world, and it is an agency of the United Nations.

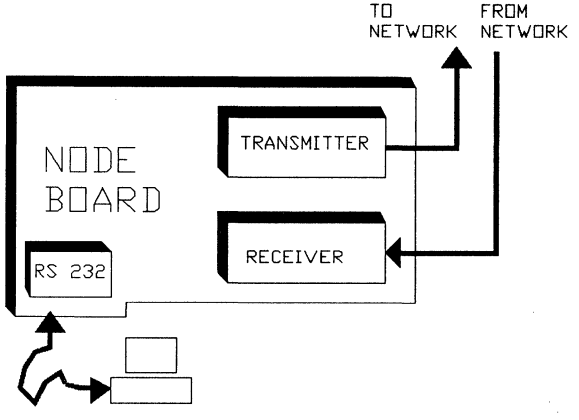


Figure 1.6 Connecting a node board to a DTE via RS 232.

The *seven layers model,* which was adapted by the ISO, is a recommended model to be used when developing and implementing networks. The concept of the model is to divide the network into seven distinct sections (layers), as follows:

Layer 1: Physical layer

Layer 2: Data link layer

Layer 3: Network layer

Layer 4: Transport layer

Layer 5: Session layer

Layer 6: Presentation layer

Layer 7: Application layer

The ISO seven layers model is recommended for the implementation not only of LANs but also of wide area networks (networks that utilize the telephone lines), as well as other types of communication networks. However, our discussion is restricted to the relations between the ISO model and LANs.

The seven layers are implemented in hardware and software and are installed on both sides of the communication link, the transmitter side, and the receiver side. Figure 1.7 is a pictorial representation of the model.

Each layer is responsible for performing a specific task in the network. When the transmitter wishes to transmit, the process starts at layer 7. Layer 7 prepares the data to be transmitted, and once layer 7 completes its task, it transfers the data to layer 6. Layer 6 also "works" on the data and passes the data to layer 5. The data keep propagating

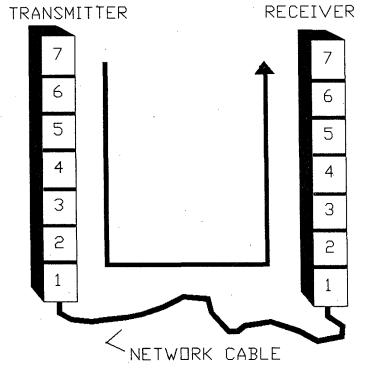


Figure 1.7 The ISO seven layers model.

downward, until thus reach layer 1, the physical layer. Layer 1 is the layer that actually transmits the data. By the time the data reach layer 1, there are several control bytes attached to data. These control bytes are added by the different layers.

On the receiver side, data are being received by layer 1, and then passed to layer 2. Layer 2 analyzes the control bytes that are related to the task assigned to it, strips off these control bytes, and passes the data upward to the next upper layer. Data keep propagating upward and eventually reach layer 7.

Example 1.4 As an example of one of the control bytes that is added to the data by the layers, consider the case where a large data file is to be transmitted.

In LANs, only one PC is allowed to transmit at any given time. To avoid monopolization of the network cable by a single PC, long transmissions are not allowed.

The transmitting PC must divide the large data file into several small segments. This breaking of the data files into several small segments is a task performed by one of the layers on the transmitter side. That layer breaks the large file into several small segments and attaches control bytes to each of the segments indicating the sequential segment number. Each segment propagates downward, eventually reaches layer 1, and is transmitted separately.

One of the layers on the receiver side has the task of reassembling the segments (in the right sequence) to reconstruct the original large data file. That layer is able to do so, since the segment numbers are attached to the segments as control bytes. If, for some reason, one of the segments does not reach the receiver, the layer has the ability to detect that it is missing a segment by simply keeping track of the segment numbers.

The advantage of developing a network that follows the ISO seven layers model is the ability to upgrade, change, or modify any of the layers without the need to change or modify the other layers. That is, a layer can be "unplugged" from the network, modified, and then "plugged" back to the network.

The tasks assigned to each of the seven layers are based on experience that was accumulated by a variety of network designers through the years.

1.8 Equating the LAN Protocol to the ISO Seven Layers Model

Figure 1.8 illustrates the seven layers of a LAN protocol as viewed by the Institute of Electrical and Electronics Engineers (IEEE). This organization, just like the ISO organization, took upon itself the tasks of developing and publishing standard protocols.

The committees that are responsible for developing LAN protocols are the 802 group of committees. The 802 group is divided into eight committees, each responsible for dealing with a different LAN subject.

As shown in Fig. 1.8, the IEEE deviates somehow from the original ISO seven layers model by further dividing layer 2 into two layers: the *logical link control* layer, or LLC for short, and the *media access control* layer, or MAC for short.

1.8.1 Layer 1, the physical layer

Layer 1, the physical layer, has the task of defining and implementing the physical connection. Included in this layer protocol are the specifications for the type of cable, connector specifications, and everything else that is directly related to the physical connection.

1.8.2 The media access control (MAC) layer

The media access control (MAC) layer at the transmitter side has the task of receiving data from a higher level layer (i.e., the LLC layer) in accordance with an agreed upon set of rules. The MAC layer is then responsible for executing the access method, deciding whether the network is free and ready for its transmission. Once this layer decides that it is time to transmit, it adds certain control bytes to the data, formats the data to a certain format (called a *frame*), and hands over the data to the physical layer for transmission.

On the receiver side, the MAC layer receives data from layer 1, checks that the control bytes of the data (which were attached by the MAC layer on the transmitter side) conform to the set of rules, and decides whether or not the data were corrupted during its transmis-

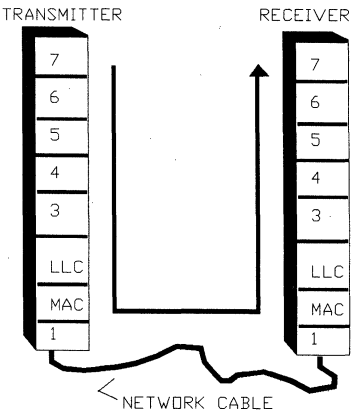


Figure 1.8 Equating the IEEE 800 LAN models to the ISO model.

sion due to noise and other misfortunes. The layer then removes the control bytes from the data and transfers the data (without the control bytes) to the next higher layer level, the LLC layer.

The tasks assigned to the MAC at the transmitter side are

- Receiving data from its LLC layer.
- Making a decision when to transmit (executing the access method).
- Formatting the data and attaching control bytes (constructing a frame).
- Handing over the frame to layer 1 for transmission.

The tasks assigned to the MAC at the receiver side are

- Receiving the data from layer 1.
- Examining the control bytes for the purpose of making the decision whether the data were corrupted on their way.
- Stripping off the control bytes and handing over the data upward to the next higher level.

Example 1.5 The commercial LAN software known as Netware (manufactured by Novell) is a software package that implements the LLC layer, layer 3, layer 4, and layer 5.

The following examples illustrate how the seven layer model distributes different network tasks among the different layers.

Example 1.6 Suppose that a PC wishes to receive a data file from another PC in the network. The request is prepared by the upper layers and passed to the MAC layer for transmission. The MAC layer constructs a frame, transmits the frame, and reports back to the upper layers that the transmission was successful.

After a few minutes, the PC program notices that the requested file still has not been received. Which group of layers is responsible for retransmitting the request?

Solution The upper layers are responsible for generating the request again. The MAC layer has no means to examine the intelligent content of the data that it is transmitting to the network.

By reporting back to the upper layers that the transmission was successful, the MAC layer merely indicates that no collision was detected during the transmission.

Example 1.7 Suppose that the MAC layer receives data (from its upper layers) for transmission. The MAC layer constructs a frame and transmits the data. However, at the middle of the transmission, the MAC layer discovers that it is transmitting together with another PC in the network at the same time (i.e., the MAC layer is transmitting into a collide network cable).

MAC protocols require the MAC layer to automatically retransmit the frame again. The MAC protocol also specifies that if the MAC layer keeps retransmitting

the frame and keeps encountering collisions one time after the other, the MAC layer should abort the transmission and report the situation back to its upper layers. The MAC protocol specifies the number of consecutive collisions that must occur before the MAC layer should give up on the transmission of that frame.

1.9 The IEEE 802 LAN Protocols

Figure 1.9 illustrates the approach we take to implement layer 1 and the MAC layer. As shown, we choose to implement these layers in hardware and firmware on a separate board, the node board. The rest of the layers are implemented as software executed by the PC.

There are several IEEE standard LAN protocols available for implementing layer 1 and the MAC layer. These protocols are specified as a set of protocols called the IEEE 802 standard LAN protocols.

1.9.1 The IEEE 802.3 standard LAN protocols

The IEEE 802.3 is a set of LAN protocols for the implementation of layer 1 and the MAC layer. Each of the LANs specified in the 802.3 group has a different set of physical specifications (cable type, cable length, topology, etc.). However, all LANs within this group of protocols utilize the same access method, the CSMA/CD access method. The model representation for these LANs is shown in Fig. 1.10*a*.

1.9.2 The IEEE 802.4 standard LAN protocol

Figure 1.10*b* is a representation of a different LAN, the 802.4 LAN. This protocol specifies the *token bus* as the access method.

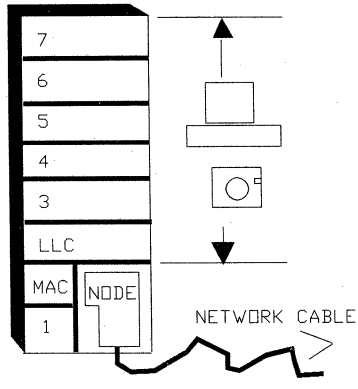


Figure 1.9 Realizing the seven layers model in hardware and software.

1.9.3 The IEEE 802.5 LAN protocol

Figure 1.10*c* is a representation of yet a different LAN, the 802.5 LAN. This protocol specifies the *token ring* as the access method.

1.9.4 The IEEE 802.6 LAN protocol

Another standard LAN protocol is the LAN that conforms to the IEEE 802.6 standard LAN protocol. This LAN is called the *metropolitan area network,* or MAN for short. Although we refer to this as a "LAN" protocol, this LAN may have a network fiber-optic cable that may be up to 50,000 m long.

1.9.5 The advantages of using the ISO seven-layers models

Figures 1.9 and 1.10 illustrate the advantage of using the ISO seven layers model. That is, no matter what node board (MAC layer plus layer 1) is used, the upper layers remain the same. Of course, proper interfacing must be incorporated in the design to allow the interface between the node board and its upper layers.

1.10 The Various LANs in the IEEE 802.3 LAN Protocols

As mentioned, the 802.3 protocol defines different LANs, each with a different set of specifications for the cable type, cable length, LAN

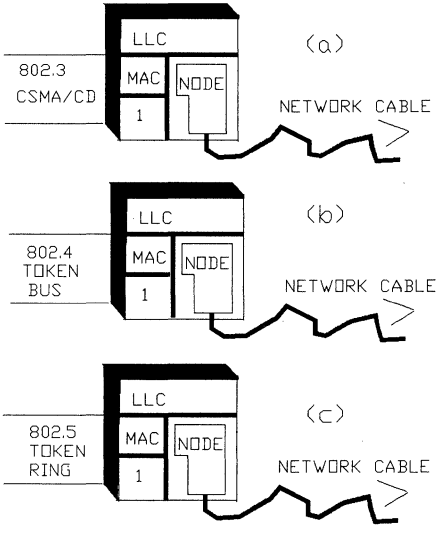


Figure 1.10 Models of the various IEEE 800 LAN protocols.

topology, and others. However, the access method for all the LANs in the 802.3 group is the same, the CSMA/CD access method.

1.10.1 The xTYPEy LAN name

Each of the LANs within the 802.3 group is identified by its designation name. The LANs have the following designation format:

```
xTYPEy Name
```

where x = the data rate in megabits per second (Mbit/s) at which data are transferred over the network cable.
TYPE = indicates the type of modulation used in the transmission of the data over the network cables.
y = indicates one of the major characteristics of the network cable.
Name = the name of the LAN.

Usually there is no space separating the "x" from the "TYPE" and no space separating the "TYPE" from the "y." The designation name is not intended to fully describe the particular LAN, but simply to indicate some of the special characteristics that make the LAN different from the other LANs in the group.

Example 1.8 The IEEE 802.3 10BASE5 (Ethernet) is the designation of a LAN with the following main characteristics:

- The "10" represents the fact that data are transmitted over the network cable at a rate of 10 Mbit/s.
- The "BASE" represents the fact that data are being transmitted in baseband transmission.
- The "5" represents 500 m, the maximum permitted length of network cable when the network is configured in a minimum configuration.

Note that Ethernet is a registered trademark of Xerox Corporation.

Example 1.9 The IEEE 802.3 1BASE5 (Starlan) is the designation of a LAN with the following main characteristics:

- The "1" represents the fact that data are transmitted over the network cable at a rate of 1 Mbit/s.
- The "BASE" represents the fact that data are being transmitted in baseband transmission.
- The "5" represents 500 m, the maximum permitted length of network cable when the network is configured in a minimum configuration.

Note that Starlan is a registered trademark of AT&T.

Example 1.10 The IEEE 802.3 10BASE2 Cheapernet is the designation for a LAN called Cheapernet:

- The "10" represents the fact that data are transmitted over the network cable at a rate of 10 Mbit/s.

- The "BASE" represents the fact that data are being transmitted in baseband transmission.
- The "2" represents 200 m, the maximum permitted length of network cable when the network is configured in a minimum configuration.

Example 1.11 The IEEE 802.3 10BROAD36 Broadband Ethernet is the designation for a LAN called Broadband Ethernet:

- The "10" represents the fact that data are transmitted over the network cable at a rate of 10 Mbit/s.
- The "BROAD" represents the fact that data are being transmitted as a broadband transmission.
- The "36" represents 3600 m, the maximum permitted length of network cable when the network is configured in a minimum configuration.

Example 1.12 The IEEE 802.3 10BASE-T is the designation for a LAN called 10BASE-T:

- The "10" represents the fact that data are transmitted over the network cable at a rate of 10 Mbit/s.
- The "BASE" represents the fact that data are being transmitted as a baseband transmission.
- The "T" represents the type of network cable, a twisted pair wire for this LAN.

1.11 CSMA/CD Access Method

The access method specified in the IEEE 802.3 protocols is called the CSMA/CD access method (Fig. 1.10*a*).

The "CS" in CSMA/CD is the abbreviation of *carrier sense.* While data are traveling over the network cable, the cable is said to contain a carrier. Each node must sense the cable prior to transmission and decide if the cable carries a carrier. If the node senses no carrier, it can start transmitting.

If, however, the node senses a carrier, the node starts to execute the *deferring* procedure. The deferring procedure is the procedure whereby the node waits for a certain amount of time and then senses the cable again. It is quite possible that on its next try, the node again finds a busy network, while another, luckier node might find a free network on its first attempt. The CSMA/CD access method protocol is a probabilistic type, since there is no guarantee of when a node might be able to access the network. Thus, the CSMA/CD access method does not implement any priority mechanism; every node in the network has the same chance to access the network.

The "MA" in CSMA/CD is the abbreviation of *multiple access.* This term refers to the fact that after completing its transmission, a node can sense the cable again, and if it finds a free network, it may transmit again. Thus, the CSMA/CD access method does not implement a fairness mechanism (i.e., it is not fair that a node accesses the cable numerous times while other nodes are still waiting for a free network).

Example 1.13 Other access methods specified in other protocols might not permit multiple access. In a non-multiple-access protocol, each node waits for its turn to transmit. Once a node completes its transmission, it waits for its next turn to transmit again. Thus, these access methods implement a fairness mechanism.

The "CD" in CSMA/CD is the abbreviation of *collision detect.* This term refers to the fact that the node must have the capability to detect collisions on the cable. That is, if it happens that more than one node is transmitting at the same time, the node must be able to detect that there are no valid data on the cable, but a "mixed" signal that was generated due to a collision. Naturally, the node should regard that signal as useless data. The 802.3 protocols specify an elaborately detailed list of actions that the node should take upon discovering a collision. We shall soon see what the actions are to be taken and how these actions are implemented in hardware and software by the node.

1.12 Quick Review of Manchester Format

The 802.3 protocols specify that data must be transmitted over the network cable in *Manchester format.* When a PC in the network wishes to send data to another PC in the network, it transfers the data to its node in a parallel form, byte after byte. The node serializes the data, processes the data, and sends the processed data in a Manchester format, bit after bit to the network. Thus, the data present on the wires of the network are in a Manchester format.

A *nonreturn to zero* signal, or NRZ for short, is a signal in which 1 is represented by a high voltage and a 0 is represented by a low voltage.

Figure 1.11 shows how an NRZ format can be converted to a Manchester format by using an exclusive OR gate. One of the inputs to the exclusive OR gate is the NRZ signal to be converted, the second input to the gate is a clock. The output of the gate is the Manchester signal. The clock is a 50 percent duty-cycle square wave, and its frequency is such that it completes a cycle during the duration of a single 0 or 1 of the NRZ signal.

Since the data have to be transmitted in Manchester format, the node must have the capability of converting NRZ (supplied to it by its PC) to a Manchester format. The node must also have the capability to convert a Manchester signal (supplied to it by the network) to an NRZ signal.

The Manchester format has the characteristic that no matter what the original NRZ signal was, the Manchester format always has a transition from 0 to 1 or from 1 to 0 at least every two half-cycles of the clock.

The advantage of using the Manchester format for transmission is that the receiving node can examine the incoming bits (which are coming in a Manchester format), and if it sees a Manchester code violation,

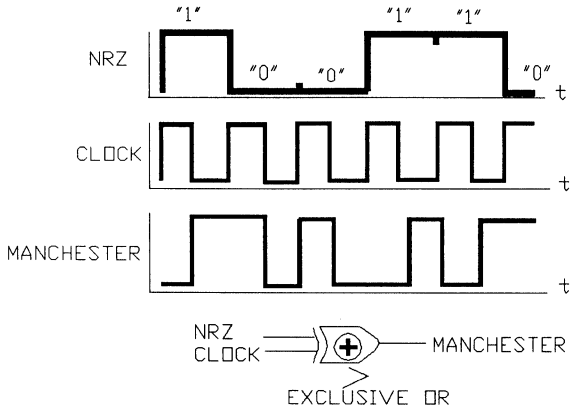


Figure 1.11 Generating Manchester signals.

it concludes that "something wrong" happened during the transmission. The code violation is easy enough to detect. If the incoming bits do not make a transition from 0 to 1 or from 1 to 0 during two half-cycles of the clock, there is a code violation.

Example 1.14 What method could be employed by the receiving node to convert the incoming Manchester data back to NRZ format?

Solution To convert the incoming Manchester format back to an NRZ format, an exclusive OR could be utilized again. That is, if one input to the gate is the same clock that is used to generate the Manchester code and the other input is the incoming Manchester signal, the output of the gate is the original NRZ signal. This can be verified from Figure 1.11 by exclusive-ORing the clock and the Manchester signals and obtaining the NRZ signal of Fig. 1.11.

Of course the trick is to be able to use the same exact clock that was used to generate the Manchester, which is the reason for also calling the process of converting a Manchester signal back to an NRZ signal the *clock recovery* process.

1.13 The Frame

Prior to transmission, the node formats the data in accordance with a specific frame format. The format of the frame in the IEEE 802.3 protocols is shown in Fig. 1.12. The first 7 bytes of the frame are called the *preamble field.* These 7 bytes consist of alternating 0s and 1s (101010 . . . 10). When a node in the network receives the preamble bytes, it knows that a frame is coming. The reason for having the preamble bits in the frame is to enable the node to prepare its clock recovery circuitry, which is the reason for also calling these the *training* bits.

The next field of the frame is a 1-byte *start of frame* field, which always contains the byte AB hex (10101011). This byte is an indication to the receiving node that following this field are actual useful data.

The next field of the frame is a 6-byte *destination address* field. During the setup of the network, each node is assigned an identification (ID) number. The destination address is the ID number of the node for which the frame is intended.

The next field in the frame is a 6-byte *source address* field. This field contains the ID number of the node that transmits the frame. Thus, the receiving node is able to tell who sent it the frame.

The next field of the frame is a 2-byte *length* field. This field contains the number of real data in the next field.

The next field of the frame is the *information* field. This field must have a minimum of 46 bytes and a maximum of 1500 bytes. It contains the data information. The whole objective of sending the frame is to send this field.

Following is a 4-byte CRC field. This field contains the CRC number that is generated based on the destination address field, source address field, length field, and information field. A CRC number is similar in concept to a parity bit. That is, the receiving node calculates the CRC of the data that it receives, and if it finds that the value is different from the CRC transmitted to it, it concludes that there is a transmission error. Unlike the parity bit, the CRC is more complicated to generate (requires more mathematical manipulations); however, it is unlikely (with regard to probability) that transmission errors occurred if the CRC calculated by the receiving node matches the CRC in the received frame.

1.13.1 Who is responsible for generating the fields of the frame?

Upon initialization, the PC assigns the node with the source address. Thereafter, the only fields that the node receives from the PC are the destination address field and the information field. All the rest of the frame fields are generated and inserted by the node.

As mentioned above, the information field is required to have a minimum of 46 bytes. If the PC instructs the node to transmit information that is less than 46 bytes, the node must insert dummy bytes to make the information field 46 bytes long. This process is called *padding*.

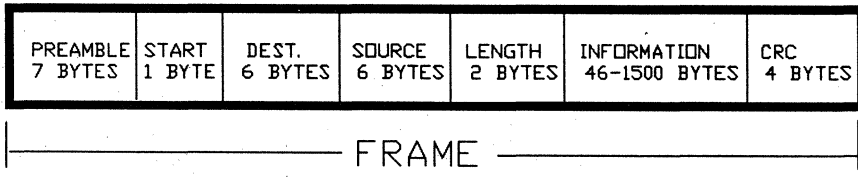


Figure 1.12 The 802.3 frame.

Example 1.15 Suppose that the PC instructs the node to transmit 150 bytes of information. What should the length field be?

Solution In this case, the length field should be 150, indicating that the information field contains 150 bytes of real data.

Example 1.16 Suppose that the PC instructs the node to transmit 40 bytes of information. What should the length field be?

Solution In this case, the length field should be 40, indicating that the information field contains 40 bytes of real data.

The transmitting node must pad the information field with 6 padding bytes to make the information field 46 bytes long. Upon receiving the frame, the receiving node examines the length field and realizes that real data are contained only within the first 40 bytes of the information field.

1.14 The Address of a Node

An *address* of a node is composed of 6 bytes. Upon booting up the node, the PC assigns its node with an address. Each node in the network is assigned a unique address.

In addition to assigning a unique address to the node, the PC could also assign a *group address* to its node. A group address is an address that is assigned to several nodes in the network.

When a node transmits data to another node, it inserts the unique address of the destination node in the destination address field of the frame. The frame is then transferred over the network cable and reaches all the nodes in the network. The destination address field enables the destination node to compare the content of the destination address field to the address that was assigned to it upon booting and to decide if the frame is intended for itself.

There are instances when a PC wishes to transmit the same data to all the other PCs in the network. In these cases, instead of transmitting the same frame separately time after time to each of the nodes in the network, the transmitting node sends the frame only once with a destination address equal to all 1s. When a node in the network sees an incoming frame that has all 1s in the destination address field, it realizes that this frame is intended for all the nodes in the network and therefore accepts the frame. This type of transmission is called *broadcasting.*

Another scenario that occurs frequently (and therefore is specified in the 802.3 protocols) is the case in which a PC wishes to send the same data to a special group of PCs in the network. Instead of transmitting the same frame separately time after time to each of the nodes of the group, the node transmits the frame only once with the destination address field containing the address of the group. A group address must have a 1 in the LSB of the address. When a node sees a 1 in the

LSB of an incoming frame, it realizes that the frame is intended to a group of nodes. The node then examines to see if the group address specified in the frame matches its own group address that was assigned to it upon booting. If so, it accepts the frame. This type of transmission is called *multicasting.*

1.15 Principle of Operation of an 802.3 LAN

We can now turn to Fig. 1.13 and explain how an IEEE 802.3 LAN operates. The operations discussed below do not describe any particular existing LAN. It is presented merely for the purpose of describing and defining additional procedures and terms specified in the 803.2 protocols. Subsequent chapters discuss the operation of some particular 802.3 LANs.

As shown in Fig. 1.13, each node is connected to the network cable via an *interface board.* The number of wires that connect the node to its interface board depends on the particular LAN. Also, the number of wires in the network cable depends on the particular LAN. Since the description below does not pertain to any particular LAN, Fig. 1.13 does not specify the number of wires in the network cable.

Example 1.17 As we shall see, the interface board is called a *hub* board in the 802.3 Starlan LAN, a *transceiver* board in the 802.3 Ethernet LAN, a *multiport repeater* in the 10BASE-T LAN, and in the 802.3 Cheapernet LAN, the interface board is an integral part of the node board. The network procedures performed by the interface board are also slightly different from one LAN to another.

The model describing the tasks performed by the interface board is shown in Fig. 1.14. In the model shown, the interface board is capable of transmitting signals into the network cable by using its transmitter, as well as receiving signals from the network cable via its receiver.

1.16 Reception of Frames by the Node

The incoming signals from the network cable are fed to the logic circuit of the interface board (Fig. 1.14). This logic circuit analyzes the incoming signals, and based on its analysis positions an electronic switch.

If the logic circuit decides that the incoming signal from the network cable is a valid Manchester signal, it positions the electronic switch to its upper position, enabling the incoming signal to reach the node. The node examines the incoming frame from the interface board and extracts the destination address field from the frame. If the node identifies its own address, it accepts the rest of the frame. While receiving valid signals, the node is not permitted to transmit data to the interface board. If the node has something to transmit, it defers the transmission for a later time (the deferring procedure).

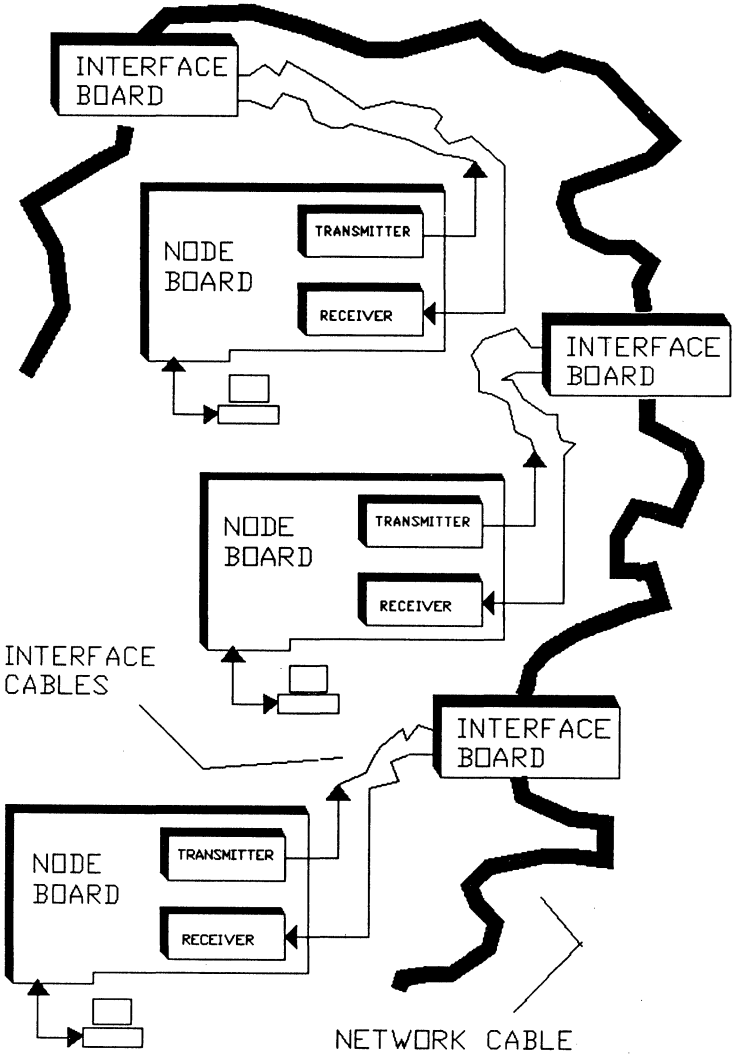


Figure 1.13 Connecting node boards to the network cable.

If the logic circuit of the interface board detects a Manchester code violation in the incoming signal from the network cable, it concludes that there are two or more nodes transmitting simultaneously and positions the electronic switch to its lower position, connecting the node to a special signal generator. This generates a special signal called the *collision presence signal.* The collision presence signal is then transferred to the node. Upon receiving this signal, the node concludes that there is a collision going on the network cable. If the node

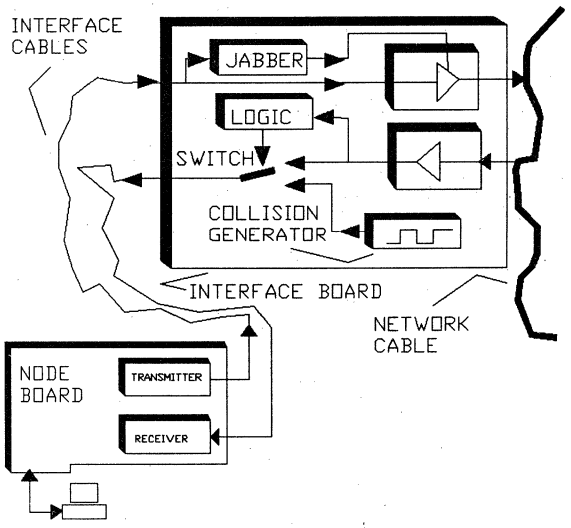


Figure 1.14 The interface board.

is busy receiving a frame and all of a sudden it receives the collision presence signal, it must abort the reception of the frame.

1.17 Transmission of Frames by the Node

If no signal arrives at the logic circuit of Fig. 1.14, the logic circuit positions the electronic switch at the upper position, transmitting no signal (silence) to the node. Upon receiving no signal from the interface board, the node concludes that the network is free.

When a node concludes that the network is free (and if it has a frame to transmit), it starts transmitting the frame to the interface board. The frame is transmitted in Manchester format starting with the preamble bits. Upon receiving the frame, the interface board transfers the frame into the network cable. The frame is then transmitted over the network cable to all the interface boards in the network. Each of the interface boards then transfers the incoming signals to their nodes.

In the LAN of Fig. 1.14, while transmitting bits into the network cable, the interface board also receives the bits just transmitted and transfers them back to its node. Thus, the transmitting node is transmitting bits and almost immediately receives back the bits it just transmitted.

If it happens that another interface board transmits to the network cable at the same time, the network cable contains a "mixture" of two signals. This mixture is analyzed by the logic circuit of the interface board, which is supposed to position the electronic switch at the lower

position, causing the node to receive the collision presence signal. Upon receiving the collision presence signal, the node realizes that it is transmitting into a collide network and it aborts the transmission.

If there is no collision, the node receives back the same exact bits that it just transmitted. The transmitting node compares the bits that it is transmitting to the incoming bits, and if it discovers that they are not identical, it concludes that some other node transmits at the same time, and it aborts the transmission. This mechanism of discovering a collision by comparing the transmitted bits to the received bits is called *collision detection* by bit comparison.

Example 1.18 Consider the case where a node is transmitting, and its interface board discovers a collision. The interface board then transfers the collision presence signal to its node. It therefore looks as if the collision detection by bit comparison is not necessary. Explain.

Solution The interface board decides whether or not there is a collision going on, merely on the basis of Manchester code violation. While it is not likely that the mixture that was generated due to a collision is a valid Manchester signal, it is still possible that the mixture is a valid Manchester signal (although not with great probability). If the collision was not detected by the interface board, it would be detected by the collision detection by bit comparison mechanism implemented by the node.

If while transmitting, the transmitting node discovers that it is transmitting together with another node (i.e., a collision), it starts executing the *jamming procedures.* The jamming procedure is a procedure whereby once the transmitting node discovers that it is transmitting together with another node, it keeps transmitting a special sequence of bits called the *jam pattern.*

A node that receives the jam pattern realizes that the network is in a collision. The reason for implementing the jamming procedure is to ensure that all other nodes in the network detect the fact that there is a collision on the network cable.

During a collision, there is more than one node transmitting at the same time (which is of course the definition of a collision). Upon detecting the collision, these nodes complete transmitting the jam pattern, wait for a random period of time, and then sense to see if they can retransmit the frame.

Example 1.19 Explain the reason for waiting a random period of time after transmitting the jam pattern.

Solution Since this waiting time is a random period of time, one of the nodes waits less time than the other nodes and hence finds a free network. The other nodes that waited for a longer period of time are unlucky, since when they complete their waiting period, they sense the network and find that the lucky node is already transmitting. This waiting time must be random, since if it would not be a random time, all the transmitting nodes would wait for the same

amount of time after the jamming, all would sense the network at the same time, and all would find a free network. All these nodes would start transmitting, a collision would occur again, and the LAN would never get out of the collision state.

The node should be designed with enough intelligence to generate the random time based on past-history experience. If the node realizes that it is already consecutively trying several times to retransmit the same frame unsuccessfully, it should increase the range of the random waiting period. The algorithm for calculating the proper range of the waiting period is called the *back-off algorithm.*

If the node attempts to transmit a frame and it encounters a collision during the transmission, the node should attempt to retransmit the same frame again without consulting with its PC. If after retrying 16 consecutive times, the frame is still not transmitted successfully, it is the responsibility of the node to inform the higher level software layers (the PC) about the situation.

Another procedure described in the 802.3 protocol is the *jabbering procedure.* The jabbering procedure is included to ensure that no node is able to monopolize the network cable. In the model of Fig. 1.14 we assigned the job of executing the jabbering procedure to the interface board. The jabber circuit on the interface board keeps track of how long the node is transmitting continuously. If the node is transmitting for longer than a fixed period of time, the jabber circuit disables further transmissions from the node.

Example 1.20 Is there a redundancy in implementing the jabbering procedure?

Solution There is a redundancy since the node is limited to the maximum period of time that it may transmit by virtue of the fact that the maximum length of the frame is limited.

Nevertheless, if implemented, the jabbering procedure assures that if something goes wrong with the node, the node will be disabled.

1.18 How Could a Collision Occur?

The carrier sense procedure (finding if the network is free prior to transmitting) is also referred to as the *listen-before-talk* procedure. Even though the listen-before-talk procedure is executed prior to each transmission, collisions may occur. One obvious cause of a collision is due to the fact that two nodes sense the network at the same time, both find a free network, and both start transmitting. While it is unlikely that two nodes start to transmit at the very same instant of time, cables in LAN could be quite long. Thus, it is possible that a node does not hear somebody else transmitting merely because the signal is still propagating on the cables toward the node.

1.19 The Different Sections of the 802.3 Node Board

Based on the discussion of the CSMA/CD access method protocol, it is evident that the node board has a substantial number of tasks to perform. Since there is no central controller in 802.3 LANs, each node is on its own and has the responsibilities of determining when it can transmit. The node must be prepared to accept data from the network (unsolicited frames) at any time. The node is also responsible for executing all the other procedures dictated by the protocol (e.g., Manchester encoding-decoding, constructing the frame, calculating the CRC, etc.).

Based on the various tasks that the node has to perform, the node board is divided into several sections as shown in Fig. 1.15. Each of the sections is responsible for implementing a group of different tasks specified in the network protocol. By distributing the various tasks among several circuits, the design of a node board becomes manageable.

1.19.1 The LAN manager section of the node

The LAN manager section of the node (Fig. 1.15) is the brain of the node. It is responsible for executing and managing all the procedures and data processing outlined in the LAN protocol.

1.19.2 The microprocessor section

The microprocessor section of the node (Fig. 1.15) contains a microprocessor chip, RAM chips, and ROM chips. This section provides the link between the PC and the LAN manager section.

When the PC has data to transmit, it interrupts the microprocessor section, stores the data in the RAM chips of the microprocessor section, and instructs it to transmit the data. The microprocessor rearranges the data into a format acceptable by the LAN manager and then instructs the LAN manager to transmit the data to the network. The microprocessor monitors the transmission process and interrogates the LAN manager to find whether the transmission was successfully completed.

Once the PC is ready to accept frames from the network, it interrupts the microprocessor and instructs it to enable frame reception. The microprocessor responds by instructing the LAN manager to start receiving frames. The microprocessor monitors the frame reception process, and once the received frames are processed by the LAN manager, the microprocessor interrupts the PC and transfers to it the received data.

Some node boards do not contain a microprocessor section. In these cases, the PC directly controls and monitors the operation of the LAN manager.

1.19.3 The Manchester decoder/encoder section

The NRZ-to-Manchester section (Fig. 1.15) receives an NRZ signal from the LAN manager section and converts it to Manchester signal.

The Manchester-to-NRZ section (Fig. 1.15) receives a Manchester signal from the network and converts it to an NRZ signal. As previously discussed, converting from a Manchester signal back to an NRZ signal is a process also called *clock recovery,* and requires high degree of accuracy. The quality of the LAN depends on the degree of accuracy accomplished by the clock recovery process.

1.19.4 The transmit and transmit control sections of the node

The transmit and transmit control sections (Fig. 1.15) are responsible for the actual transmission of frames. As shown, the transmit section accepts the data in Manchester format from the NRZ-to-Manchester section and transmits the data provided that it is enabled by the transmit control section. The signal to be transmitted is denoted TxD in Fig. 1.15.

The transmit control section makes the decision whether or not to enable the transmit section based on the control signals supplied to it by the LAN manager and based on the TxD signal.

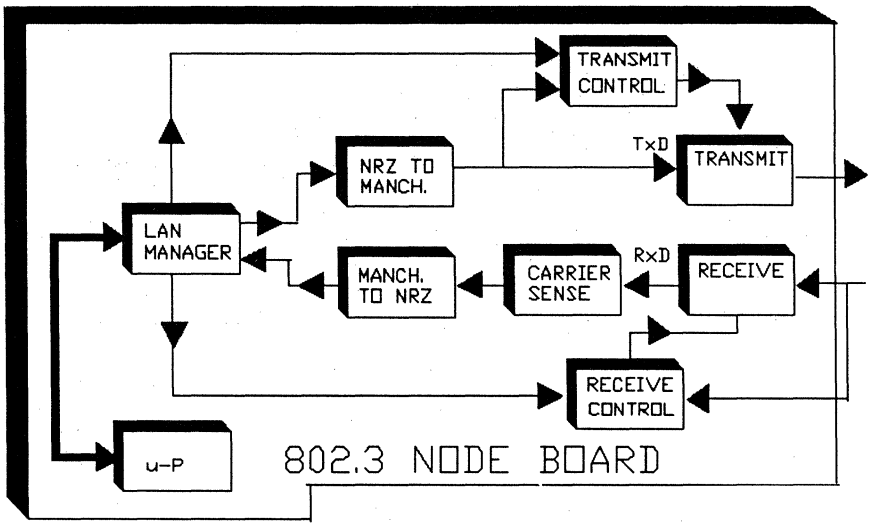


Figure 1.15 The 802.3 node board.

1.19.5 The receive and receive control sections of the node

The receive and receive control sections (Fig. 1.15) are responsible for the actual reception of frames. The receive section accepts the data in Manchester format from the network and transfers the data (denoted RxD) to the next section provided that it is enabled by the receive control section. As shown, the incoming signal from the network is fed to the receive control section as well.

The receive control section makes the decision whether or not to enable the receive section based on the control signals supplied to it by the LAN manager, as well as the incoming signal from the network. The receive and receive control sections must incorporate the appropriate circuitries to filter out incoming noise signals.

1.19.6 The carrier sense section of the node

The carrier section (Fig. 1.15) is responsible for determining the status of the network. This section generates control signals which provide information about the presence or absence of a Carrier on the network cable. This section makes such decisions based on the RxD signal.

1.19.7 Note about the node implementation

The implementation of the various sections of the node is accomplished in hardware and software. When designing a node, care must be exercised not to "reinvent the wheel." That is, fortunately, many of the components that are needed for the implementation were already designed by reputable integrated circuits manufacturers and are available as inexpensive, off-the-shelf chips. The designer can therefore concentrate the design efforts on performance improvement, reliability, and firmware development.

While it is not our intention to present the subject of node design as too trivial, as we shall soon see, designing a node board consists of finding the proper off-the-shelf chips that are capable of implementing the various sections in Fig. 1.15.

Example 1.21 The 802.3 node board model of Fig. 1.15 is applicable for the implementation of all the baseband LANs in the 802.3 group. That is, this model is utilized for the implementation of the 1BASE5 Starlan node board, the 10BASE5 Ethernet node board, the 10BASE2 Cheapernet node board, and the 10BASE-T Twisted Pair Ethernet node board, since all these LANs conform to the CSMA/CD access method (with baseband transmission). However, since each of these LANs drives different type of cables, the implementation of the transmit and receive sections in Fig. 1.15 are different from LAN to LAN.

1.20 Summary

This chapter discusses the principle of operation of LANs. Figures 1.9 and 1.10 represent the most popular standard LANs that exist today: the CSMA/CD, the token bus, and the token ring. The token bus and token ring are deterministic types. That is, one can determine in advance when and for how long each node would be able to transmit. On the other hand, the CSMA/CD LAN is probabilistic (i.e., there is no guarantee of when a node would be able to access the network cable).

There are situations that demand the utilization of the deterministic type LANs. For example, such situations exist in automatic manufacturing environments where the machines (robotic stations) report status information to a central computer via the LAN. The central computer analyzes the status reports and issues updated work orders to the work stations based on these reports.

However, in office applications the trend is to utilize the CSMA/CD LANs. Each station is independent from the other stations in the network and thus allows the removal or addition of a station quickly and easily.

1.20.1 What's next?

Chapters 2 through 7 deal with implementing (hardware and software) the equipment of existing CSMA/CD LANs. Chapter 2 discusses the principle of operation of the 1BASE5 (Starlan), and Chap. 3 deals with the hardware implementation of the transmitter and receiver of the 1BASE5 node board. Chapter 4 discusses the LAN manager section of the 1BASE5 node board utilizing the 82588 LAN controller, and Chap. 5 discusses the software implementation for the hardware introduced in Chap. 4. Although Chaps. 2 through 4 are oriented toward the 1BASE5 implementation, these topics are important not only for the 1BASE5 implementation but also for understanding and implementing any CSMA/CD LANs, and providing the know-how and design philosophy that are common among all existing (and future) CSMA/CD LANs.

Chapter 6 deals with the 10BASE5 (Ethernet) and 10BASE2 (Cheapernet) LANs. These CSMA/CD LANs are shown to be implemented with the 82586 LAN controller chip.

Chapter 7 deals with the latest CSMA/CD LAN, the 10BASE-T (the Twisted Pair Ethernet LAN). This chapter discusses the implementation of 10BASE-T node boards and 10BASE-T multiport repeaters.

Chapter

2

The Principle of 1BASE5 (Starlan)

2.1 Introduction to Starlan

This chapter discusses the principle of operation of a local area network (LAN) called Starlan. Subsequent chapters show how to design and implement this LAN in hardware and software. The Starlan introduced is designed to comply with the standard LAN protocol IEEE 802.3 1BASE5 Starlan. One of the most popular applications of Starlan is to establish communication between several PCs.

To build a Starlan network, two types of boards are needed: node boards and interface boards called hub boards.

2.1.1 The node board

Figure 2.1 shows a Starlan node board. The node accepts data from the PC, processes the data, and transmits the processed data to the network via its transmitter. The node receives data from the network via its receiver, processes the data, and sends the processed data to its PC. As shown in the diagram, the Starlan node uses two twisted pairs of wires; one pair is used to transmit data, and the other is used to receive data. The node board is shown plugged into one of the input-input (I/O) slots of the PC mother board.

Example 2.1 The Starlan node board of Fig. 2.1 is shown as plugged into one of the I/O slots of the PC mother board. Does this represent one of the requirements specified in the IEEE 802.3 1BASE5 Starlan LAN protocol?

Solution As discussed in Chap. 1, the DTE (which is the PC in our case) is not part of the protocol. The Starlan node board could be a stand-alone card and it can be connected to any type of DTE.

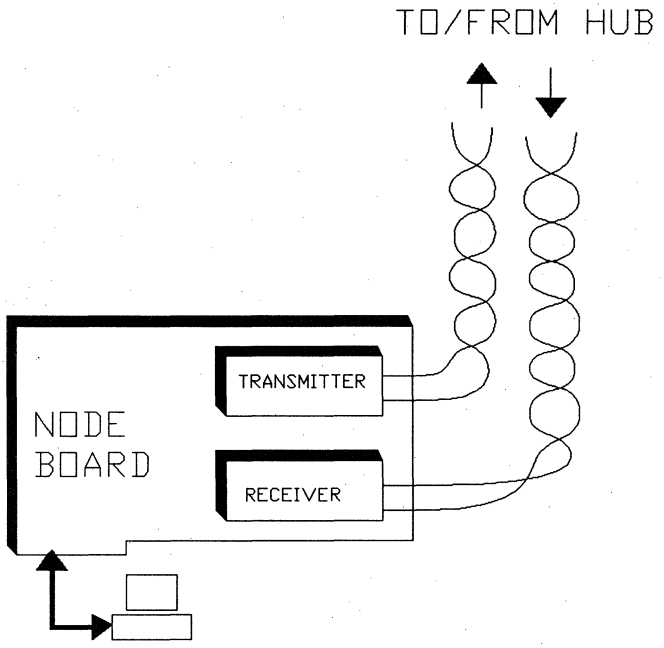


Figure 2.1 Starlan node board.

2.1.2 The hub board

Figure 2.2 is a diagram of a *hub board.* A hub board has several ports called *node ports.* The hub board shown in Fig. 2.2 happens to have four node ports. Each of the node ports consists of a transmitter and a receiver. As shown in the diagram, each transmitter and each receiver is connected to a twisted pair of wires.

The hub board also has one port called a hub port. The hub port also has a transmitter and a receiver, each connected to a twisted pair of wires.

2.2 Connecting Node Boards to the Hub Board

To build a Starlan network, stations (PCs together with their nodes) have to be connected to the hub as shown in Fig. 2.3. The hub of Fig. 2.3 has four node ports and therefore can support up to four stations. We chose to connect only two nodes to the hub of Fig. 2.3. Typically, a hub board has 4 to 12 node ports; however, the Starlan protocol puts no limitation on the number of node ports that a hub can have.

As indicated in Fig. 2.3, the receivers of the node boards are connected to the transmitters of the hub, and the transmitters of the node boards are connected to the receivers of the hub.

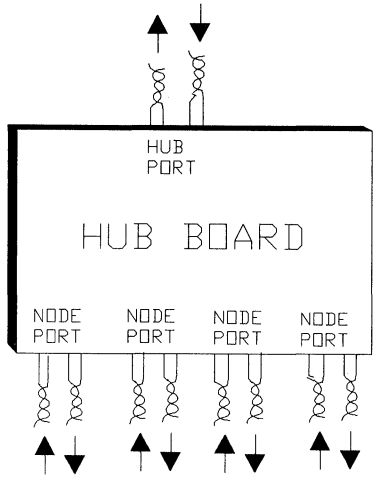


Figure 2.2 Starlan hub board.

The topology shown in Fig. 2.3 is called a *star topology*. (The hub board is the center of the star, and the cables connecting the nodes to the hub are the rays of the star.)

Note that the hub port of the hub board is connected to itself; that is, its transmitter is connected to its receiver.

2.2.1 Generating additional levels

Figure 2.4 shows how an additional hub level is added to the star. By convention, the hub levels are numbered sequentially from bottom to top.

The network shown in Fig. 2.4 can be extended to the one shown in Fig. 2.5. Now there are two hub boards in hub level 1. Starlan protocol puts no limitation on the maximum number of hub boards at any hub level, except for the top level, which must always contain a single hub.

To further extend the network, an additional hub level can be added as shown in Fig. 2.6. In Starlan, a maximum of five levels is permitted.

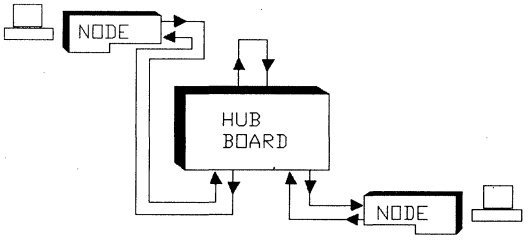


Figure 2.3 Single-level Starlan (minimum configuration).

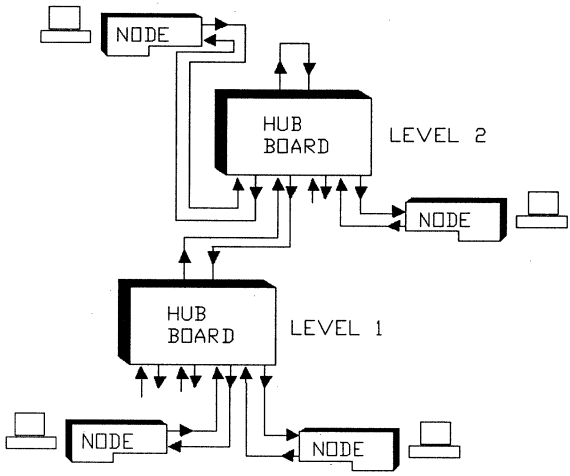


Figure 2.4 Two-level Starlan.

Note that the hub port of the hub located at the top level in Figs. 2.3, 2.4, 2.5, and 2.6 has its hub port connected to itself, while a hub located in a lower level has its hub port connected to a node port of a higher level hub.

The 1BASE5 Starlan protocol specifies the use of unshielded 24-gauge twisted pairs of wires for connecting node to hub and hub to hub.

Example 2.2 Assuming that each hub board has four node ports, what is the maximum number of stations that can be connected to the network if

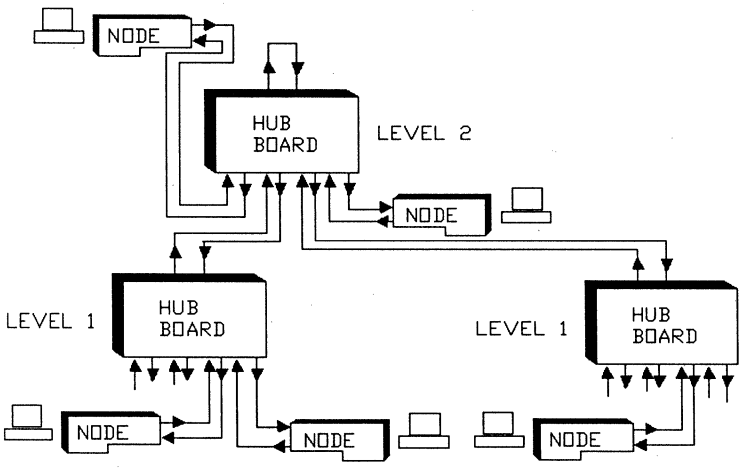


Figure 2.5 Two-level Starlan (with level 1 having two hubs).

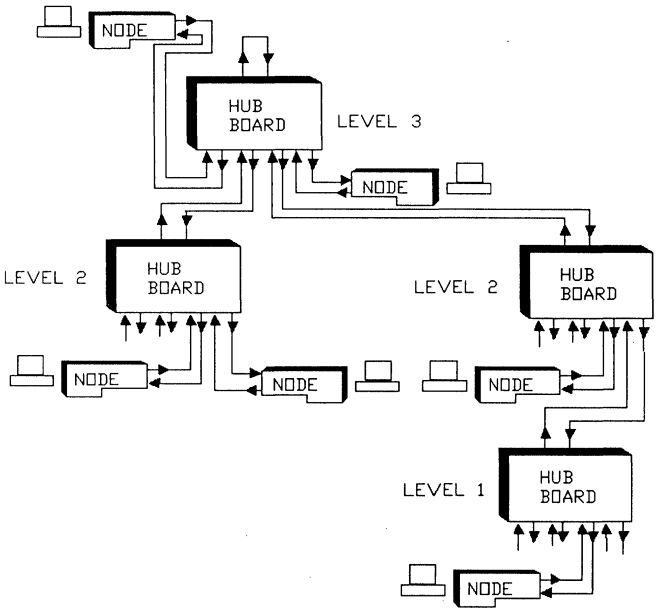


Figure 2.6 Three-level Starlan.

1. The Starlan has one hub level?
2. The Starlan has two hub levels?
3. The Starlan has three hub levels?
4. The Starlan has four hub levels?
5. The Starlan has five hub levels?
6. The Starlan has six hub levels?

Solution

1. One hub level: As shown in Fig. 2.7, the maximum number of stations for this case is 4.
2. Two hub levels: As shown in Fig. 2.8, the maximum number of stations for this case is 16.
3. Three hub levels: The maximum number of stations for this case is 64.
4. Four hub levels: The maximum number of stations for this case is 256.
5. Five hub levels: The maximum number of stations for this case is 1024.
6. Six hub levels: The maximum number of hub levels permitted in 1BASE5 Starlan is 5. Six hub levels are not permitted in 1BASE5 Starlan.

2.2.2 Length of wires in 1BASE5 Starlan

When using 24-gauge wires of the type specified in the network protocol, the maximum length of wires connecting two hub boards is 250 m. The maximum length of the wires connecting a node to its hub is also 250 m.

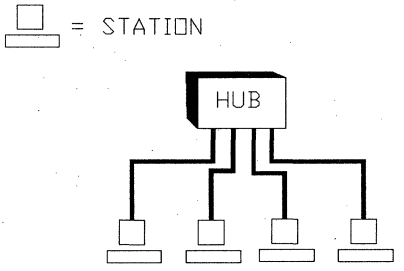


Figure 2.7 Single-level Starlan with four nodes.

Example 2.3 What is the maximum distance between any two stations when the Starlan has a single-level hub configuration? (Assume that 24-gauge wires are used.)

Solution In a single-level hub configuration (like the one shown in Fig. 2.3), the left node could be 250 m away from the hub, and the right node could also be 250 m away from the hub. The maximum distances between the two nodes is therefore 500 m.

This is the reason for the "5" in the protocol name "1BASE5". That is, the 5 indicates the maximum distance in hundreds of meters between two nodes in the minimum configuration (a single-level hub is the minimum configuration).

Example 2.4 Is the configuration shown in Fig. 2.9 permitted in 1BASE5 Starlan? Assume that 24-gauge wires are used.

Solution The configuration shown is permitted since

1. The configuration shown has five hub levels, which is permitted in Starlan (a maximum of five Hub levels is permitted).
2. The configuration shown has 250-m node-to-hub and hub-to-hub cables, which is permitted in Starlan (250 m is the maximum length allowed).

Example 2.5 What is the maximum allowed distance between any two stations in Starlan?

Solution Figure 2.9 represents the maximum configuration of Starlan. The distance between the two stations is 2500 m.

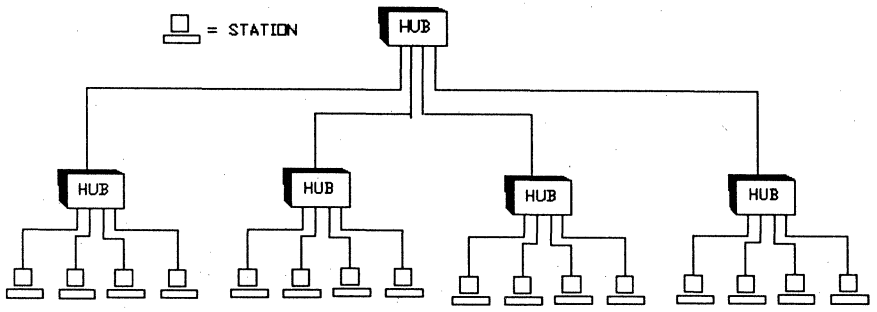


Figure 2.8 Two-level Starlan with 16 nodes.

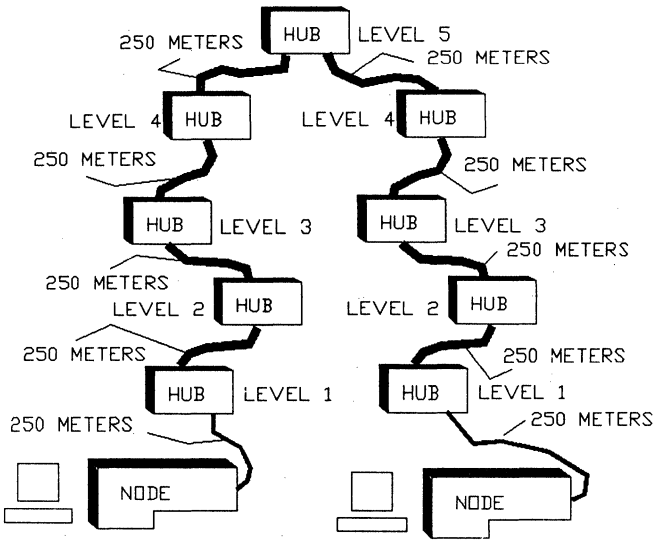


Figure 2.9 Five-level Starlan (maximum configuration).

2.3 Upstream and Downstream Sections of the Hub

The logic block diagram of a hub is shown in Fig. 2.10. As shown, the hub has two sections: the *upstream* section and the *downstream* section.

2.3.1 The downstream section of the hub

The downstream section has one input at its top and several outputs at its bottom. The downstream section retransmits the signal supplied to its input (by a higher level hub) to all the devices connected to its outputs. The devices connected to the outputs of the downstream section

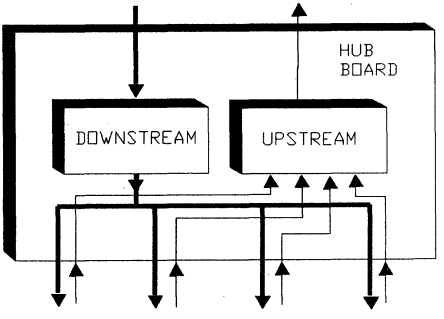


Figure 2.10 Hub board logic diagram.

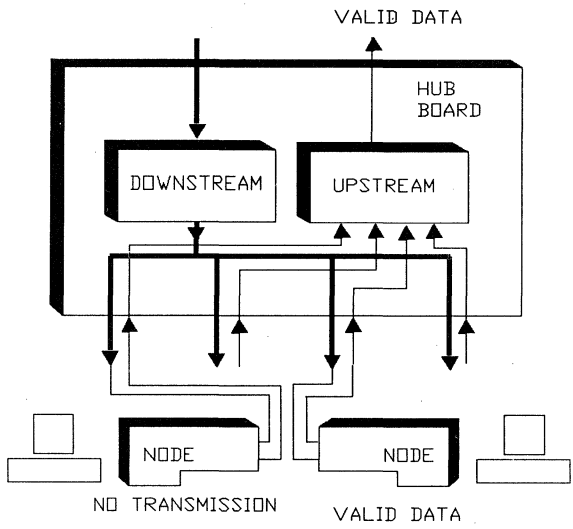


Figure 2.11 The hub board as a repeater.

are nodes and/or lower level hub boards. The hub of Fig. 2.11 is shown to be connected to two nodes. As indicated in Fig. 2.11, all the devices connected to the outputs of the downstream section receive the same identical signal.

Since the nodes or the hubs which are connected to the outputs of the downstream section could be as far as 250 m away from it, and the higher level hub that supplies the signal to the input of the downstream section could also be as far as 250 m away from the hub, a repeater circuit is required. The downstream section of the hub serves the function of a repeater which supplies a fresh clean signal.

2.3.2 The upstream section of the hub

As shown in Fig. 2.10, the upstream section has several inputs at its bottom and a single output at its top. The upstream section of the hub outputs a signal based on the signals supplied to its inputs. The logic functions performed by the upstream section are

1. If the upstream section senses that it receives valid data at only one of its inputs and senses no data from the rest of its inputs, the upstream section serves as a repeater, retransmitting the valid data signal upward. This case is illustrated in Fig. 2.11.
2. The upstream section of the hub generates a special signal called the collision presence signal at its output whenever it senses a collision at its inputs. This case is illustrated in Fig. 2.12.

In Fig. 2.12, two of the devices connected to the inputs of the upstream section are transmitting simultaneously, which is of course a collision. The upstream section senses this collision and generates the collision presence signal. This signal is deliberately generated by the upstream section with a Manchester code violation.

3. The upstream section of the hub generates the collision signal presence if it senses the collision signal presence at one or more of its inputs. This case is illustrated in Fig. 2.13.

   Figure 2.13 shows the case where a hub in a lower level senses a collision and therefore generates the collision signal. This collision signal is supplied to the higher level hub. The higher level hub senses the collision signal at one of its inputs, and therefore it too generates the collision presence signal.

4. If the upstream section does not receive any signal at its inputs, it does not output any signal.
5. The upstream section keeps track of how long each of its inputs is transmitting continuously. If one of its inputs transmits continuously longer than a fixed amount of time, the upstream ignores any further inputs from that input. As discussed in an earlier chapter, this function is known as executing the jabbering procedure.

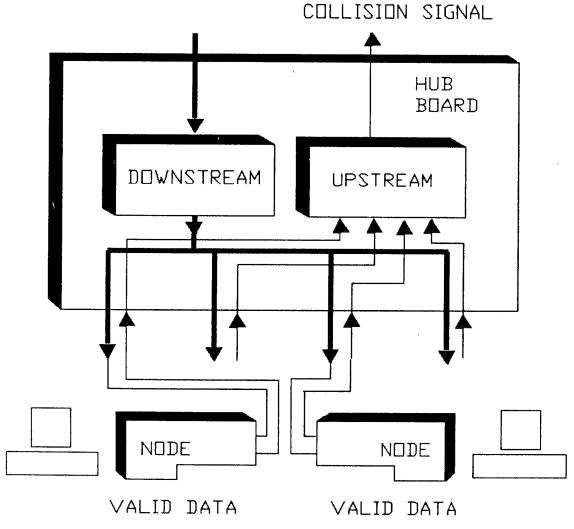


Figure 2.12 Generation of the collision presence signal.

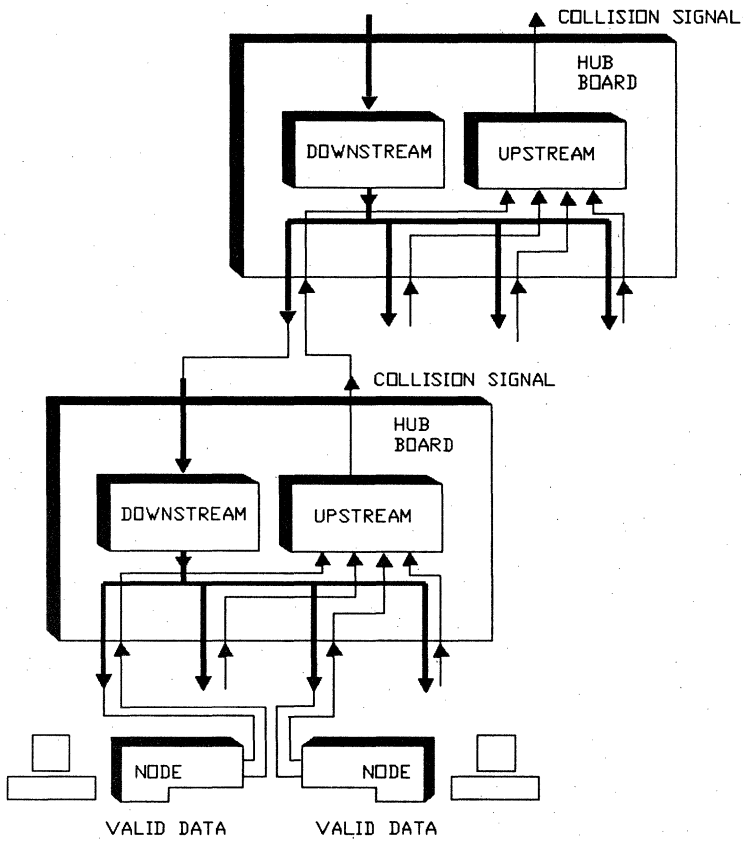


Figure 2.13 Regenerating the collision presence signal.

2.4 The Header Hub

The hub board located at the top level of a Starlan network is called a *header hub,* or HHUB for short. As shown in Fig. 2.14, the output of the upstream section is directly connected to the input of the downstream section.

Any HUB board that is not located at the top level is called an *intermediate hub,* or IHUB for short. The IHUB and HHUB boards are identical and could be interchanged within the Starlan. The only thing that makes a hub a HHUB or an IHUB is the way we connect the top of the hub (the hub port).

2.5 Principle of Operation of Starlan

We can now turn back to Fig. 2.6 and explain how the Starlan works.

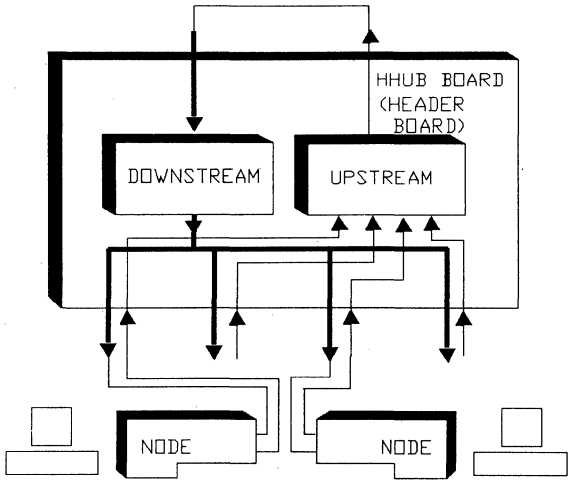


Figure 2.14 The header hub (HHUB).

2.5.1 Transmitting and receiving a frame

When a PC wishes to transmit data to another PC in the Starlan network, it sends the destination address and the information data to the node. The node then executes the "listen-before-talk" procedure. That is, it examines the signals applied to its receiver, and if it finds a free network, it constructs a frame and sends the frame in a Manchester format to its hub. The data are then traveling to the hub connected to this node.

The hub examines the signals supplied to its inputs, and outputs a signal to the higher level hub in accordance with the logic function shown in Figs. 2.11, 2.12, and 2.13. The signal (whatever was generated) is then traveling upward to the next higher level hub, which performs the same logic function.

Eventually, the very top level hub (the HHUB) receives the signal, and it too generates the corresponding output signal. The top level hub, however, does not have a hub at a higher level, so it turns around the signal, supplying it to the input of its downstream section, as shown in Fig. 2.14.

The signal that was turned around by the HHUB is then distributed to all the IHUBs and nodes connected to it. The next lower level IHUB receives the signal, and it too distributes the signals to all devices connected to it. Eventually, all nodes in the network receive the signal. This signal can be either the original signal transmitted or the collision presence signal.

Each node examines the incoming signal, and if the signal is valid (i.e., not the collision presence signal), it analyzes the destination

address field and makes a decision whether the frame is intended for it. A node which decides that the frame is intended for itself receives the rest of the frame.

On the other hand, if the nodes receive the collision presence signal, they know that there is a collision going on and disregard the incoming data. If a collision occurs during the middle of the reception of a frame, the receiving node aborts the reception of the rest of the frame.

The transmitting node is also receiving the data that it is currently transmitting. It compares the data that it transmits with the data that it receives, and if it finds that they are not identical, it concludes that there is something wrong going on in the network (either a collision, open wire, etc.). As discussed in an earlier chapter, this is known as collision detection by bit comparison.

If the transmitting node finds that it is receiving the collision signal, it starts executing the jamming procedure. As discussed in an earlier chapter, the jamming procedure consists of aborting the transmission and transmitting the jam pattern.

In addition to the procedure described above, all the other procedures described in the 802.3 protocol, which were described in an earlier chapter, are executed as well (e.g., the back-off procedure, reporting back to the PC about occurrence of 16 consecutive collisions, etc.).

2.5.2 The propagation delays

The maximum allowed propagation delay in 1BASE5 Starlan is specified as 4 μs.

Example 2.6 The 4-μs maximum propagation delay specified in IEEE 802.3 1BASE5 means that a signal generated by any node in the network should reach all the nodes in the network in no more than 4 μs. As previously discussed, the maximum distance between any two nodes in 1BASE5 Starlan is 2500 m. When designing the Starlan equipment, the node board and the hub board should be designed to process the signals so that the processing time plus the time it takes a signal to travel a distance of 2500 m, would not exceed 4 μs.

2.6 The Telephone Wires and Starlan

2.6.1 The telephone wiring

One of the main reasons for the attractiveness of the Starlan network is its ability to utilize existing telephone wires. Figure 2.15 illustrates how the telephone company installs a telephone service to its customer.

A cable is brought from the telephone pole to the building and is terminated at a wiring closet called a HHUB wiring closet. That cable is

typically composed of 25 pairs of twisted, unshielded 24-gauge wires, the type specified in 1BASE5 Starlan.

The wires are then distributed from the HHUB wiring closet to wiring closets in each room. These are called IHUB wiring closets. Each cable in Fig. 2.15 represents a cable composed of two twisted pairs. The cables are then distributed from the IHUB wiring closets to jacks in each of the rooms. Although a regular telephone requires only one pair of wires, each jack is supplied with two pairs. The telephone instrument utilizes only two out of the four wires supplied to it by the jack.

The heavy lines shown in Fig. 2.15 represent extra connections that the telephone company made when the service was first installed at the building. The telephone company installed these extra connections so that it could supply additional telephone lines to the building upon customer request. The telephone company would be able to accommodate the customer immediately, since all the wiring is already installed. That is, connecting the new jacks to one of the extra connections is all that is required to install a new telephone line.

Although Fig. 2.15 shows that there are only two extra connections, typically, the telephone company installs many more extra connections.

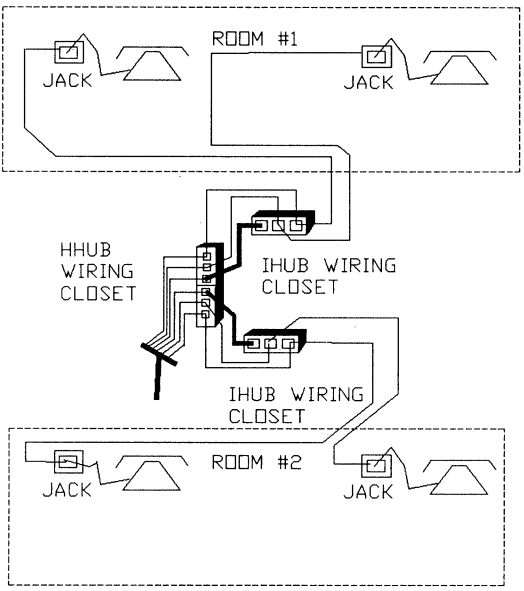


Figure 2.15 Typical telephone wiring.

2.7 Using Existing Telephone Wiring for Starlan

Figure 2.16 illustrates how the extra connections of Fig. 2.15 are used to install a Starlan network. That is, a HHUB board is connected to the HHUB wiring closet, and IHUBs are connected to the IHUB wiring closets. In Fig. 2.17 we removed the telephone lines to clarify that the resultant network has a legal Starlan topology. Recall that each heavy line in Figs. 2.15, 2.16, and 2.17 represents two pairs of twisted pairs of wires, which is exactly the number of twisted pairs that are required to connect a node to its HUB and a HUB to a higher level HUB.

Example 2.7 Does the resultant topology of Fig. 2.17 comply with 1BASE5 Starlan as far as distances are concerned?

Solution Probably yes, since the telephone company rarely stretches wires from the HHUB wiring closet to the IHUB wiring closets for distances that exceed 250 m. Of course it is not a coincidence that the telephone wirings are complying with the protocol requirements. The people who drafted the 1BASE5 Starlan (the IEEE 802.3 committee) designed the protocol so that it can utilize existing extra telephone wires.

Example 2.8 When the transmit circuits of the node boards and the hub boards are designed in strict compliance with the Starlan protocol, there are no cross-talks and no interferences between the LAN and the live telephone wires.

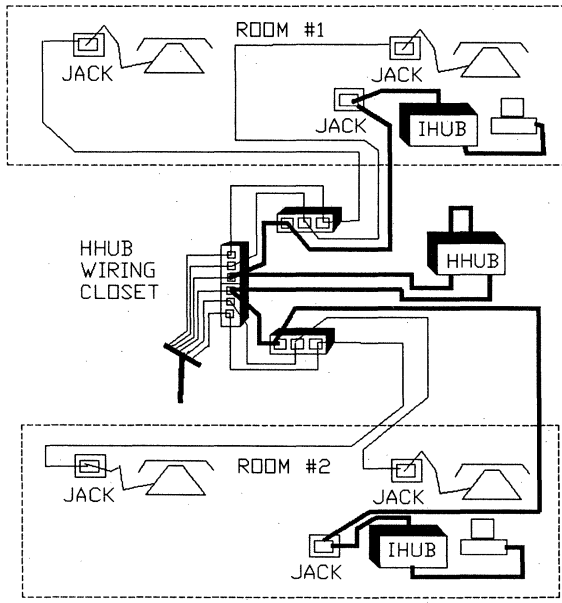


Figure 2.16 Utilizing the unused telephone wiring for Starlan.

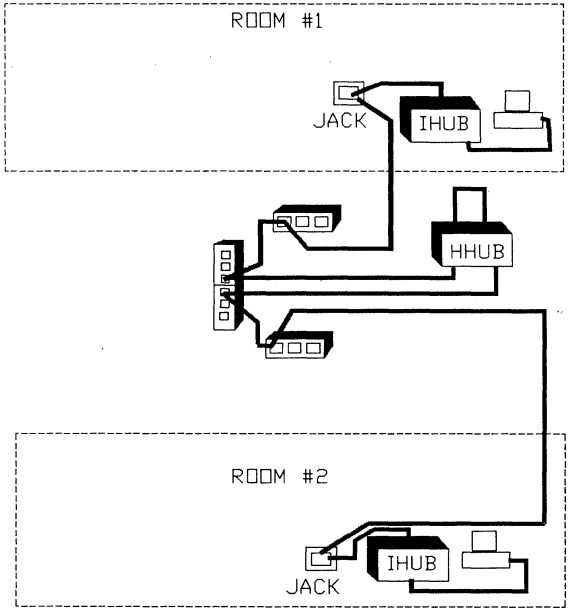


Figure 2.17 The resulting star topology.

2.8 Summary

This chapter discusses the principle of operation of the Starlan. The components needed for the implementation of this LAN are hubs, node boards, and DTEs. The main attractiveness of this LAN is the ability to use inexpensive twisted pairs of wires, the type of wires used by the telephone company. Another important advantage of the Starlan is the fact that the Starlan topology is the same as the topology used by the already installed telephone wiring. Thus, the end user may save installation costs by utilizing the already installed, unused telephone wires. A disadvantage of the 1BASE5 Starlan is the fact that its data rate is limited to only 1 Mbit/s.

Chapter

3

The Transmitter and Receiver of the 1BASE5 Node Board

3.1 Introduction

This chapter deals with the design of the transmit, transmit control, receive, and receive control sections of the node board (see Fig. 1.15). These sections are designed to comply with the IEEE 802.3 1BASE5 Starlan protocol. The chapter introduces some design options for the implementation as well as additional terms, procedures, and specifications dictated by the 802.3 protocol.

Although the material presented in this chapter is oriented toward the Starlan node, many of the topics are applicable for other 802.3 LANs as well.

3.1.1 The transmit section of the node

The signals outputted by the transmit circuit have to travel long distances that are too far for a regular single-ended transistor-trans logic (TTL) signal to travel without getting distorted on its way.

Several techniques for sending signals over long wires are available in communication. The 802.3 LANs use the technique of transferring the data as differential signals. Figure 3.1 shows a differential amplifier that accepts a single-ended signal at its input and converts it to a differential signal. Experience and theory show that a differential signal is capable of traveling without getting distorted for longer distances than a single-ended signal. To further protect the differential signal from getting distorted, the two wires that carry the differential signal are twisted together.

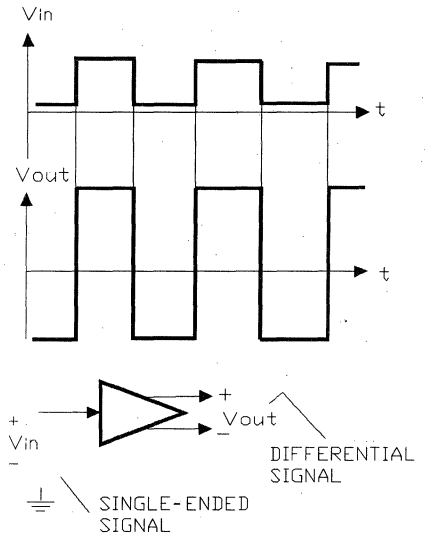


Figure 3.1 Single-ended to differential conversion.

Example 3.1 Explain the reason for choosing to transfer data as differential signals over long wires.

Solution While it is not within the scope of this book to provide the mathematical proof, it can be seen intuitively that if a spike of noise manages to inject itself into one of the two wires that carry the differential signal, the noise would probably be injected into the other wire too. Thus, the voltage on each of the wires increases (or decreases) by the same amount. However, the differential signal (the voltage difference between the two wires), remains the same.

The only problem with the output signal shown in Fig. 3.1 is that the signal has a fast rise time and fast fall time. A wire that carries a signal with fast transition radiates energy to its surroundings. This energy may severely distort the signals traveling on the nearby LAN wires as well as interfere with nearby electronic equipment. We therefore need to increase the rise and fall times of the signal.

Example 3.2 Design a circuit that converts a single-ended signal to a differential signal. The differential signal should not have fast rise and fall times.

Solution Figure 3.2 shows how the 26LS30 differential amplifier is utilized to slow down the rise and fall times. The 26LS30 is equipped with the proper pins to increase the rise and fall times of the differential output. By connecting the 5-pF capacitors as shown, the rise time and fall time increase to about 150 ns. The signal generated is therefore trapezoidal.

The 26LS30 is equipped with an enable input pin. If a 0 is applied to the enable pin, the 26LS30 performs the conversion. If a 1 is applied to the enable pin, the 26LS30 is disabled, causing the differential output voltage to be at 0 V.

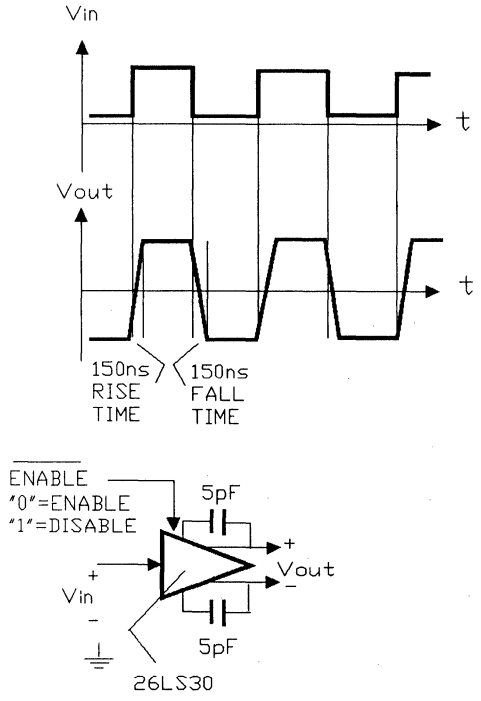


Figure 3.2 The 26LS30 as a single-ended to differential converter.

The 26LS30 is also called an RS-422 driver since it is used as a driver in RS-422 application. It is inexpensive, generic, and available as an off-the-shelf item.

The transmit and transmit control sections of Fig. 1.15 are shown in the greater detail in Fig. 3.3. As indicated, the output of the 26LS30 is fed to the primary of a pulse transformer. The secondary of the pulse transformer supplies the signal to a twisted pair of wires. The pulse transformer is used for the purpose of providing dc isolation between the node and the rest of the network. As we shall soon see, the receiver circuit also requires a pulse transformer of the same type. The pulse transformers of both the transmit and receive circuits are available as commercial items from a variety of vendors in a single package (dual transformers), which is a printed-circuit mounted package. These are generic, inexpensive, off-the-shelf items, designed especially for 802.3 LAN applications.

3.1.2 The TxD signal

As indicated in Figs. 1.15 and 3.3, the data to be transmitted are denoted as the TxD signal. The transmit circuit receives the TxD signal at its input from the previous sections of the node and converts it

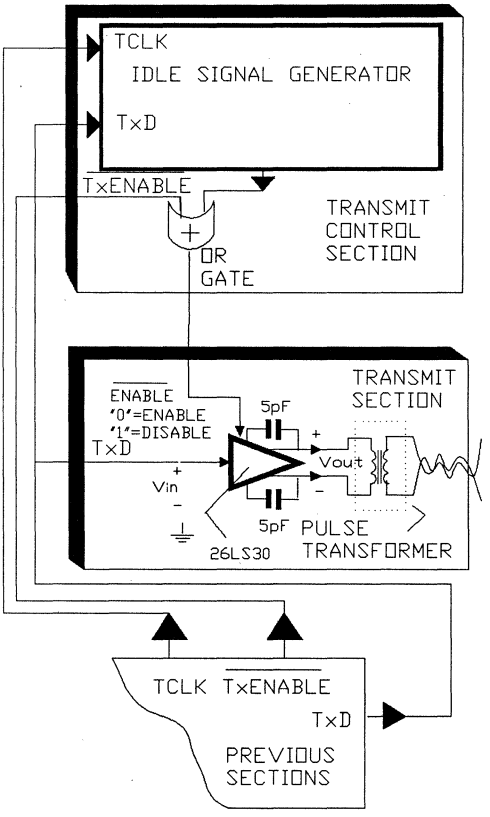


Figure 3.3 The transmit and transmit control circuits.

to a differential signal. Whenever the node has no data to transmit to the network, it outputs a continuous 1 to TxD.

Our objective is to disable the transmit circuit whenever there are no data to be transmitted (i.e., whenever the TxD is continuously at 1).

Example 3.3 What is the reason for disabling the 26LS30 when TxD is at continuous 1 (i.e., when there is no transmission)?

Solution We must disable the 26LS30, since its output is directly connected to the primary of the pulse transformer. If the 26LS30 is not disabled when the TxD is continuously 1, the differential output of the 26LS30 is also continuously high (a dc voltage). Obviously, we cannot apply a constant dc voltage to a transformer (i.e., applying a dc voltage to the transformer is the same as shorting out a power supply). The transmit control circuit of Fig. 3.3 must disable the 26LS30 whenever it detects that TxD is continuously high.

3.2 The Transmit Control Circuit

The transmit control circuit of Fig. 3.3 is composed of an OR gate and a circuit called the *idle signal generator*. The transmit control circuit

receives three signals from the previous sections: TxENABLE, TCLK, and TxD. Based on these signals, the transmit control circuit decides whether to enable or disable the 26LS30 of the transmit circuit.

Since the LAN network serves several users, it is quite possible that a user is applying power to its station (a PC together with the node) while the rest of the network is already in operation. If the 26LS30 is enabled at power-up, the signal present on the TxD at that time would pass to the network. This is of course a gross violation of the network protocol, since the node should sense the cable prior to transmitting (listen before talk).

Example 3.4 Explain the purpose of the OR gate in Fig. 3.3.

Solution The previous sections of the node are responsible for keeping TxENABLE at 1 upon power-up. When the TxENABLE supplied to the OR gate of Fig. 3.3 is 1, the transmit circuit is disabled, regardless of the value of the second input to the OR gate.

The previous sections keep TxENABLE at 1 until the LAN manager decides that it is time to enable the transmit circuit (i.e., after the node board was properly configured and initialized). When the TxENABLE is 0, the second input to the OR gate determines whether the transmit circuit is enabled or disabled.

The second input to the OR gate is based on the values of two signals, TCLK and TxD.

In the following paragraphs we are going to make use of the TCLK signal that happens to be available from the previous sections. The TCLK signal is called the transmit clock signal. When the previous section is transmitting data, it transmits the data to the TxD, bit after bit at a rate dictated by this clock as shown in Fig. 3.4. This clock is ticking even if there are no data to be transmitted (i.e., even if TxD is continuously at 1).

As discussed, the transmit control circuit has to disable the 26LS30 whenever the TxD is at continuous 1. The IEEE 802.3 protocol requires

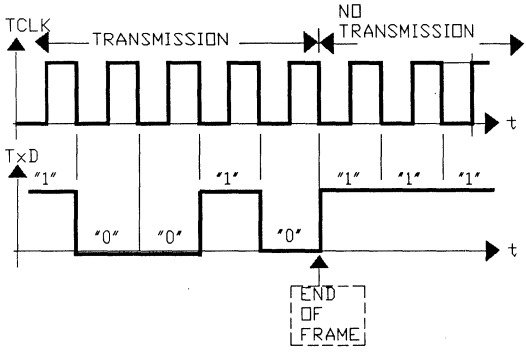


Figure 3.4 The TCLK clock of the node.

that the above disabling will not occur immediately at the end of the frame. Rather, the protocol requires that the transmitter output (in our case the 26LS30 output) stays high for a while, and then slowly reduces to zero. The reason for this requirement is that the last bit of the frame might be still in the process of being received by the hub board. If we immediately cut off the 26LS30, the last bit might be incorrectly interpreted by the hub board. Figure 3.5 illustrates this requirement.

The signal that is transmitted by the 26LS30 at the end of the frame is high for a while, reduces slowly to zero, and then stays at zero. It is called the *idle signal.* The idle signal generator of Fig. 3.3 is responsible for generating the idle signal.

Example 3.5 Construct the idle signal generator circuit by using a shift register.

Solution In the following discussion, we assume that the node board is already configured, and therefore the previous section supplies TxENABLE = "1".

One possible circuit is shown in Fig. 3.6. The shift register used in the 74LS164. The data to be shifted is a constant 1, connected to the input of the shift register via a pull-up resistor.

The TCLK signal from the previous section is connected to the clock pin of the shift register, and the TxD is connected to the clear pin of the shift register. The input to the OR gate is taken from the third cell of the shift register.

Whenever the TxD goes to 0, the shift register is cleared, and all the cells of the shift register are filled with 0s, thus supplying a 0 to the OR gate and enabling the 26LS30.

When TxD goes to 1, the shift register is enabled, and every time the shift register receives a clock from TCLK, the 1 at its input is pushed to the next cell. The third cell will be filled with 1 only if TxD is at 1 for three consecutive TCLK clocks.

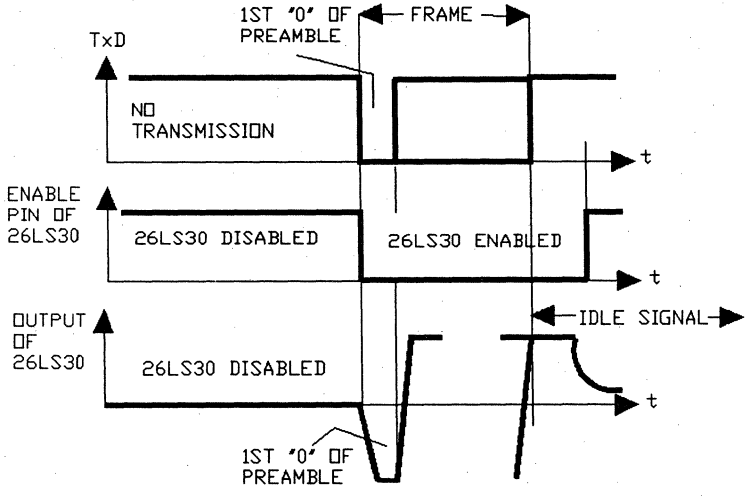


Figure 3.5 Enabling and disabling the transmitter.

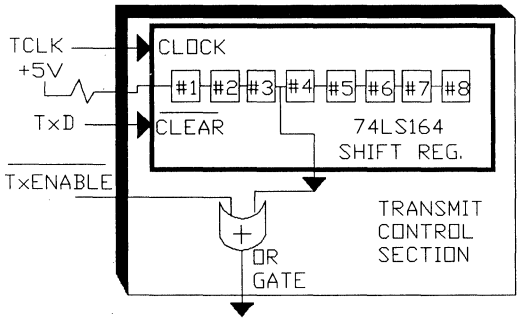


Figure 3.6 Realization of the transmit control section.

Since TxD is a Manchester code, the TxD could not be at 1 for more than two TCLK clocks during the transmission of the frame. Thus, during the transmission of the frame, the 1 could be pushed as far as the second cell, and then when the TxD goes to 0, all the cells are filled with 0s again.

At the end of the frame, TxD remains 1 continuously, which causes the third cell to be filled with 1 after three consecutive 1s. This 1 causes the OR gate to output a 1 which disables the transmitter. Thus, we met the requirement of disabling the 26LS30 only after TxD remains at 1 for a while at the end of the frame.

Figure 3.7 reflects the choice of hardware components for the implementation of the transmit and transmit control sections of the 802.3 Starlan node board.

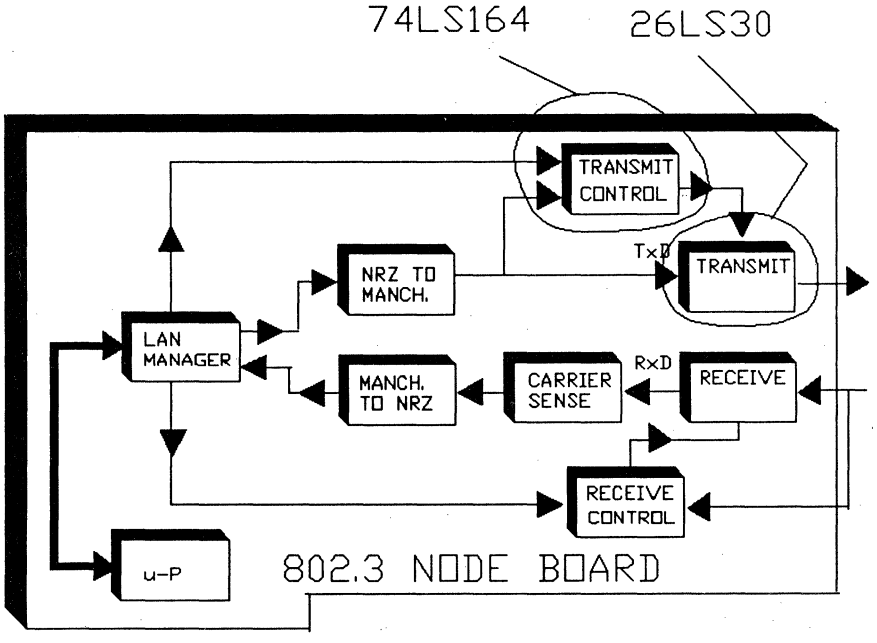


Figure 3.7 Realization of the transmit and transmit control sections of the node board.

3.3 The Receive and Receive Control Sections of the IEEE 802.3 1BASE5 Starlan Node

3.3.1 The receive circuit of the node

The signal received by the receive circuit of the node is the signal transmitted by the transmit circuit of the hub board. The transmit circuit of the hub board is identical to the transmit circuit of the node. Thus, the receive circuit of the node is receiving a differential signal generated by a 26LS30.

As shown in Fig. 1.15, the incoming signal from the interface board is fed to the receive circuit. The receive circuit of Fig. 1.15 is shown in greater detail in Fig. 3.8. The objective is to generate the RxD signal from the incoming differential signal and to supply the RxD signal to the next sections of the node. The receive circuit must generate RxD = "1" whenever there are no incoming data. Whenever the node senses a continuous 1 on its RxD, it concludes that the network is free (nobody else is transmitting).

Recall that during the reception of data, the incoming data from the network cannot be at continuous 1 since the incoming data are in

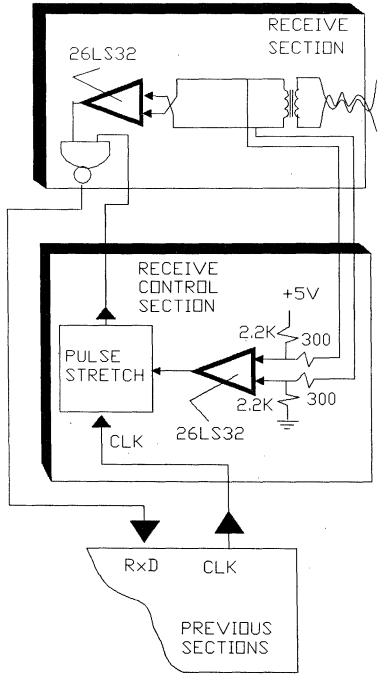


Figure 3.8 The receive and receive control circuits.

Manchester format, which cannot be at 1 for more than two consecutive bits.

The device utilized to convert the incoming differential signal to a single-ended signal is the 26LS32 (which can be thought of as the counterpart of the 26LS30 utilized in the transmit circuit). The 26LS32 is also called an RS-422 receiver since it is utilized as a receiver in RS-422 applications. The operation of the 26LS32 is shown in Fig. 3.9. As shown, the 26LS32 simply serves as a zero crossing device. That is, it outputs a 1 whenever the differential signal is positive, and outputs a 0 whenever the differential signal is negative. The transition occurs whenever the input signal crosses the 0-V line. We refer to the signal V_{out} of Fig. 3.9 as the true representation of the differential signal V_{in}.

By reversing the wires to the inputs of the 26LS32, the output of the 26LS32 is inverted as shown in Fig. 3.10. Now the output of the 26LS32 is 0 for positive differential input and is 1 for negative differential inputs. We refer to the signal V_{out} of Fig. 3.10 as the inverted representation of the differential signal V_{in}.

Example 3.6 Some designers choose to include a filter in the receive circuit. Figure 3.11 shows a filter that is designed to filter out high-frequency noise. The resistor R_m is used for impedance matching purposes. That is, the resistor value is chosen so that most of the power of the incoming signal is delivered to the transformer, not to the distributed resistance of the wires that connect the hub board to the node board. A typical value of R_m is 110 Ω. The capacitor C, inductor L, and resistor R are connected to the secondary of the transformer. These components form the filter, rejecting the high-frequency noise in the incoming signal. A value 150 μF for the capacitor, 56 μH for the inductor, and 698 Ω for the resis-

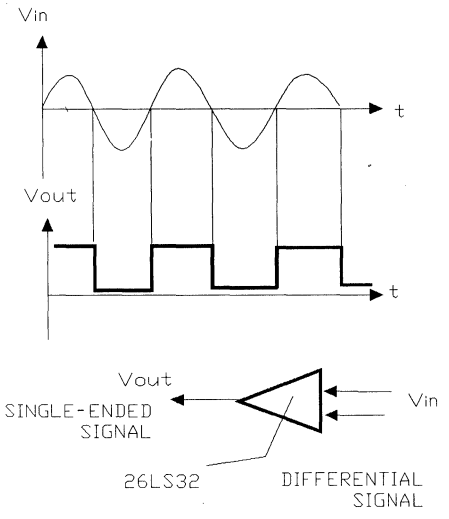


Figure 3.9 Using the 26LS32 as differential to single-ended converter.

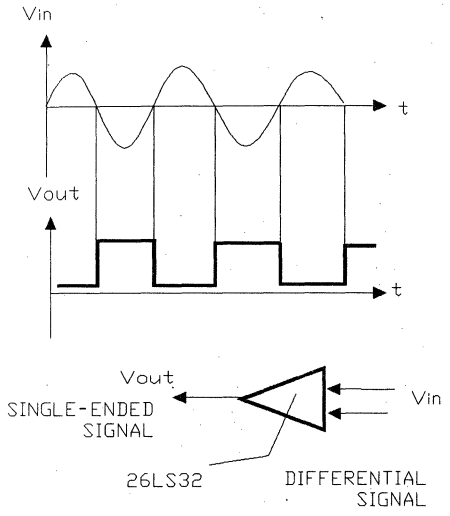


Figure 3.10 The polarity reversal of differential to single-ended signals.

tor consist of a 4-MHz, two-pole Butterworth filter, which is the recommended filter according to the 1BASE5 protocol.

3.4 Squelching the Incoming Signal

The signal shown in Fig. 3.12*a* is a typical incoming signal to the input pins of the 26LS32. The incoming signal is shown as rounded and full of distortions and ripples (not a clean trapezoidal signal). These distortions and rounding occur due to the fact that the signal is traveling over long wires and passing through the transformers.

Note that there are several voltage variations shown as noise in the incoming signal of Fig. 3.12*a*. Indeed, it is very unlikely that the signal presented to the inputs of the 26LS32 would be at absolute zero voltage when there is no transmission. These fluctuations around the 0-V line are expected due to switching on nearby wires and other noise sources.

The signal shown in Fig. 3.12*b* is the output generated at the output pin of the 26LS32 in response to the input shown at Fig. 3.12*a*. The 26LS32 responds to each zero crossing of the inputs. That is, when the

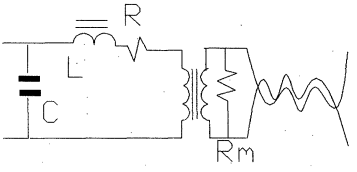


Figure 3.11 Filtering the incoming signal.

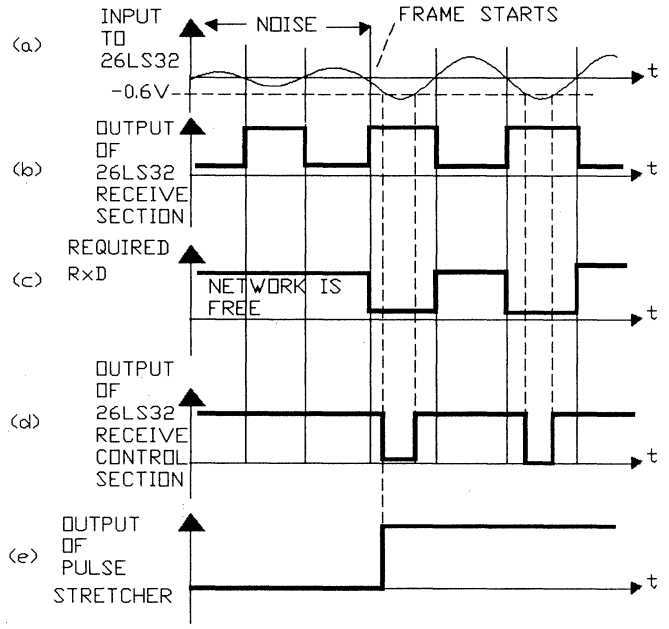


Figure 3.12 The signal processing of the incoming signal.

differential input is positive, the output of the 26LS32 is 0, and when the differential input is negative, the output of the 26LS32 is 1. Unfortunately, the 26LS32 responds to the small noise voltage variations as well, since these also consist of a zero crossing input signal. If we were to feed this signal to the RxD, this node would never transmit (since it would never observe a continuous 1 on its RxD).

The required signal to be fed to the RxD is shown in Fig. 3.12*c*. This signal is the inverted version of that shown in Fig. 3.12*b* and with the noise removed. This is indeed the required RxD, since during the no-transmission period (when the network is free), the RxD is at a continuous 1.

It is the responsibility of the receive control circuit of Fig. 3.8 to pass only the good pulses to the RxD. The receive control circuit is also called the *squelching circuit.*

Example 3.7 Figure 3.12*a* shows that the first good bit of the received differential frame swings toward the negative direction. Explain.

Solution Indeed, the very first bit of the frame is the first 0 of the preamble field, which is the first 1 to 0 transition in the frame.

3.5 The Receive Control Circuit

The single-ended output of the 26LS32 (which is an inverted version of

the incoming signal) is fed to one of the inputs of a NAND gate (Fig. 3.8). If the other input to the NAND is 1, the single-ended signal is inverted by the NAND gate and then fed to the RxD. Since the incoming signal is inverted twice, RxD is fed with the true representation of the incoming signal.

If the signal coming from the receive control circuit to the second input of the NAND gate is a 0 during the duration of the noise, the RxD sees a continuous 1, as required whenever there are no incoming data.

Figure 3.8 shows that the incoming signal from the hub is fed to the receive control circuit as well. Due to the pairs of bias resistors, the 300- and 2200-Ω resistors, the 26LS32 of the receive control circuit generates a pulse at its output only for incoming signals that are more negative than 0.6 V. This makes the 26LS32 of the receiver control circuit not a zero crossing detector, but a –0.6-V crossing detector. The output of the 26LS32 of the receive control circuit is shown in Fig. 3.12*d* to change from 0 to 1 and from 1 to 0 only when its input crosses the –0.6-V line. The reason for setting the bias at –0.6 V is that in IEEE 802.3 1BASE5 Starlan, the incoming signals are specified to have a minimum swing of ±0.6 V.

Example 3.8 Explain the 300-Ω and 2.2 -kΩ resistors values (Fig. 3.8).

Solution The 300-Ω and 2.2 -kΩ resistors form a voltage divider, dividing the +5 V among the 300-Ω and 2.2-kΩ resistors:

$$\text{DC voltage across 300-}\Omega\text{ resistor} = \frac{5 \times 300}{300 + 2200} = 0.6\text{ V}$$

The *pulse stretcher* circuit of the receive control circuit (Fig. 3.8) has the responsibility of generating a 1 to the input of the NAND gate upon observing that the 26LS32 of the receive control circuit has become more negative than –0.6 V. That is, upon receiving the first bit of the incoming frame, the pulse stretcher is supposed to keep supplying a 1 to the NAND gate until the end of the frame. The frame is fully received when there are no more incoming signals with a voltage less than –0.6 V.

The output of the pulse stretcher is shown in Fig. 3.12*e*. Since the inputs of the NAND are the signals shown in Fig. 3.12*e* and *b*, the output of the NAND gate in the signal shown in Fig. 3.12*c* as required.

Example 3.9 Construct the pulse stretcher circuit of Fig. 3.8 by using an up-down counter.

Solution One possible circuit is shown in Fig. 3.13. The circuit makes use of the CLK signal that happens to be available from the previous sections on the node board. As we shall see, this signal is a clock used as the system clock of the node. The bits coming out of TxD and into RxD have pulse widths that are dictated by the data rate of the LAN protocol. The pulse widths of the TxD and RxD bits are multiple integers of the CLK clock. Since the incoming signal from the network is a Manchester signal, it must change from 1 to 0 or from 0 to 1 after a certain

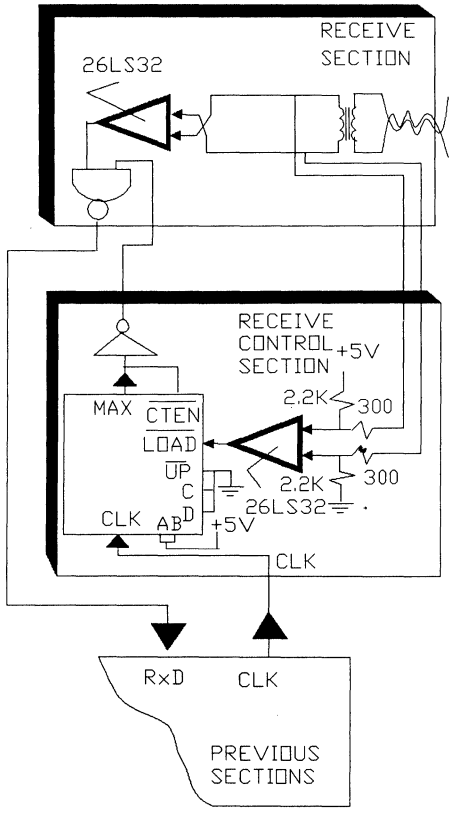


Figure 3.13 Realization of the receive and receive control circuits.

maximum number of CLK pulses. The implementation below assumes that during the duration of the frame, the incoming signal contains a transition to 0 within a maximum period of 12 CLK clocks.

The 74LS191 counter is utilized for implementing the pulse stretcher. This chip counts from a certain number up to 15 in accordance with the clock supplied to its CLK pin.

The CLK clock from the previous sections is connected to the clock of the 74LS191 up-down counter.

The 74LS191 is capable of counting up or down. Since we connect its up pin to ground, the counter counts up. The counting is being performed if the chip is enabled. To enable the chip, a 0 must be applied to the count enable (CTEN) pin.

The 74LS191 has four input pins, D, C, B, and A. These pins determine the starting point of the count. We connect pins D and C to ground, and pins A and B to V_{cc}. This setting consists of the decimal number 3 (i.e., DCBA = 0011 = 3). This setting of the A, B, C, and D pins causes the counter to count from 3 to 15 (15 is the maximum count that the 74LS191 can count).

When the 74LS32 receives at its input a signal more negative than –0.6 V, a 0 is supplied to the load pin of the 74LS32. This causes the 74LS191 to start counting up, starting at 3. As long as the counter did not reach its maximum count of 15, the maximum count (MAX) pin remains at 0. This 0 is inverted and supplied

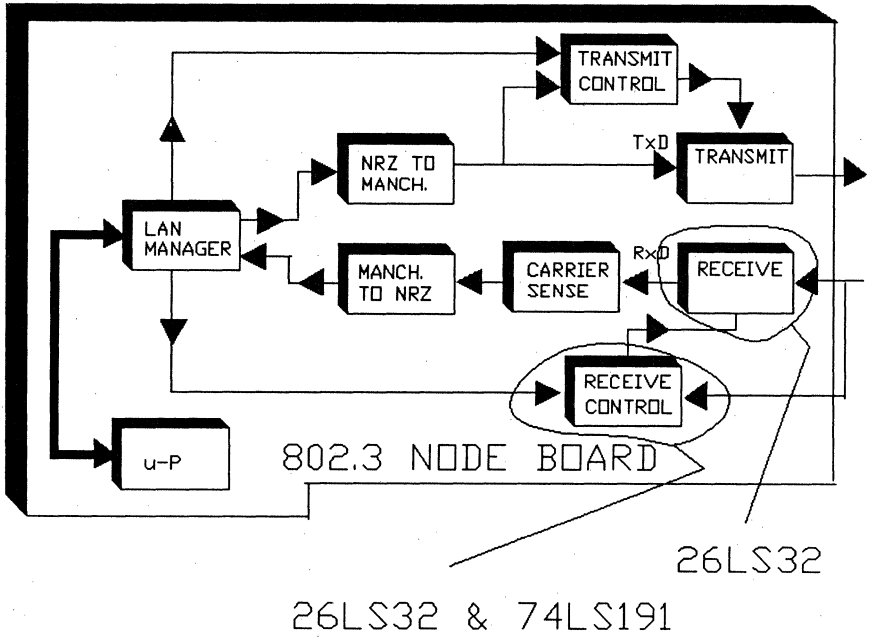


Figure 3.14 The realization of the receive and receive control section.

to the NAND gate. The signal to the other input of the NAND gate is therefore inverted and passed to the RxD.

The counter keeps counting up, but before it reaches its maximum count, a new 0 is applied to the load pin, which starts the count all over again starting from 3. Thus, the maximum count (MAX) output pin remains at 0. The maximum count pin is also connected to the count enable (CTEN) pin, which enables the counter to perform the counting (i.e., supplying a 0 to the CTEN pin whenever the count is less than 15).

The only time the maximum count pulse could be at 1 is when the counter is able to reach its maximum count. This means that a new pulse to the load pin did not arrive. The new pulse to the load pin did not arrive since the frame was fully received already.

We set the pins A, B, C, and D to decimal 3, since we want the counter to count 12 pulses (15 – 3 = 12) before reaching its maximum count.

Figure 3.14 reflects the choice of hardware components for the implementation of the receive and receive control sections of the 802.3 Starlan node board.

3.6 Summary

This chapter discusses the hardware implementation of the transmitter and receiver circuits of the 1BASE5 Starlan node board. These circuits are designed to transmit (and receive) differential signals over twisted pairs of wires. As demonstrated, the implementation consists of fairly simple and inexpensive circuitries.

Chapter

4

The LAN Manager Section of a Node in 1BASE5 LAN

Now that the transmit, transmit control, receive, and receive control sections of the Starlan node have been discussed, we can proceed with the discussion of the implementation of the rest of the sections of the 802.3 node board (see Fig. 1.15).

Although our discussion is oriented toward the implementation of a Starlan node board by using the 82588 LAN controller chip, the discussion illustrates a general design scheme that is applicable when using other LAN controller chips for the implementation of the Starlan as well as for the implementation of other 802.3 node boards.

4.1 The Manchester Decoder-Encoder and the Carrier Sense Sections

The Manchester decoder-encoder and the carrier sense sections serve as the interface between the transmit-receive sections and the LAN manager section (see Fig. 1.15).

4.1.1 The microprocessor section

The microprocessor section (Fig. 1.15) serves as the interface between the PC and the node board. When the PC instructs the microprocessor to enable the node for the reception of frames from the network, the microprocessor responds by preparing a place for the expected frames in its RAM chips and instructs the LAN manager to start reception. Once frames are received and processed by the LAN manager, the microprocessor interrupts the PC and transfers to it the received frames.

When the PC has data to be transmitted, it interrupts the microprocessor, stores the data in the RAM chips of the microprocessor section, and instructs it to transmit the data. The microprocessor responds by rearranging the data in a form acceptable by the LAN manager and then instructs the LAN manager to transmit the data.

4.1.2 The LAN manager section

The LAN manager section of the node (Fig. 1.15) is the brain of the node. It is responsible for executing and managing all the procedures and data processing outlined in the LAN protocol. Evidently, a sophisticated processor is required for such a job.

Fortunately, the Intel corporation designed a family of processors known as LAN controllers, designed specifically to be utilized in LAN applications. These chips are designed to perform all the required LAN procedures outlined in the LAN protocol. As we shall soon see, these LAN processors are capable of performing the CRC calculations, address recognition, frame construction, and so on.

Best of all, the program responsible for executing the LAN procedures is already embedded on the LAN controller chip. When the microprocessor has data to transmit, it interrupts the LAN controller and commands it to transmit. All by itself, the LAN controller accesses the RAM chips, fetches the data, constructs a frame, and transmits the frame in accordance with the LAN protocol. Similarly, once the microprocessor commands the LAN controller to start reception of frames, the LAN controller is on its own, it processes the received frames and stores the received data in a preagreed location in RAM, all without any help from the microprocessor.

The node board of Fig. 1.15 is shown again in Fig. 4.1. The diagram indicates our choice of hardware components for the implementation of the Starlan node board. As indicated, the 82588 has the capability of accomplishing the tasks of the LAN manager section, the NRZ to Manchester section, the NRZ to Manchester section, and the carrier sense section, all on a single chip.

Figure 4.1 also indicates that the microprocessor chosen for the implementation is the Intel 80188 microprocessor.

4.1.3 Using the 80188 as the host processor

The reasons for choosing the 80188 as the microprocessor are

1. As we shall soon see, in order to access the RAM chips for the purpose of fetching the data to be transmitted from the RAM, and for the purpose of storing the received frames in the RAM, the 82588 utilizes direct memory access (DMA) controllers. The 82588 could utilize one

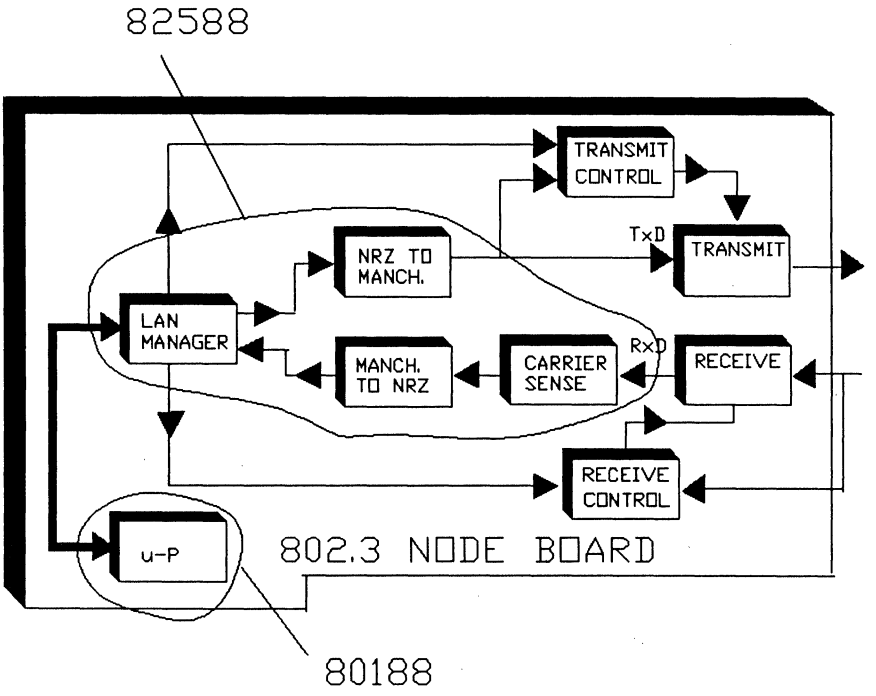


Figure 4.1 Realization of the node board with off-the-shelf ICs.

DMA controller, but two DMA controllers would be even better. The 80188 has two independent DMA controllers on-chip, just what we need.

2. As we shall soon see, it takes a straight pin-to-pin connection to interface the 82588 to the 80188.

4.2 A Quick Review of the Intel 80188 Processor

Since many of the design applications introduced in this book utilize the Intel 80188 microprocessor, a quick review of this processor is in order.

The Intel 80188 is an 8-bit high-integration microprocessor. It is called a high-integration processor, since a variety of peripheral devices are integrated into the chip. The 80188 processor (together with ROM and RAM chips) is shown in Fig. 4.2.

4.2.1 A quick review of the peripheral chip select pins of the 80188

The 80188 is equipped with a set of seven pins called the *peripheral chip select* pins (PCS0, PCS1, PCS2, PCS3, PCS4, PCS5, and PCS6).

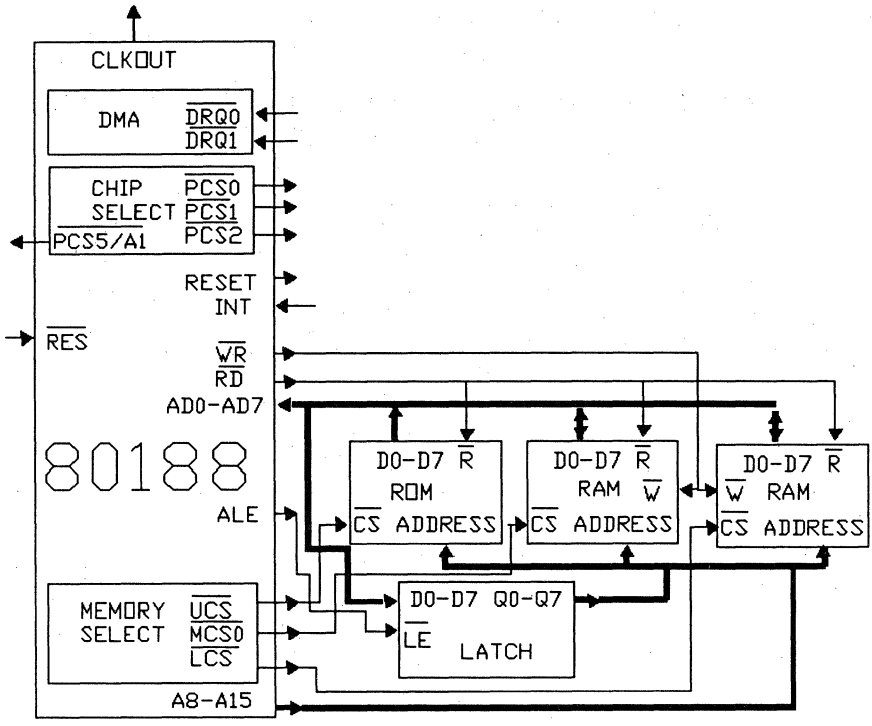


Figure 4.2 Connecting memory (RAM and ROM) to the 80188.

The 80188 can be programmed to use these pins as address decoders for peripheral devices.

Upon reset, all the peripheral chip select pins are set to 1.

Example 4.1 Show how a peripheral device can be connected to the 80188 and decoded to address location 01000 using the PCS0 pin of the 80188.

Solution The chip select (CS) pin of the peripheral device is connected to the PCS0 pin.

Upon initialization, the program of the 80188 writes into the proper internal registers of the 80188, causing the 80188 to associate the PCS0 with address location 01000. Thereafter, whenever the 80188 encounters an instruction to read a byte from address 01000, or to write a byte to address 01000, the pin PSC0 is being asserted automatically.

The read (RD) and write (WR) pins of the 80188 have to be connected to the read (R) and write (W) pins of the peripheral device.

4.2.2 The PCS5/A1 and PCS6/A2 pins of the 80188

Two of the peripheral chip select pins, the PCS5 and PCS6 pins of the 80188, can be programmed to serve as latches. They are denoted as PCS5/A1 and PCS6/A2.

Example 4.2 Show how the PCS5/A1 pin can be utilized as a latch.

Solution Upon reset, all the peripheral chip select pins are set to 1. Thereafter, the program of the 80188 may set the PCS5/A1 pin to a 0. The pin stays (latched) at 0 until the program sets it back to 1.

4.2.3 A quick review of the memory chip select pins of the 80188

The 80188 is equipped with a set of output pins called the memory chip select pins. The 80188 can be programmed to use these pins as address decoders for memory devices (ROM and RAM chips). These pins are

- The lower memory chip select pin (the LCS pin). This pin is assigned to a range of addresses in the lower portion of the memory (i.e., starting at address 00000).
- *The upper memory chip select pin (the UCS pin).* This pin is assigned to a range of addresses in the upper portion of the memory.
- *The memory chip select pins (the MCS0, MCS1, MCS2, and MCS3 pins).* These pins are assigned to addresses in between the LCS and the UCS.

Example 4.3 Show how ROM and RAM chips may be decoded by the various memory chip select pins of the 80188.

Solution In Fig. 4.2, the CS pin of the ROM is connected to the UCS pin of the 80188, the CS pin of one of the RAM chips is connected to the MCS0 pin of the 80188, and the CS pin of the other RAM chips is connected to the LCS pin of the 80188.

Upon initialization, the program of the 80188 writes into the proper internal registers of the 80188, causing the 80188 to associate each of the memory chip select pins with a certain address location range.

Thereafter, whenever the 80188 encounters an instruction to read a byte from a certain address or to write a byte to a certain address, the memory chip select pin that is associated with that address is being asserted automatically.

The read (RD) pin of the 80188 is connected to the read (R) pins of the memory chips, and the write (WR) pin of the 80188 is connected to the write (W) pins of the memory chips.

4.3 Connecting ROM and RAM Chips to the 80188

The address of the 80188 is composed of 20 address bits, A0 to A19, and 8 data bits, D0 to D7. The eight data bits D0 to D7 share the same pins as the eight least-significant address bits, A0 to A7. The shared pins are called AD0 to AD7. (Figure 4.2 shows only address pins A8 to A15, since in our application we are using only these pins.)

When the 80188 encounters an instruction in its program to read a byte from a certain address location in memory or to write a byte into a certain memory location, it executes the command in two stages:

1. During the first stage, the 80188 outputs the eight least-significant bits of the address specified in the instruction into the AD0 to AD7 pins, and then asserts its output pin address latch enable (ALE). As shown in Fig. 4.2, a latch chip is connected to the 80188. The AD0 to AD7 are latched into the latch during the first stage [since the ALE pin is connected to the latch enable (LE) pin of the latch]. The A0 to A7 bits are therefore available at the output of the latch at the end of stage 1.
2. During the second stage of the execution of the instruction, the 80188 outputs the rest of the address bits of the address specified in the instruction into pins A8 to A15, and uses the AD0 to AD7 pins as data pins. That is, if the instruction is to read a byte, the AD0 to AD7 pins become input pins, and if the instruction is to write a byte, the AD0 to AD7 pins become output pins.

The address bus is composed of bits A0 to A7 available from the output of the latch, and the A8 to A15 bits, available from the A8 to A15 pins. The address bus is available during stage 2. The data bus is composed of the D0 to D7 bits available from the AD0 to AD7 pins and is also available during stage 2.

The address and data bus of the 80188 are connected to the corresponding address and data pins of the ROM and RAM chips.

4.4 A Quick Review of the On-Chip DMA Controllers of the 80188

The 80188 has two on-chip DMA controllers designated as channel 0 and channel 1. Each of these DMA controllers can be programmed to transfer data from one location in memory (the source) to another location in memory (the destination).

Prior to the transfer of a block of data, the program of the 80188 writes into the registers of the corresponding DMA controller the following information:

- The starting address of the block of data to be transferred (the source).
- The starting address of the destination address.
- The number of bytes to be transferred.

The 80188 has an input pin called data request 0 (DRQ0), which is associated with DMA controller 0, and an input pin called DRQ1, which is associated with DMA controller 1 (Fig. 4.2).

Once the registers of the DMA controllers are programmed, an external device could initiate the data transfer by asserting the proper data request pins of the 80188. Thus, if the registers of DMA channel 0 are programmed, and an external device asserted the DRQ0 pin of the 80188, data are transferred as specified in the registers of DMA controller 0.

Likewise, if the registers of DMA channel 1 are programmed, and an external device asserted the DRQ1 pin of the 80188, data are transferred as specified in the registers of DMA controller 1.

In most applications, the external device that requests the data transfer requires to see an acknowledgment signal, indicating that the data transfer is being executed. DMA controller devices are usually equipped with data transfer acknowledge output pins that are asserted by the DMA controller when the data transfer is being executed.

Example 4.4 Unfortunately, the 80188 does not have data acknowledgment output pins. A data request acknowledgment signal can be generated by using the peripheral chip select pins of the 80188. To cause a peripheral chip select pin to be asserted when a byte transfer is being executed, we need to program the address range of the peripheral chip select pin to the same address range of the data transfer specified in the register of the DMA controller. Thus, whenever the DMA controller executes the byte transfer, the peripheral chip select pin is asserted since it is programmed to be asserted to the addresses that the DMA controller accesses.

4.5 The Rest of the 80188 Pins

The rest of the pins shown in Fig. 4.2 are

- *The RESET output pin.* Whenever the 80188 is reset, this output is asserted. Thus, if this pin is connected to the reset input pin of an external device, the external devices are reset whenever the 80188 is reset.
- *The INT input pin.* The 80188 has several interrupt input pins. In our application we need a single interrupt pin. Whenever an external device requires the attention of the 80188, it asserts the INT pin of the 80188.
- *The RES input pin.* The 80188 is being reset by asserting this pin.
- *The CLKOUT output pin.* The clock of the 80188 is available at this pin to be utilized by other external devices.

Example 4.5 What is the frequency of the signal CLKOUT?

Solution Just like any other processor, the 80188 also requires a clock for its operation. The frequency of operation of the 80188 is half the frequency of a clock which has to be applied to the X1 and X2 pins of the 80188. As shown in Fig. 4.3, we connect a 16-MHz crystal to the X1 and X2 pins of the 80188. This causes the 80188 to operate at a frequency of 8 MHz (16 MHz/2 = 8 MHz).

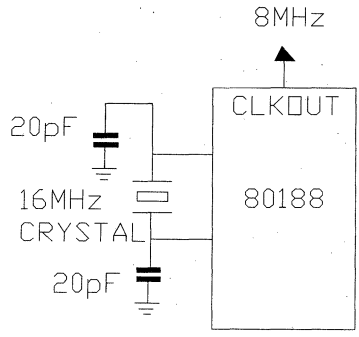


Figure 4.3 Generating 8-MHz clock.

The CLKOUT pin output pin of the 80188 in Fig. 4.3 is therefore an 8-MHz clock.

Example 4.6 The 80188 is equipped with a pin called *nonmaskable interrupt* (NMI). Although we are not utilizing this pin for the 80188-82588 interface in our design, this interrupt pin may be utilized by the PC to interrupt the 80188. If the pin is not used, it should be connected to ground to negate its operation. Other input pins of the 80188 which are not used should be negated as well (e.g., the HOLD pin should be grounded if it is not used).

4.6 The High Integration Mode and High Speed Mode of the 82588

The 82588 LAN controller can work in one of two modes, the high integration mode and the high speed mode. When the 82588 is programmed to work in the high speed mode, it is capable of transmitting and receiving frames to and from the network at a data rate of up to 5 Mbit/s. While having this impressive high data rate capability, several tasks specified in the Starlan protocol are not implemented by the 82588 in the high speed mode, and therefore external components must be designed to accomplish tasks not performed by the 82588. For example, the collision detection capability of the 82588 is not implemented in the high speed mode. Thus, if the high speed mode is chosen, a collision detection circuit must be implemented as an external circuit to the 82588.

As suggested by Fig. 4.1, we choose to utilize the 82588 in its high integration mode, incorporating all external circuits to the 82588. In the high integration mode, the maximum data rate of transmitting and receiving frames to and from the network is only 2 Mbit/s. However, in this slower mode, all the tasks specified in the Starlan protocol are performed on-chip, and no additional external components are required.

Since the 1BASE5 Starlan specifies 1 Mbit/s data rate of transmitting and receiving frames, the high integration mode of the 82588 has sufficient speed to accommodate the protocol.

4.7 The 82588-80188 interface

This section is included to provide the reader with a "design know-how" of interfacing the 82588 LAN controller with the 80188 processor. We enclosed "design know-how" in quotes, since this know-how consists of simple, straight, pin-to-pin connections between the two devices. Figure 4.4 illustrates that the interface between the 80188 and the 82588 indeed consists of a straight pin-to-pin connection.

The 82588 has two interface ports, the system interface port (which interfaces the 82588 with the processor) and the serial interface port (which interfaces the 82588 with the transmit-receive sections).

4.7.1 The system interface port

The 82588 has a *data transfer request channel #0* output pin, the DRQ0 pin. This pin is asserted whenever the 82588 requires data transfer to be preformed by DMA #0 of the 80188. As shown in Fig. 4.4, the DRQ0 pin of the 82588 is directly connected to the DRQ0 pin of the 80188. When asserting the DRQ0 pin, the 82588 assumes that the DMA0 of

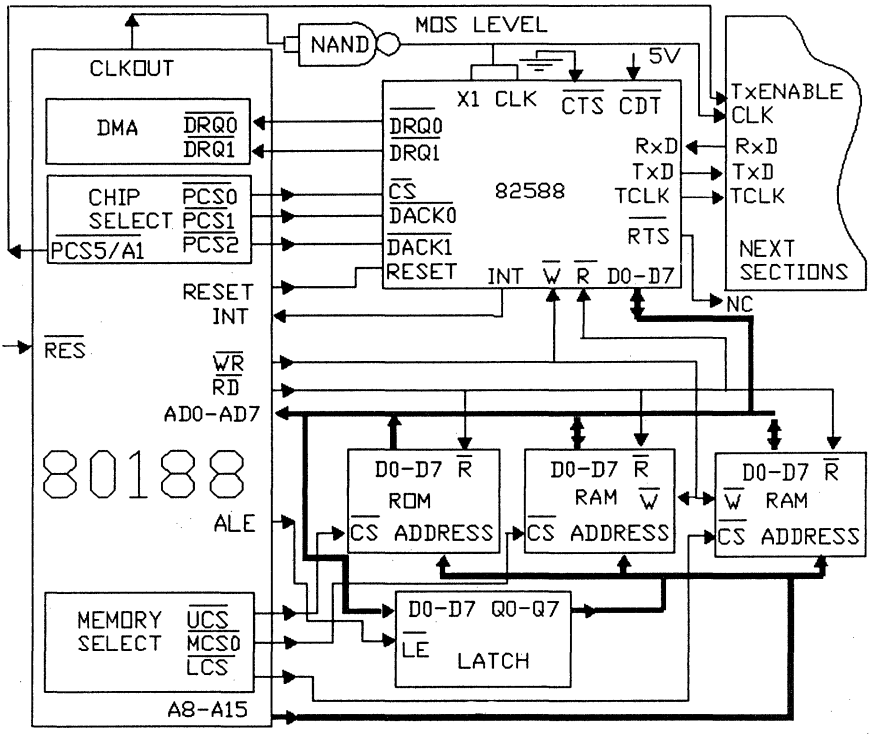


Figure 4.4 Connecting the 80188 to the 82588.

the 80188 was previously programmed (by the 80188) for the required data transfer.

The 82588 has a second data transfer request output pin, the DRQ1 pin. This pin serves the same function as the DRQ0 pin, except that it utilizes channel 1 (i.e., it is requesting the service of DMA1 of the 80188).

The 82588 has a data transfer acknowledgment channel #0 input pin, the DACK0 pin. Whenever the 82588 requests a data transfer from DMA #0, it expects to see its DACK0 input pin asserted as an acknowledgment signal, indicating that this data transfer is being executed. As shown in Fig. 4.4, we directly connect the DACK0 pin of the 82588 to the peripheral chip select #1 (PCS1) pin of the 80188. The 82588 assumes that the PCS1 pin was previously programmed (by the 80188) to serve as a data transfer acknowledgment of DMA #0.

The DACK1 input pin of the 82588 serves the same function as the DACK0 pin, except that it utilizes channel 1 (i.e., it is the acknowledgment for the data-transfer request from DMA1 generated by the DRQ1 pin).

The 82588 has a RESET input pin that, when asserted, causes the 82588 to be reset. As shown in Fig. 4.4, the RESET output pin of the 80188 is directly connected to the RESET input pin of the 82588. Thus, we can be sure that the 82588 is reset whenever the 80188 is reset.

As we shall soon see, the 80188 has to access the 82588 for the purpose of writing and reading bytes into and from the 82588. Such an access can be accomplished provided that the chip select (CS) input pin of the 82588 is asserted. The peripheral chip select #0 (PCS0) of the 80188 is programmed for the address of the 82588. Thus, the two pins are connected directly.

Whenever bytes are written into the 82588 or read from the 82588, the data are presented to the D0 to D7 pins of the 82588. These pins are therefore connected to the data bus of the 80188.

Example 4.7 The Intel Corporation has a microprocessor known as the 80186 microprocessor. The 80186 is almost identical to the 80188, except that the 80186 has a 16-bit data bus, D0 to D15. Since the 82588 has an 8-bit data bus, it makes more sense to use the 80188, which was especially designed to accommodate 8-bit devices.

As we shall soon see, the 82588 needs to interrupt the 80188. It does so by asserting its output INT pin. The INT pin of the 82588 is directly connected to one of the input interrupt pins of the 80188.

The 82588 requires a clock to perform its microprocessor duties. In our design, this clock is supplied to the 82588 from the CLKOUT output pin of the 80188.

Example 4.8 Figure 4.4 shows that the CLKOUT signal is supplied to the CLK pin of the 82588, as well as to the NEXT SECTIONS. Explain.

Solution This CLK signal is the signal supplied to the receive control circuit. It is applied to the pulse stretcher as shown in Fig. 3.8.

Note that in our design, the 80188 outputs 8 MHz to the CLKOUT pin.

The read (R) pin of the 82588 has to be asserted whenever bytes are read from the 82588. This pin is directly connected to the RD pin of the 80188.

The write (W) pin of the 82588 has to be asserted whenever bytes are written into the 82588. This pin is directly connected to the WR pin of the 80188.

4.8 The Serial Interface Port of the 82588

This section deals with the rest of the 82588 pins. These pins are related to the serial (network) operation of the 82588. The 82588 LAN controller is equipped with the proper pins to allow a straight pin-to-pin connection to the transmit, transmit control, receive, and receive control sections discussed in Chap. 3.

Following is a brief description of each of the serial interface pins of the 82588. Other LAN controller chips have identical or similar pins.

4.8.1 The TxD pin

The TxD pin is the pin by which data are transmitted from the 82588.

Example 4.9 Figure 4.4 shows that the TxD signal of the 82588 is supplied to the TxD of the NEXT SECTIONS. Explain.

Solution The TxD signal is the signal supplied to the transmit and transmit control circuits. It is applied to these circuits as shown in Fig. 3.3.

4.8.2 The X1 and X2 pins

The 82588 is equipped with two pins called the X1 and X2 pins. (Only the X1 pin is shown in Fig. 4.4, since in our design we leave the X2 pin floating.) The 82588 has an internal clock called the *serial clock*. The serial clock dictates the rate at which bits are outputted to its TxD pin. The frequency of the serial clock is derived internally by the 82588 from a clock applied to its X1 pin. The clock applied to the X1 pin needs to have 8 or 16 times the desired frequency of the serial clock.

Example 4.10 In Fig. 4.4 we supply the X1 pin of the 82588 with a clock from the CLKOUT pin of the 80188, which is an 8-MHz clock. Does this conform with the requirements of a 1BASE5 Starlan node board?

Solution Since in 1BASE5 Starlan, data are transmitted at a rate of 1 Mbit/s, the clock supplied to the X1 pin should be either an 8- or 16-MHz clock.

Upon booting and configuring the 82588, the 80188 has to send a configuration parameter to the 82588, indicating whether the 82588 should divide the clock applied to its X1 by 8 or by 16.

Since in our design we apply an 8-MHz clock (from the CLKOUT pin of the 80188) to the X1 pin of the 82588, the 80188 must configure the 82588 to generate the serial clock by dividing the X1 clock by 8. This results in a serial clock of 1 MHz, which is the required serial clock in 1BASE5 Starlan.

Example 4.11 Explain the use of the NAND gate in Fig. 4.4.

Solution The 82588 requires that a MOS-level voltage be applied to its X1 pin. A MOS-level voltage is 0.6 to 3.9 V. The NAND gate shown in Fig. 4.4 is the 74HCT00, which provides the X1 pin with the MOS-level voltage as required.

Thus, the gate is used as a level converter (from TTL to MOS), not as a logic gate.

4.8.3 The serial clock

The serial clock of the 82588 is available at the TCLK pin.

Example 4.12 Figure 4.4 shows that the TCLK signal of the 82588 is supplied to the TCLK of the NEXT SECTIONS. Explain.

Solution The TCLK signal is the signal supplied to the transmit control circuit. It is applied to the idle signal generator circuit as shown in Fig. 3.3.

4.8.4 The RxD pin

The bits from the receive circuit are received by the RxD pin of the 82588. This is the same RxD described in Fig. 3.8.

4.8.5 The collision detect pin

The 82588 is equipped with a collision detect (CDT) input pin. When this pin is asserted (when a 0 is applied to it), the 82588 interprets it as if there were a collision in the network and responds accordingly. This pin is utilized in applications where the 82588 is configured in the high speed mode. In the high speed mode, an external circuitry is required for sensing the collision. The output of the external collision detection circuit is then applied to the CDT pin of the 82588, informing it whether or not there is a collision. However, in our design, we utilize the 82588 in the high integration mode, and therefore the collision detect circuitry is implemented internally on the 82588 chip. In this case, the CDT is applied with a constant 1 as shown in Fig. 4.4.

4.8.6 The CTS pin

As shown in Fig. 4.4, we connect the *clear to send* (CTS) input pin of the 82588 to ground. When asserted (when 0 is applied to it), the CTS pin

enables the 82588 to transmit bits to its TxD pin. If a 1 is applied to the CTS pin, the 82588 cannot transmit bits to its TxD pin. In our design, we apply a constant 0 to the pin (constantly enabling transmission), since we choose not to take advantage of this feature.

4.8.7 The RTS pin

The 82588 is equipped with a *request to send* (RTS) output pin. As shown in Fig. 4.4, our design does not make use of this pin (the pin is left floating). The 82588 asserts this pin whenever it has data to transmit.

Example 4.13 Although the design shown in Fig. 4.4 does not utilize the RTS output pin, this example shows how the RTS may be utilized in other applications.

Usually, the RTS is used together with the CTS pin. Initially, the external device supplies a 1 to the CTS pin of the 82588, disabling the transmission, and the 82588 outputs a 1 to the RTS pin, indicating that it has nothing to transmit.

Once the 82588 has data to transmit, it asserts its RTS pin, requesting permission to transmit from an external device.

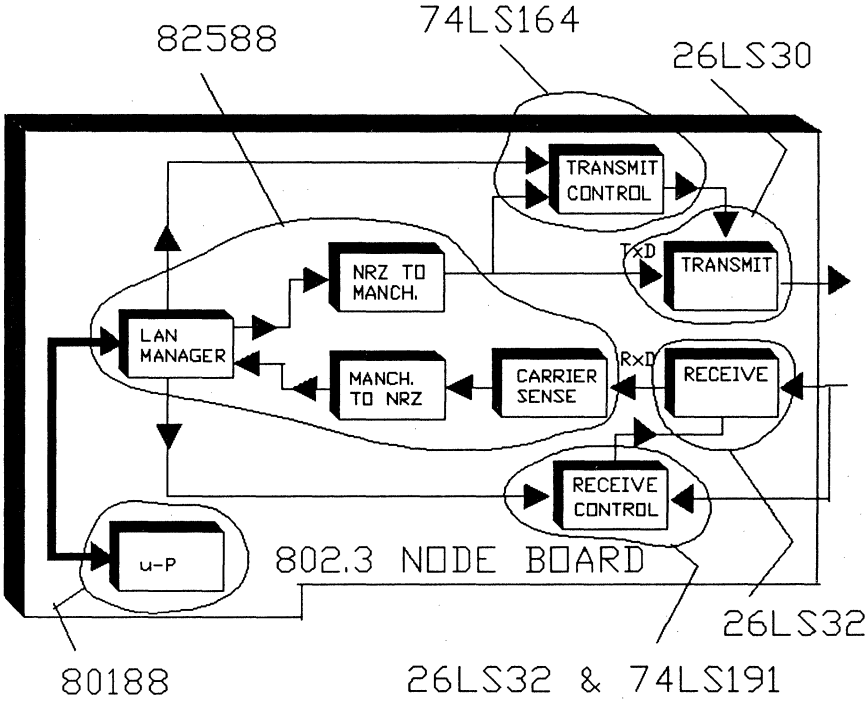


Figure 4.5 Complete realization of the Starlan node board.

Once the external device receives a 0 on the RTS line, it realizes that the 82588 has data to transmit. If the external device is not ready to receive the data, it denies the transmission, by simply leaving the CTS line at 1. However, if the external device is ready to receive the data, it asserts the CTS pin. Upon receiving a 0 on its CTS pin, the 82588 starts the transmission.

4.8.8 The TxENABLE generation

Figure 4.4 shows how the TxENABLE signal to the transmit control circuit of Fig. 3.3 is generated. That is, upon reset, the PCS5/A1 pin is at 1, which is the required logic to disable transmission. Once the 80188 completes configuring the 82588, it enables transmission by latching a 0 onto the PCS5/A1 pin.

4.8.9 A complete 1BASE5 Starlan node board

This concludes the discussion of the hardware pins of the 82588. Figure 4.5 reflects the choice of hardware components for the implementation of the complete 802.3 Starlan node board.

4.9 Summary

This chapter discusses the hardware implementation of the LAN manager section of the Starlan node board. The main component used in the implementation is the 82588 LAN controller chip. This chip is capable of executing the required LAN managing tasks dictated by the 802.3 protocol, and it is connected to the 80188 host microprocessor via straight pin-to-pin connections. The next chapter discusses the software programming of the 82588-80188 chips.

Chapter

5

The Software of the 1BASE5 Node Board

This chapter discusses how the implementation of the software for a node board in Starlan is accomplished. The hardware of Figs. 4.4 and 4.5 is used for the illustration. That is, the Intel 82588 is utilized as the LAN controller, and the Intel 80188 is utilized as the host processor. An elaborate description of operating the 82588 is included.

Although the discussion is oriented toward the implementation of the software (firmware) with the above hardware, similar software techniques are utilized when implementing the Starlan node (and other LAN boards) with different hardware components.

5.1 Transmitting a Frame

When the PC wishes to send data to the node, it transfers the data to the memory chips located on the node board. The PC then informs the 80188 (e.g., by using interrupts not shown in Fig. 4.4) that data are available for transmission.

The 80188 configures its on-chip DMA controllers by writing into the DMA registers the starting address and the number of bytes of the data to be transmitted. The 80188 then sends a command to the 82588, commanding it to start transmitting.

The 82588 responds by decoding the command, and once realizing that this is the TRANSMIT command, it checks to see if the network is busy (by examining its RxD pin). If the network is free, the 82588 issues a request to the DMA of the 80188, requesting bytes. The 82588 request data transfer by asserting its DRQ0 or DRQ1 pin. The 82588 is told in advance which of the DRQ pins should be used.

The DMA responds by transferring bytes to the 82588 in accordance with its preprogrammed starting address and number of bytes. The 82588 processes the bytes by computing the CRC, constructing a frame, inserting all the necessary fields in the frame, and it sends the frame in a Manchester code to the transmit circuit via its TxD pin.

5.1.1 Receiving frames from the network

The 80188 has to allocate space in memory (called a buffer) for the received frames. The 80188 programs one of its DMA according to the allocated RAM space and sends the RECEIVE-ENABLE command to the 82588.

When a frame is received by the 82588 from the network, the 82588 converts the data from Manchester code back to NRZ code and performs all the other tasks specified in the 802.3 Starlan protocol. The 82588 then requests data transfer from the DMA controller of the 80188, requesting the transfer of bytes from its internal registers to the memory chips. The 82588 assumes that the DMA controller is already programmed by the 80188.

When a complete frame had been received and stored in memory, the 82588 asserts its INT pin, interrupting the host processor to indicate that a new frame is available in memory.

5.2 Preparing Data for Transmission in the RAM Chips

Figure 5.1 shows how the data to be transmitted is placed in the RAM chips by the 80188. This block of data is called the transmit data block. The 82588 constructs the frame (see Fig. 1.12) based on the content of the transmit data block.

Figure 5.1 shows that the first 2 bytes of the transmit data block specify the length of the rest of the block.

The next 6 bytes contain the destination address. The 82588 constructs the destination address field of the frame by using these 6 bytes.

The next 2 bytes are the length field of the frame, specifying the length of real data in the information field. The 82588 constructs the length field of the frame by using these 2 bytes. The rest of the bytes in the block are the actual data. The 82588 constructs the information field of the frame by using these bytes.

Example 5.1 Specify the two rules regarding the actual data to be transmitted.

Solution

1. The maximum number of bytes in the actual data is 1500 bytes.
2. The minimum number of bytes in the actual data is 46. If the PC supplies less

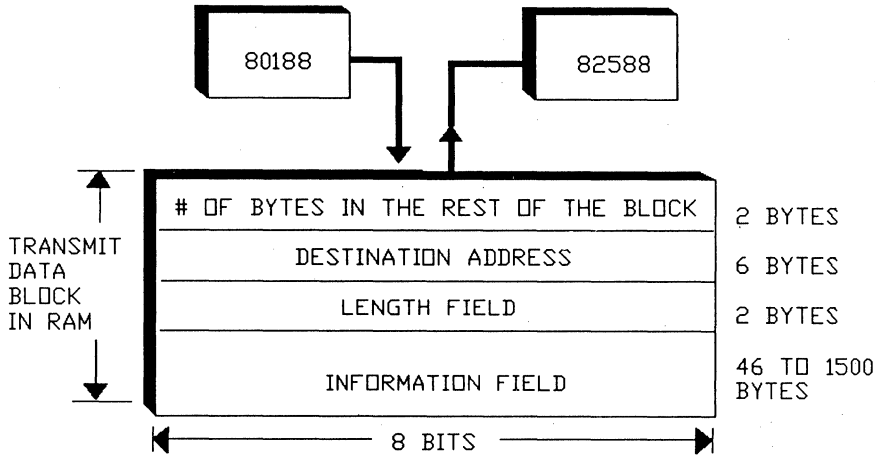


Figure 5.1 Data flow, 80188 to 82588.

than 46 bytes of actual data, the program of the 80188 should add dummy bytes to make the length of the actual data 46 bytes long (as discussed in Chap. 1).

Example 5.2 Specify the fields of the frame that the 82588 has to construct (see Fig. 1.12).

Solution

- *The preamble field.* This field is generated by the 82588.
- *The start of frame field.* This field is generated by the 82588.
- *The destination address field.* The 82588 extracts the 6 bytes of the destination address from the transmit data block (Fig. 5.1).
- *The source address field.* Part of the initialization procedure that the 80188 must perform is to supply the 82588 the source address of the node. The source address is then stored in the 82588 internal registers. Thereafter, the 82588 inserts the source address in each of the frames that it constructs.
- *The length field.* The 82588 extracts the 2 bytes of the length field from the data that it receives from transmit data block (see Fig. 5.1).
- *The information field.* The 82588 extracts the bytes of the information field from the transmit data block (Fig. 5.1).
- *The CRC field.* The 82588 calculates the CRC by itself.

5.3 Storing the Received Data in RAM Chips

5.3.1 The promiscuous mode

Upon receiving a frame, the 82588 examines the destination address field of the frame and determines if the frame is intended for itself. The 82588 may be configured to work in a promiscuous mode. In this mode, the 82588 receives all frames, regardless of the destination address.

5.3.2 Storing the received data in RAM

Once a frame is received from the network, the 82588 processes the frame, removes a few fields from the frame, adds two frame status bytes, and requests a DMA transfer for storing the data in the RAM. The data are then stored in RAM as shown in Fig. 5.2. The two frame status bytes contain information regarding the reception of the frame.

As shown in Fig. 5.2, the data that are transferred from the 82588 to the buffer RAM consist of the following:

- The 6 bytes destination address of the incoming frame
- The 6 bytes source address of the incoming frame
- The 2 bytes length field of the incoming frame
- The information field
- The 2 frame status bytes that the 82588 appends

Example 5.3 What is the destination address that is received from the network and then transferred to the RAM?

Solution The destination address received from the network could be any one of the following:

- The address of this node, that is, the 82588 recognizes its own source address in the destination address field of the incoming frame and processes the frame.
- The multicasting address, that is, the 82588 recognizes its own group address in the destination address field of the incoming frame and processes the frame.
- The broadcasting address, that is, the 82588 recognizes all 1s in the destination address field of the incoming frame and processes the frame.
- If the 82588 is in the promiscuous mode, the destination address may be any valid address.

Example 5.4 What is the source address that is received from the network and then transferred to the RAM?

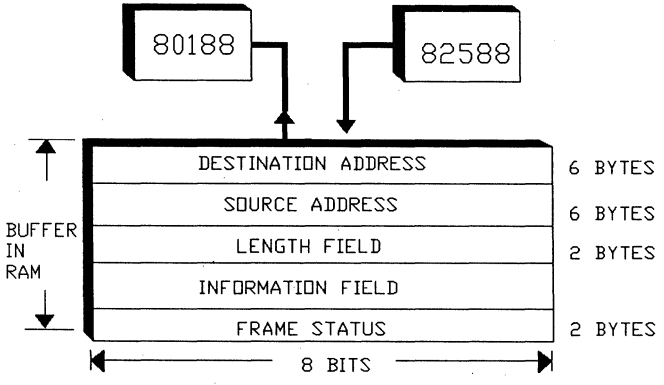


Figure 5.2 Data flow, 82588 to 80188.

Solution The source address received is the address of the far node that transmits the frame.

Example 5.5 How does the 80188 calculate the address location of the first frame status byte?

Solution The location of the frame status bytes can be calculated based on the value of the length field that is always located in the thirteenth and fourteenth bytes of the buffer.

5.4 The Two Status Bytes Attached to the Data

This section includes a detailed description of the two frame status bytes that the LAN controller reports to the host processor. Although the format (and names) of these bytes is slightly different when using other LAN controller chips, most LAN controllers provide identical or similar status information to their host processors.

The discussion is included to provide the reader with a feeling of the type of information that is available from the LAN controller for writing the host processor software program. The two status bytes that are attached to the data (Fig. 5.2) are shown in detail in Fig. 5.3.

5.4.1 The NO EOF bit of the frame status bytes

As shown in Fig. 5.3, bit 6 of the first frame status byte is called the NO EOF bit.

The 82588 can be configured to construct two types of frames:

- *A bit-stuffing frame.* In a bit-stuffing frame, the 82588 adds a special field called the *end of frame* field (EOF) at the end of the frame. If a frame is received with a missing EOF field, the 82588 writes a 1 into the NO EOF bit location.
- *An 802.3 compatible frame.* This is the type of frame discussed in Chap. 1. The EOF field is not implemented in the 802.3 frame.

Example 5.6 In our application we are configuring the 82588 to construct 802.3 frames (not bit-stuffing frames), and therefore the NO EOF bit has no relevance to our application.

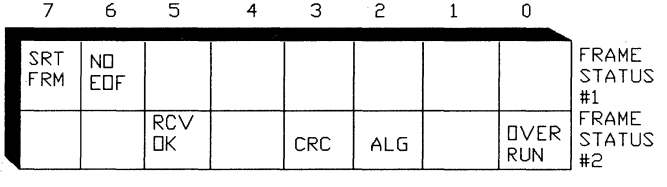


Figure 5.3 The two status bytes.

5.4.2 The short frame bit of the frame status byte

As shown in Fig. 5.3, bit 7 of the first frame status byte is called the SRT FRM bit (short frame bit).

As discussed in Chap. 1, the incoming frame must have a minimum number of bytes. If the 82588 receives a frame that is less than 6 bytes, it is obvious that something went wrong during the transmission of that frame, so the 82588 does not pass that frame to the RAM chips (i.e., it does not request a DMA transfer for the frame).

If the incoming frame is received with more than 6 bytes in it, but yet it does not contain the minimum required number of bytes in it, the 82588 passes that frame to the RAM chips. The 82588 writes a 1 in bit 7 of the first frame status byte, an indication to the 80188 that the frame is too short. If the frame complies with the minimum number of bytes requirements, the 82588 writes a 0 into this bit location.

5.4.3 The OVERRUN bit of the frame status bytes

As shown in Fig. 5.3, bit 0 of the second frame status byte is called the OVERRUN bit. Once bits are coming from the network into the 82588, the 82588 constructs bytes and asserts its DRQ0 or DRQ1 pin to request DMA byte transfer of these bytes to the RAM.

If it happens that the DMA of the 80188 is not ready and does not keep up with the requests, the 82588 knows about that, since it expects to receive a data transfer acknowledgment on its DACK0 or DACK1 pin. This situation is called an *overrun* error, and the 82588 reports this error in the OVERRUN bit of the frame status byte. Evidently, some bytes got lost due to this overrun error.

Example 5.7 How can the overrun error be avoided?

Solution The overrun error can be avoided by designing an efficient program for the 80188, making the DMA ready at all times (since incoming frames could be arriving at any time).

Upon receiving an interrupt from the 82588 and discovering that a frame was received and stored in the RAM, the 80188 should immediately program its DMA for the next buffer, since it is possible that a new frame is already on its way to this node (see discussion of multiple access in Chap. 1).

5.4.4 The ALIGNMENT ERROR bit of the frame status bytes

As shown in Fig. 5.3, bit 2 of the second frame status byte is called the *ALG bit* (the alignment error bit). If during the reception of the frame, the 82588 encounters alignment errors in the incoming frame, it writes a 1 into this bit location.

In an 802.3 frame, the number of bits following the start of frame field of the frame must be in multiples of eight. If the 82588 discovers that this is not the case, it writes a 1 into the ALIGNMENT ERROR bit location of the frame status byte.

5.4.5 The CRC bit of the frame status bytes

As shown in Fig. 5.3, bit 3 of the second frame status byte is called the *CRC error bit.* If during the reception of the frame, the 82588 calculates the CRC of the incoming frame and finds that it is different from the CRC contained in the CRC field of the incoming frame, it writes a 1 into the CRC ERROR bit of the frame status byte.

5.4.6 The RCV OK bit of the frame status bytes

As shown in Fig. 5.3, bit 5 of the second frame status byte is called the *RCV OK bit.* The 82588 writes a 1 into this bit location if it encounters no errors during the reception of the frame.

Example 5.8 The RCV OK bit is 1 only if all the rest of the error bits in the frame status bytes are 0s.

5.4.7 Reclaiming the buffer

Upon discovering that a frame was received with error(s), the 80188 might reclaim the RAM buffer area. That is, the program of the 80188 may set the DMA controller to transfer the next incoming frame to the very same area of RAM (i.e., no sense wasting RAM on bad frames). However, the program should include a routine that records the number of received bad frames. The higher layer software level (executed by the PC) might make use of such statistical information and alert the user if there were too many received bad frames.

The other bits shown in Fig. 9.3 are not utilized.

5.5 The Amount of RAM Buffer Required for Storing Incoming Frames

In most cases, the 80188 does not know in advance the size of the incoming frames. Therefore, it has to allocate a RAM buffer large enough to be able to store the largest possible frame.

Example 5.9 What is the size of the buffer that the 80188 should allocate for a frame?

Solution The RAM buffer should contain enough space to be able to accommodate the largest possible frame. The largest possible frame is (Fig. 5.2):

6 bytes for the destination address

6 bytes for the source address.

2 bytes for the length field.

1500 bytes for the information field.

2 bytes for the frame status bytes.

Total: 1516 bytes

Example 5.10 Suppose that the 80188 of Fig. 4.4 has access to 64K bytes of memory, divided in the following fashion:

- 16K bytes are dedicated for the ROM chip, storing the 80188 program.
- 16K bytes of RAM are dedicated to store data for transmission, and for various operations of the 80188 program.
- 16K bytes are dedicated for storage area (buffer) for the received frames.

How many frames can be received before the 80188 must transfer the data to the PC?

Solution The previous example illustrates that 1516 bytes are required for the storage of a single received frame:

$$\frac{1024 \times 16}{1516} = 10.8$$

Only 10 frames!

During normal operation of the LAN, the lengths of the frames are less than the maximum allowed. Obviously a more efficient method of utilizing the RAM is needed.

5.5.1 Two methods of storing received data into the RAM

There are two methods of storing the received data into the RAM chips:

1. The single buffer reception method
2. The multiple buffer reception method

The single buffer method is used in the previous example.

5.5.2 The multiple buffer reception method

The multiple buffer reception method is by far a more efficient method of RAM utilization. To use this method, the 80188 informs the 82588 in advance that the multiple reception method is being used.

In this method, the 80188 prepares a buffer table in the RAM as shown in Fig. 5.4. Each entry of the buffer table contains the address of a buffer in the RAM. In the example of Fig. 5.4, the buffer table has four entries,

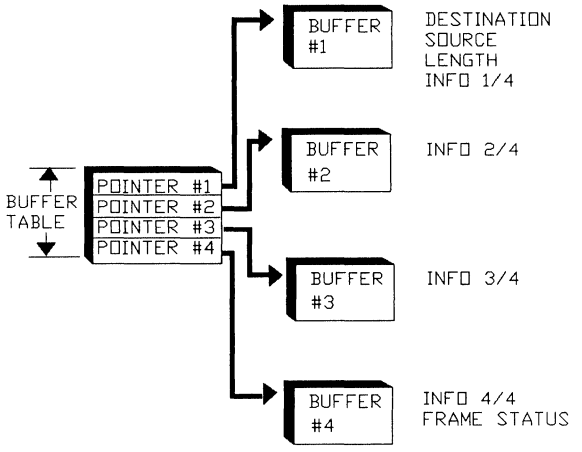


Figure 5.4 Using buffer table.

each pointing to a buffer area in the RAM. Each buffer must have the same size. The minimum size of each of the buffers is 4 bytes, and the maximum size is 1024 bytes. The size of the buffer must be in multiples of 4 (i.e., the size of each buffer must be 4, 8, 12, . . . , or 1024 bytes).

The 82588 must know the exact size of the buffers in advance. Initially, the 80188 sets its DMA #0 to transfer bytes from the 82588 to buffer #1. Upon receiving a frame, the 82588 requests data transfer to buffer #1 by asserting its DRQ0 pin. A portion of the received frame is then transferred from the 82588 to buffer #1. Immediately after the first byte is transferred to buffer #1, the 82588 interrupts the 80188. This interrupt is an indication to the 80188 that buffer #1 is currently being filled. The 80188 responds to the interrupt by preparing its DMA #1 to transfer bytes to buffer #2.

When the 82588 completes filling buffer #1, it automatically requests the service of DMA #1 by asserting its DRQ1 pin. Bytes are then transferred from the 82588 to buffer #2 via DMA #1 (which is already programmed). Upon filling the first byte in buffer #2, the 82588 interrupts the 80188. This interrupt is an indication to the 80188 that buffer #2 is currently being filled. The 80188 responds to the interrupt by preparing its DMA #0 to transfer bytes to buffer #3.

Eventually, buffer #2 is full, and the 82588 requests data transfer by asserting its DRQ0 pin. Since the DMA #0 is already programmed for buffer #3, bytes are then transferred from the 82588 to buffer #3.

The process continues until the whole received frame is stored into the RAM. By using this method, the RAM is better utilized, since each of the buffers (except possibly for the last buffer in the chain) is filled with data. Of course the program of the 80188 is responsible for managing the buffer tables.

5.6 The Command Register of the 82588

This section is included to provide the reader with a feeling of what it takes to program the 82588 LAN controller.

5.6.1 Programing the 82588

The 82588 is a sophisticated processor responsible for a variety of complicated computations and data processing tasks specified in the 802.3 protocol. The program that is responsible for accomplishing all these tasks is already embedded within the chip.

5.6.2 The program of the 82588

The program of the 82588 can be viewed as having 16 major routines. To cause the 82588 to execute any of its routines, the 80188 sends a command to the 82588, telling it which routine should be executed. Thus, there are 16 commands, one for each routine. An example of one of the commands is the TRANSMIT command, which instructs the 82588 to start transmitting data to the network.

5.6.3 The interrupt pin

Upon the occurrence of certain events, the 82588 asserts its output INT pin. Since the INT pin of the 82588 is directly connected to the input INT pin of the 80188 (see Fig. 4.4), the 82588 has a means of attracting the attention of the 80188 when these certain events occurs.

5.6.4 Configuration parameters

By loading the internal registers of the 82588 with a set of configuration parameters, the 80188 can configure the 82588 to work in a variety of ways. For example, when transmitting, the 82588 automatically inserts the preamble bits at the beginning of the frame. Although the 802.3 protocol specifies 56 bits for the preamble bits, the 82588 can be configured to insert a different number of preamble bits. One of the configuration parameters is the one that specifies the number of the preamble bit that the 82588 should insert at the beginning of the frame.

5.6.5 Reporting the results of command execution

The 82588 is equipped with on-chip status registers. Once the 82588 receives a command from the 80188, it decodes the command and executes it. During and after the execution of the command, the 82588 reports to its internal status registers the status results of the execution. For example, at the end of the execution of the TRANSMIT com-

mand, the status registers are updated with information such as whether or not the transmission of the frame was successful. If not successfully implemented, the 82588 writes the cause of the failure into the status registers.

The 80188 reads the content of the 82588 status registers, and based on the content, takes the appropriate actions.

5.7 The Programming Model of the 82588

The 82588 programming model consists of one 8-bit command register and four 8-bit status registers (Fig. 5.5). The four status registers are called status register #0, status register #1, status register #2, and status register #3.

5.7.1 The command register

When the 80188 wishes the 82588 to execute a command, it writes the desired command into the command register of the 82588. The 80188 writes the command into the 82588 command register as if it were a regular address location. That is, the byte presented to the D7 to D0 pins of the 82588 is written into the command register provided that the 80188 asserts the chip select (CS) and write (W) pins of the 82588 and negates the read (R) pin of the 82588. Based on the hardware (see Fig. 4.4), to issue a command to the 82588, the program of the 80188 simply writes a byte into the address assigned to the PCS0 pin.

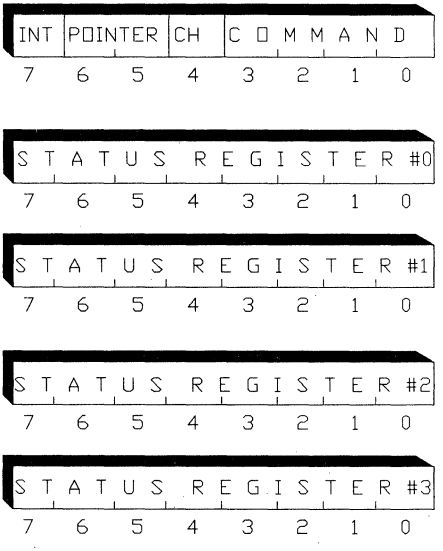


Figure 5.5 The command and status registers of the 82588.

The command register is a write-only register. That is, the value contained in this register cannot be read by the 80188. As we shall soon see, upon reading the command register, the 80188 receives the content of a different register.

5.7.2 The format of the command

The command byte consists of four fields (Fig. 5.5):

- The command field
- The channel field (CH field)
- The pointer field
- The interrupt acknowledge field

5.7.3 The command bits 3, 2, 1, 0 of the command register

Bits 3, 2, 1, and 0 represent the command itself. The 4 bits can represent 16 possible combinations, each corresponding to a different command. The 16 possible combinations are shown in Table 5.1.

Example 5.11 Suppose that the 82588 of Fig. 4.4 is decoded to address 01234. What is the procedure that the 80188 should follow to issue the NOP command to the 82588?

Solution Assume that the PCS0 of the 80188 is already programmed to correspond to address location 01234. The 80188 prepares the command byte and then writes the byte corresponding to the NOP command into address 01234.

The command byte is

TABLE 5.1

Bits 3, 2, 1, 0	Command
0 0 0 0	NOP
0 0 0 1	IA-SETUP
0 0 1 0	CONFIGURE
0 0 1 1	MC-SETUP
0 1 0 0	TRANSMIT
0 1 0 1	TDR
0 1 1 0	DUMP
0 1 1 1	DIAGNOSE
1 0 0 0	RCV-ENABLE
1 0 0 1	ASSIGN ALTERNATE BUFFER
1 0 1 0	RCV DISABLE
1 0 1 1	STOP RECEPTION
1 1 0 0	RE-TRANSMIT
1 1 0 1	ABORT
1 1 1 0	RESET
1 1 1 1	RELEASE POINTER or FIX POINTER

```
Bit #: 7 6 5 4 3 2 1 0
       N N N N 0 0 0 0
```

Bits 3, 2, 1, 0 are 0, 0, 0, 0, since as indicated in Table 5.1, the 4 bits corresponding to the NOP command are 0, 0, 0, 0. The values of the N, N, N, N bits will be discussed shortly. The NOP command is the no operation command, instructing the 82588 to continue doing whatever it is doing.

5.7.4 The pointer field, bits 6 and 5 of the command register

When the 80188 wishes to read the value contained in any one of the four status registers of the 82588, it simply reads a byte from the very same address location of the command register. Since the 82588 has four status registers, the 82588 must be told in advance which of the four status registers is the desired one.

As shown in Fig. 5.5, bits 6 and 5 of the command register are called the *pointer*. The pointer is an indication to the 82588, telling it which status register is being read.

Example 5.12 Suppose that the 82588 of Fig. 4.4 is decoded to address 01234. How does the 80188 read the value of status register #2?

Solution The 80188 reads the status register in two steps. In the first step, the 80188 writes a command byte into address 01234. The command byte is

```
Bit #: 7 6 5 4 3 2 1 0
       N 1 0 N 0 0 0 0
```

Bits 3, 2, 1, 0 are 0, 0, 0, 0, representing the NOP command. Bits 6, 5 are 1, 0, representing decimal value 2. (Bits 7,4 were not discussed yet.)

In the second step, the 80188 simply reads the value of address 01234. The 82588 outputs the value of status register #2, since 2 is the value currently contained in the pointer of the command register. (This 2 was set during step 1.)

The value of the pointer is then incremented automatically. Thus, after the reading, the value of the pointer in the command register becomes automatically 11 (decimal 3).

Now that the pointer contains the value 3, another reading from address 01234 is the same as reading status register #3, and the pointer goes back to 00, corresponding to status register #0. Of course, the 80188 may skip the first step if the command register happened to contain the desired pointer value.

The above example utilizes the NOP command for the sole purpose of setting the pointer in the command register. However, the pointer can be set with any of the 16 commands of Table 5.1.

5.8 The FIX POINTER and RELEASE POINTER Commands

As shown in Table 5.1, the RELEASE POINTER and the FIX POINTER commands correspond to the same 4 bits (bits 4, 3, 2, 1 = 1111). These

commands are provided for cases where the user does not want the pointer to increase automatically after each status register reading.

Since both the FIX POINTER and the RELEASE POINTER commands have the same values for bits 3, 2, 1, 0, bit 4 of the command byte determines which command is used. If bit 4 is equal to 1, it is the FIX POINTER command. If bit 4 is equal to 0, it is the RELEASE POINTER command.

The FIX POINTER command fixes the pointer at a fixed value. Once the 82588 receives this command, it causes the pointer not to advance after each status register read. The pointer is fixed at the value indicated by the pointer of the command byte.

The RELEASE POINTER command cancels the FIX POINTER command. Once the 82588 receives this command, it causes the pointer to advance after each status register read.

Example 5.13 Suppose the user wishes to keep reading status register #1 time after time.

Solution Assume that the 82588 is decoded to address 01234. The 80188 writes the following command byte into address 01234:

```
Bit #: 7 6 5 4 3 2 1 0
       N 0 1 1 1 1 1 1
```

Bits 6, 5 are 0, 1, representing the decimal value 1 for the pointer.

Bits 3, 2, 1, 0 are 1, 1, 1, 1 and bit 4 is 1, representing the command FIX POINTER command.

Now, any time the 80188 reads the address location 01234, it reads the value of status register #1, since the pointer does not increment after each reading.

Example 5.14 To cancel the FIX POINTER command issued in the previous example, the 80188 writes the following command byte into address 01234:

```
Bit #: 7 6 5 4 3 2 1 0
       N 1 1 0 1 1 1 1
```

Bits 3, 2, 1, 0 are 1, 1, 1, 1 and bit 4 is 0, representing the RELEASE POINTER command.

Bits 6, 5 are 1, 1, representing setting the pointer to status register #3.

From now on, any reading of the status register causes the pointer to increment by one after the reading.

Note that the choice for bits 6, 5 = 1, 1 is arbitrary; we could have set the pointer to any other value.

At this point the pointer is released. If the 80188 issues the RELEASE POINTER command again, nothing fatal occurs; the 82588 simply treats the second RELEASE POINTER command as a NOP command.

Generally speaking, whenever the 82588 receives a command that cannot be executed, it treats the command as a NOP command.

5.8.1 Resetting the 82588

The 82588 can be reset in one of two ways:

- *Hardware reset.* The 82588 is reset whenever the 80188 asserts the RESET pin of the 82588 (Fig. 4.4). As discussed in Chap. 4, due to that connection, the 82588 is reset whenever the 80188 is reset.
- *Software reset.* While the hardware reset is a perfectly legal way of resetting the 82588, there are occasions where the user is required to reset the 82588 via software. In these cases, the 80188 may issue a RESET command.

Example 5.15 What should be the command byte corresponding to the RESET command?

Solution The command byte is

```
Bit #: 7 6 5 4 3 2 1 0
       X X X X 1 1 1 0
```

Bits 3, 2, 1, 0 = 1, 1, 1, 0 correspond to the RESET command (Table 5.1), and the rest of the bits of the command are "don't cares" (since the 82588 is immediately reset).

Hint: As is the case with all chips, it takes a certain amount of time to get used to the 82588. During the software development of the node board, the developer may encounter situations whereby the 82588 locks into a certain state and refuses to get out of that state. For example, neglecting to clear certain flags, may cause this to occur. A RESET is a sure thing to get the 82588 out of any locking states.

5.8.2 The interrupt acknowledge bit of the command register

Bit 7 of the command register is called the *interrupt acknowledge* bit (Fig. 5.5). Once the 82588 asserts its INT pin, the pin stays asserted until the 80188 writes a command byte where its bit 7 equals 1.

The only other way to clear the INT is to reset the 82588 (hardware reset or software).

5.9 Commands That Utilize the Service of the DMA Controller of the 80188

When executed, certain commands require transfer of bytes from the RAM chips to the 82588, or transfer of bytes from the 82588 to the RAM chips. As previously discussed, such data transfer is performed by the DMA controllers of the 80188.

Example 5.16 Suppose that the 80188 issues the DUMP command to the 82588. List the proper procedure of issuing such a command.

Solution As shown in Table 5.1, the 4 bits that correspond to the DUMP command are 0, 1, 1, 0. The 82588 has 63 registers on-chip. During the development time of the software for the node board, the developer might want to examine these registers. The DUMP command causes the 82588 to deposit all 63 registers into the RAM chips.

In the discussion below, we assume that DMA #0 is utilized. The proper procedure for issuing the DUMP command is

- The 80188 programs its on-chip DMA #0, specifying the source address (the address of the 82588), the number of bytes to be transferred (63 bytes), and the address location in RAM where the registers are to be deposited.
- The 80188 programs its PCS1 pin to correspond to the same address range specified for the DMA #0.
- The 80188 writes the command byte corresponding to the DUMP command into the command register of the 82588.

Upon receiving the command, the 82588 responds by asserting its DRQ0 pin and outputs the value of the first register onto its data pins, D0 to D7.

The DMA #0 of the 80188 responds to the assertion of its DRQ0 pin by reading the byte from the 82588, storing the bytes to the destination address, and updating its own pointer (for the next byte transfer). Since PCS1 is programmed for the same address range of DMA #0, the PSC1 is asserted when the DMA #0 assesses the address of the destination address. As shown in Fig. 4.4, the PCS1 is directly connected to the DACK0 pin of the 82588.

Upon receiving a 0 on its DACK0 pin (from the PCS1 pin), the 82588 realizes that the bytes were stored in the destination. The 82588 therefore outputs the value of the second register to its D0 to D7 pins and asserts its DRQ0 pin again. The process repeats itself until all 63 bytes are transferred.

Hint: The DUMP feature is included as a developing tool for debugging the developed software, enabling the user to examine the 63 registers in case of difficulties. While the developer is encouraged to examine this feature, operating the 82588 is a straightforward process, and the developer would probably find no need to use this feature.

5.10 The Channel Bit of the Command Byte

As indicated in Fig. 5.5, bit 4 of the command byte is the *channel bit.* In the previous example, we assumed that the 80188 utilizes its DMA #0, and the 82588 uses its DRQ0 and DACK0 pins. Obviously, the 80188 must have the means to notify the 82588 which DMA channel should be used. The channel bit of the command byte is the bit that indicates which channel should be used. Channel bit = "0" is an indication to use channel 0, and Channel bit = "1" is an indication to use channel 1.

Example 5.17 What is the value of the command byte in the previous example?

Solution The command byte is

```
Bit #: 7 6 5 4 3 2 1 0
       0 0 0 0 0 1 1 0
```

- Bit 7 = "1" means that the 80188 does not require clearing of the INT signal. (We assume here that there is no need to clear the INT pin.)
- Bits 6,5 = "0,0" mean that the pointer is set to 0, pointing to status register #0 (an arbitrary choice for this example).
- Bit 4 = "0" means that the channel to be used when executing the command is channel 0. (The 82588 is told here to assert its DRQ0 pin and to accept acknowledgment on its DACK0 pin.)
- Bits 3,2,1,0 = "0,1,1,0" is the DUMP command.

Example 5.18 Bit 4 of the command byte is also used when issuing the FIX POINTER and RELEASE POINTER commands. However, for these commands, bit 4 of the command is not used as a channel bit.

5.11 Updating the Status Registers

The 82588 updates its internal status registers with various bits of information regarding the status of the network and the status of the execution of the commands.

5.11.1 The status registers

The four status registers shown in Fig. 5.5 may be divided into two groups:

- Status registers #0, #1, and #2: These status registers contain information regarding both the execution of the last command and the last frame received.
- Status register #3: This status register contains information regarding the state of the 82588.

From time to time, a certain event occurs that causes the 82588 to assert its INT pin. The 82588 keeps asserting the INT pin until the 80188 explicitly commands the 82588 to negate the INT pin.

Example 5.19 What are the proper actions that the 80188 should take upon receiving an interrupt signal from the 82588 (i.e., when the 82588 asserts its INT pin)?

Solution The 80188 should take the following actions:

- The 80188 should read the status registers to find out what event took place that caused the 82588 to assert its INT pin.
- The 80188 should instruct the 82588 to negate the INT pin (by writing a 1 into bit 7 of the command register).

5.11.2 What happens if the 80188 does not clear the INT of the 82588?

As mentioned, it is the responsibility of the 80188 to clear the INT signal. Once the 82588 asserts its INT pin, it refuses to update status

register #0, status register #1, and status register #2 unless its INT signal is cleared.

The 82588 keeps updating status register #3 even if its INT pin is not cleared. Thus, status register #3 is treated differently by the 82588 than the other status registers.

5.12 The Program of the 80188

The program for the 80188 consists of configuring the 82588, issuing the commands, and then reading the status registers to analyze the results of the executions. Of course, prior to issuing a command that requires the service of a DMA controller, the 80188 must program the corresponding DMA controller and set the appropriate channel of the 82588 channel bit (bit 4 of the command register).

5.13 Summary

This chapter discussed the software implementation that is required for the implementation of a Starlan node board when implemented with the 82588-80188 chips. Programming the 82588 LAN controller consists of issuing commands to it. The on-chip software of the 82588 interprets the commands and executes them. Data are transferred between the 82588 and the local RAM chips by the use of the on-chip DMA controllers of the 80188 host processor. In the next chapter we will design a node board (an Ethernet/Cheapernet node board) that utilizes another LAN controller chip, the 82586 chip. This LAN controller does not use the DMA controller of the host processor, but rather uses its own DMA controllers which are integrated into the chip.

Chapter

6

10BASE5 (Ethernet) and 10BASE2 (Cheapernet)

6.1 The Topology and Components of Ethernet

Figure 6.1 shows how three PCs are connected to an Ethernet network. The Ethernet network that is shown and discussed complies with the IEEE standard protocol 802.3 10BASE5. Each PC has a node board that is usually designed to be plugged into one of the I/O slots of the PC.

Each of the node boards is connected to a cable called a *transceiver cable.* The transceiver cable is connected on its other end to a board called a *transceiver board.* The transceiver board is connected to a coaxial cable called the *Ethernet segment.* The Ethernet segment is terminated with two resistors called *terminators,* one at each side of the Ethernet segment.

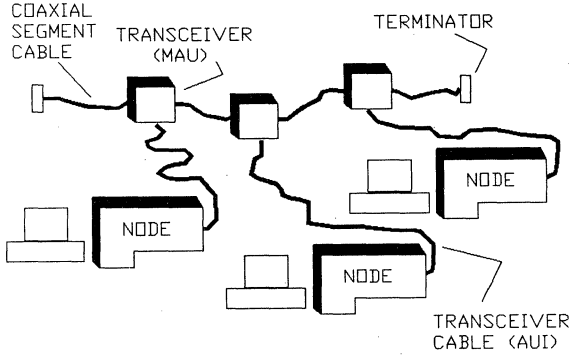


Figure 6.1 The Ethernet topology.

Example 6.1 The transceiver board is also called a *medium attachment unit,* or MAU for short. The transceiver cable is also called *attachment unit interface* cable, or AUI cable for short.

Example 6.2 The function of the terminator resistor is to eliminate reflection. *Reflection* is an electrical phenomenon where some of the signal energy is reflected back (as noise) to the source that generates the signal. By terminating the wires with the proper resistors, reflection is minimized.

6.1.1 The data rate in Ethernet

The IEEE 802.3 10BASE5 Ethernet protocol specifies a data rate of 10 Mbit/s, which is the reason for the "10" in the protocol name "10BASE5".

6.1.2 Node placement

Although Fig. 6.1 shows only three nodes connected to the segment, an Ethernet segment may have up to 100 node boards connected to it. The transceiver boards must be spaced at a minimum interval of 2.5 m from each other (Fig. 6.2).

6.1.3 The Ethernet segment

The maximum length of the Ethernet segment is 500 m, and it is made of 0.4-in-diameter double-shielded coaxial cable. The "5" in the protocol name "10BASE5" indicates that the maximum allowed length of the segment is 500 m.

6.1.4 The transceiver cable

The Ethernet protocol specifies a maximum length of 50 m for the transceiver cables (AUI cables). The transceiver cable is made of four

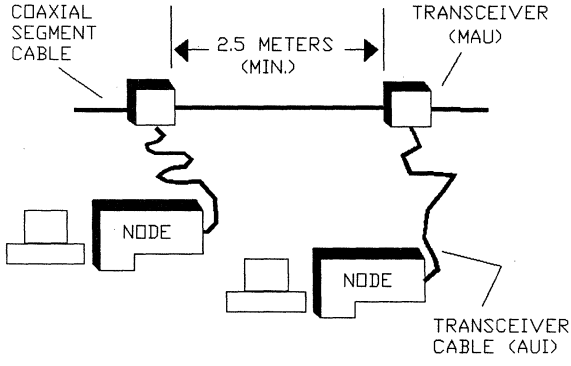


Figure 6.2 Minimum spacing between two transceivers.

groups of wires. Each group is made of twisted pairs of wires, and each group is individually shielded.

Example 6.3 Can we use the existing unused telephone wires as the transceiver cables?

Solution No, because the existing telephone wires are unshielded.

6.1.5 Connecting the transceiver board to the coaxial cable

The transceiver board (MAU board) is connected to the coaxial Ethernet segment with a special connector. Common connectors are the *clamp-on tap connector* and the *type N plug connector.*

6.1.6 Adding segments to the network

Additional segments can be added to the network by using *repeater boards* (repeaters are the subject of Chap. 7).

Example 6.4 Figure 6.3 illustrates how two additional Ethernet segments are added.

6.1.7 Cable cost

The Ethernet cables are quite expensive. The Ethernet coaxial cable costs about \$0.80/ft to \$0.90/ft, and the transceiver cable costs about \$1.45 to \$1.55/ft. Other cable-related costs include the cost of the connectors that connect the transceiver boards to the coaxial cable.

6.2 The Topology and Components of Cheapernet

In order to reduce cost, another protocol was defined, the IEEE 802.3 10BASE2 Cheapernet. The "2" in the protocol name "10BASE2" indicates that the maximum length of the Cheapernet segment is 200 m (when using the type of cable specified in the protocol), and the "10" in "10BASE2" indicates that the data rate in the network is 10 Mbit/s.

6.2.1 The Cheapernet segment cable

A Cheapernet network is shown in Fig. 6.4. In Cheapernet, the cost of the cables is reduced since the Cheapernet segment cable is made of a less expensive cable than the Ethernet segment.

The Cheapernet segment is made of 0.2-in-diameter, 50-Ω coaxial cable, which can be either double- or single-shielded. The cost of the segment cable is about 5 to 6 times less expensive than the Ethernet

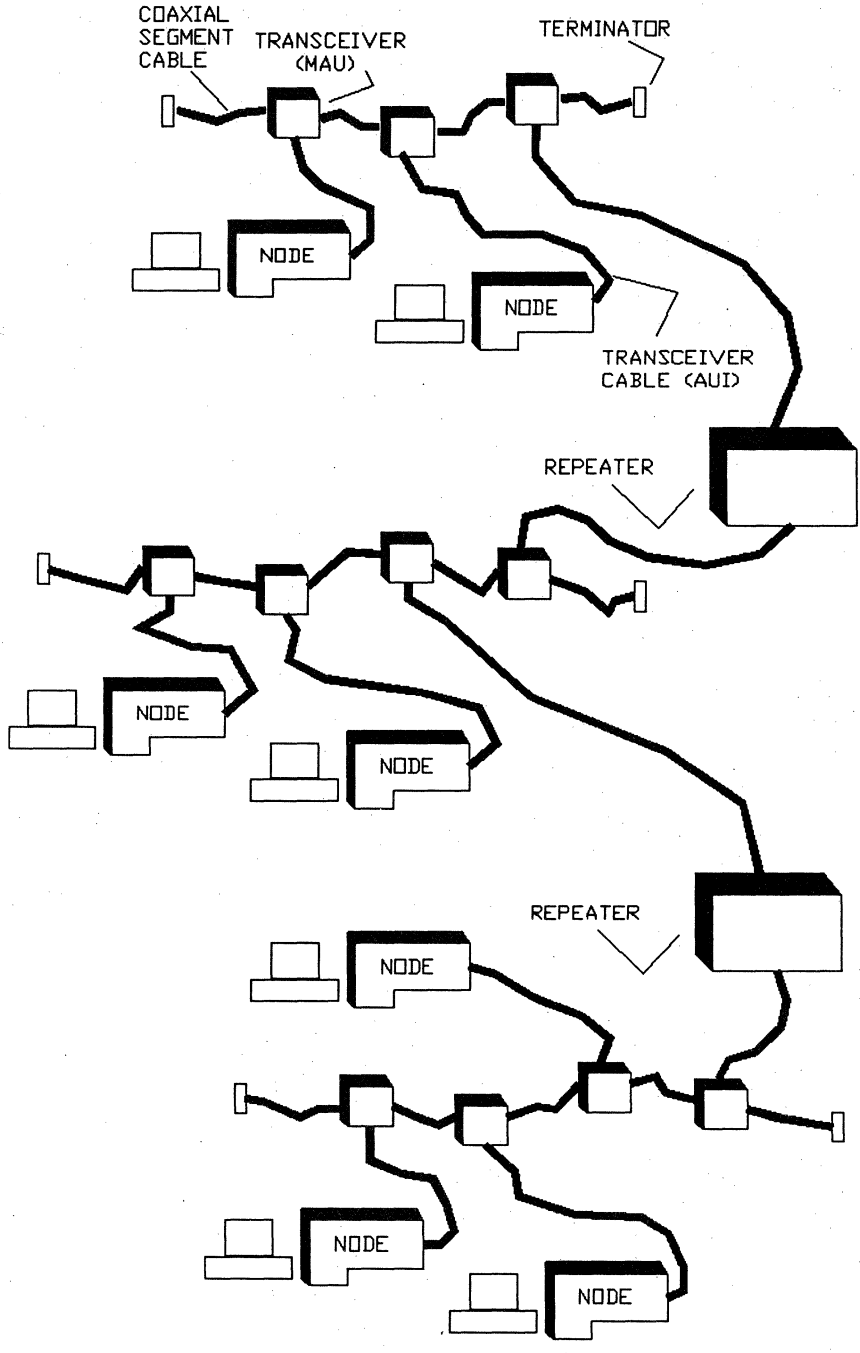


Figure 6.3 Adding additional Ethernet segments.

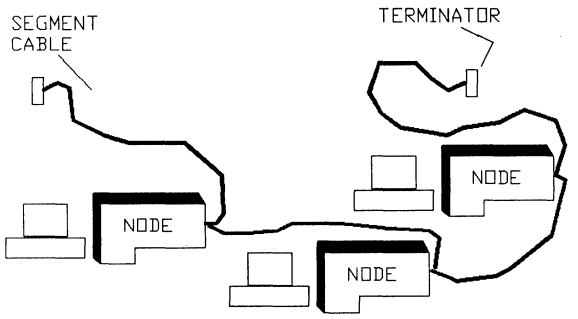


Figure 6.4 The Cheapernet topology.

segment cable. Examples of commonly used Cheapernet coaxial cables are the cables known as RG 58 A/U and RG 58 C/U cables.

Since the Cheapernet segment is thinner than the Ethernet segment, it is more flexible and can be routed easily from one place to another.

6.2.2 No transceiver cables in Cheapernet

In addition to the substantial saving in the segment-cable cost, Cheapernet topology eliminates the need for the transceiver cables. The transceiver cables are eliminated, since in Cheapernet, the transceiver circuitry is located on the node board.

6.2.3 Connecting the nodes to the Cheapernet segment

The type of connector used to connect the Cheapernet node board to the Cheapernet segment is a BNC female T connector, which is an inexpensive connector.

6.2.4 Adding additional segments

Just as in Ethernet, additional segments can be added by using repeater boards.

6.2.5 Node placement

There can be a maximum of 30 nodes per segment in Cheapernet (Ethernet can have 100 nodes per segment). The node boards must be spaced at a minimum interval of 0.5 m from each other (in Ethernet the nodes must be spaced at a minimum interval of 2.5 m from each other).

6.2.6 Combining Ethernet and Cheapernet

It is permitted to combined Ethernet segments and Cheapernet by using repeater boards.

Example 6.5 Figure 6.5 illustrates how a Cheapernet segment is added to an Ethernet segment by using a repeater. The top two segments are Ethernet segments (they have transceiver cables), and the bottom segment is a Cheapernet segment.

6.2.7 Why the name Cheapernet?

It is important to understand that there is nothing "cheap" about Cheapernet; the network is just as reliable as the Ethernet network. The only difference between Ethernet and Cheapernet is that in Cheapernet the transceiver board becomes an integral part of the node board, and it uses less expensive segment cables.

The Cheapernet is called Cheapernet to indicate the savings in cable cost.

6.2.8 Other names and terminology commonly used

The Ethernet is also called *Yellow Cable* LAN and *Thick Wire* LAN. The Cheapernet is also called *Thin Wire Ethernet* LAN.

6.3 The Transceiver Circuit in Ethernet

A simplified block diagram of the Ethernet transceiver board is shown in Fig. 6.6. The left side of the transceiver board is connected to the node board via four pairs of twisted wires. (Each of these twisted pairs should be individually shielded.) These four pairs compose the AUI cable (the transceiver cable).

The right side of the transceiver board (Fig. 6.6) is connected to the Ethernet segment (coaxial cable). The coaxial cable can be modeled as a pair of two conductors: the outer portion of the coaxial cable is one conductor, and the center wire of the coaxial cable is the other conductor.

6.3.1 The various sections of the transceiver

The transceiver board (Fig. 6.6) consists of the following main sections:

- The transmitter section
- The receiver section
- The collision detector section
- The power section

The board is called *transceiver* since it consists of a *trans*mitter and a re*ceiver*.

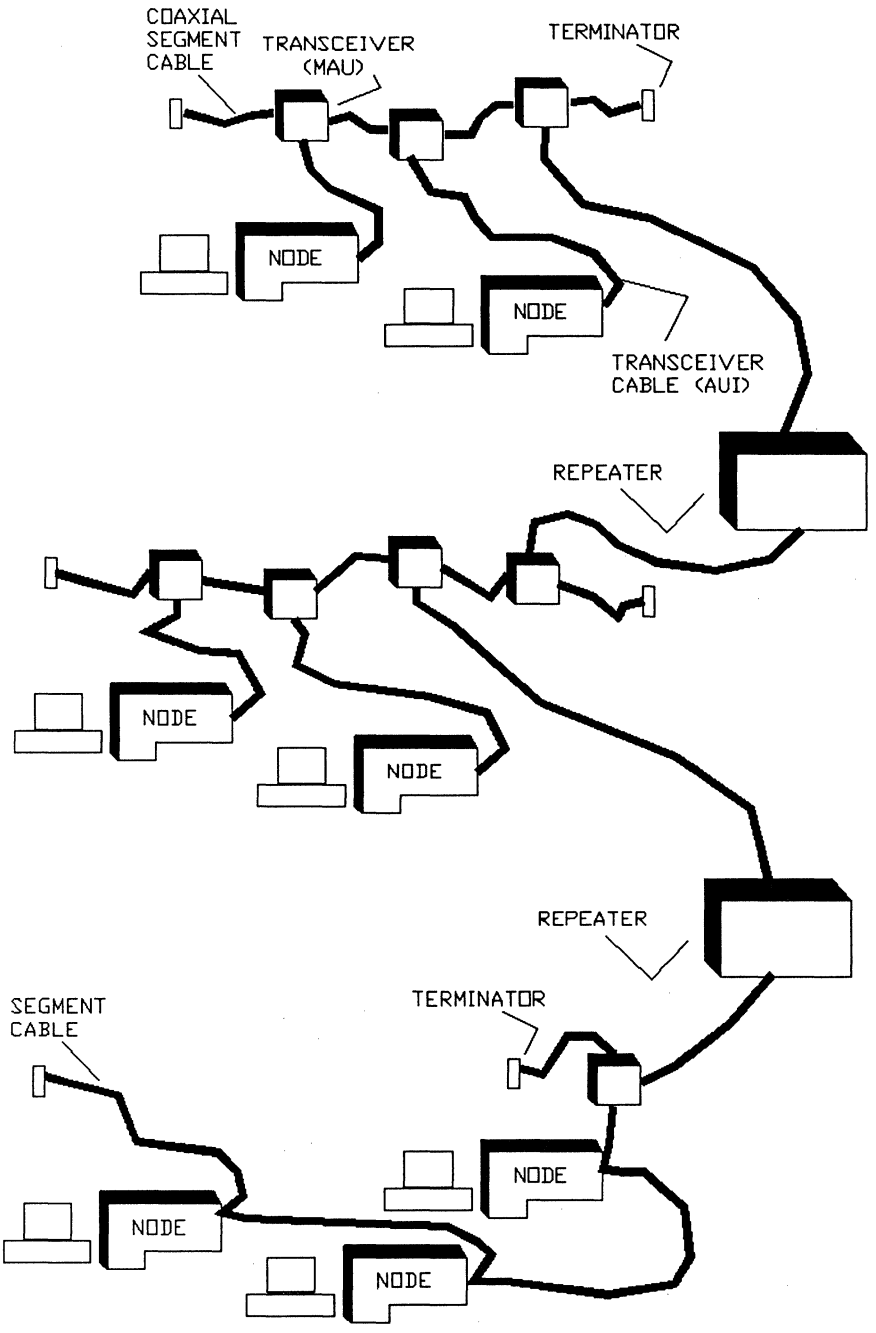


Figure 6.5 Combining Ethernet and Cheapernet segments.

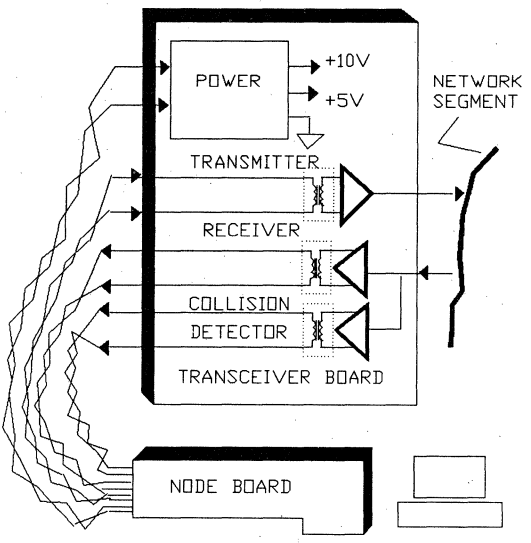


Figure 6.6 The transceiver.

6.3.2 Differential signals

Since in Ethernet the transceiver board may be located as far as 50 m away from the node board, signals are transferred between the transceiver board and the node board in differential formats. This ensures that the signals are not severely distorted during their 50-m node-transceiver trip.

6.3.3 The transmitter section of the transceiver

Upon receiving the differential Manchester signal via the AUI cable, the transmitter section of the transceiver board converts the differential signal to an NRZ signal. This NRZ signal is transmitted into the coaxial cable as shown in Fig. 6.7. That is, the 1s and the 0s at the output of the transmitter cause the current source to sink different current levels from the center conductor of the coaxial cable.

6.3.4 The receiver section of the transceiver

The signals present on the coaxial segment are fed to the input of the receiver section of the transceiver board, converted to differential signals, and then sent over the AUI cable to the node board (Fig. 6.6). The receiver interprets –200 mV on the coaxial cable as a 1, and –1.8 V as a 0. A zero voltage is interpreted as no transmission on the cable. These voltages are generated over the coaxial cable due to the current

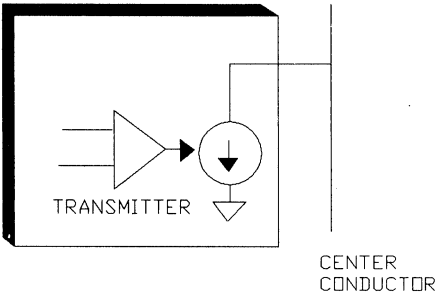


Figure 6.7 Sourcing current from the coaxial cable.

sinking (or absence of current sinking) by the transmitters of the transceiver boards that are connected to the coaxial cable.

6.3.5 The collision detector section of the transceiver

The signals from the coaxial segment are also fed to the collision detector (Fig. 6.6). The collision detector examines the incoming data and decides whether or not the received signal is a valid Manchester signal. If the incoming signal contains a Manchester code violation, the collision detector transmits to the node a differential signal that is interpreted by the node as the collision signal.

6.3.6 DC isolation between the transceiver and the node

The 802.3 protocol specifies the requirement of dc isolation between the node board and the rest of the network, which is the reason for including the transformers shown in Fig. 6.6.

6.4 Supplying Power to the Transceiver Board

In Ethernet, the node is responsible for providing the power to the transceiver board. Two voltage levels are required: the "regular" +5 V for the chips on the transceiver board and a +10 V dc for the transmitter section (Fig. 6.6).

Example 6.6 How can the node supply power to the transceiver without violating the dc isolation requirement?

Solution In order not to violate the dc isolation requirement, the ground of the node board must be isolated from the ground of the transceiver board. An isolated dc-to-dc converter is therefore needed (Fig. 6.8).

The dc-to-dc converter is connected to the node board via two wires. One input of the dc-to-dc converter is connected to the ground of the node board, and the other

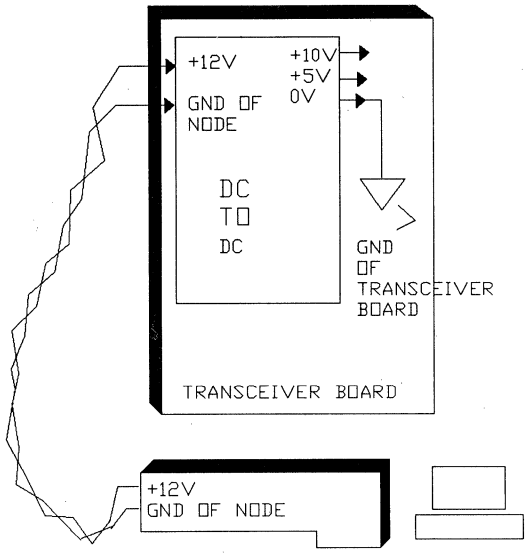


Figure 6.8 Supplying power to the transceiver board.

input is connected to a +12-V-dc supply on the node board. These two wires consist of the fourth pair of twisted wires of the AUI cable.

The outputs of the dc-to-dc converter are a +10-V-dc line, a +5-V-dc line, and a 0-V-dc line. The 0-V-dc line is connected to the ground plane of the transceiver board.

The isolated dc-to-dc converters achieve the required isolation by the use of an isolation transformer that is an integral part of the converter. Isolated dc-to-dc converters are available as off-the-shelf items from a variety of vendors.

The network protocol specifies that the voltage range on the power wires should be within the range of +11.28 to +15.75 V dc. The maximum current that these power wires are permitted to supply to the transceiver board is 500 mA.

Example 6.7 Why does the protocol specify the range of the power on the 12-V-dc line?

Solution When designing a transceiver board, the designer should assume that the 12-V-dc power supplied from the node board is within the specified range and design the board accordingly. This enables the end user to purchase the node board and the transceiver board from two different manufacturers.

6.5 Transceiver Chips

Due to the availability of chips known as Ethernet transceiver chips, a complete transceiver board may be implemented by using a single chip (and several resistors and capacitors). Our design utilizes the Intel

82502 Ethernet transceiver chip. This chip accomplishes the tasks of transmitting, receiving, and collision detecting. The 82502 is a CMOS chip and thus requires less power.

Example 6.8 Explain the choice of CMOS technology for the transceiver chip.

Solution The power consideration is important since power is brought from the node board to the transceiver board, which may be located as far as 50 m away from the node. Naturally, we would like to transfer as little power as possible over such a distance.

6.6 The Transceiver Circuit in Cheapernet

The transceiver circuit in Cheapernet is identical to the transceiver circuit of the Ethernet. However, in Cheapernet the transceiver circuit is implemented on the node board, and therefore there is no need for the AUI cable.

Example 6.9 How many ground planes should a Cheapernet node board have?

Solution The printed circuit board (PCB) of the node board in Cheapernet must be designed with two ground planes, one for the transceiver circuitry and the other for the rest of the node. The two ground planes must be isolated from each other.

6.7 The Node Board in Ethernet and Cheapernet

To design an Ethernet node board, the 802.3 node board model of Fig. 1.15 is used. This node board is shown again in Fig. 6.9, and it indicates the types of available integrated circuits utilized in our implementation.

6.7.1 The microprocessor section

The Intel 80186 processor is utilized in our design for the implementation of the microprocessor section (Fig. 6.9). The Intel 80186 is a high integration 16-bit processor. This processor is almost identical to the 80188 discussed in Chap. 4, but while the 80188 has only eight data lines (AD0 to AD7), the 80186 has 16 data lines (AD0 to AD15).

6.7.2 The LAN manager section

The Intel 82586 LAN controller is utilized for the implementation of the LAN manager section of the node board (Fig. 6.9). The Intel 82586 LAN controller is probably the most reliable and most popular LAN controller used in the industry in the high performance market. Nevertheless, the reader should be aware of the fact that there are

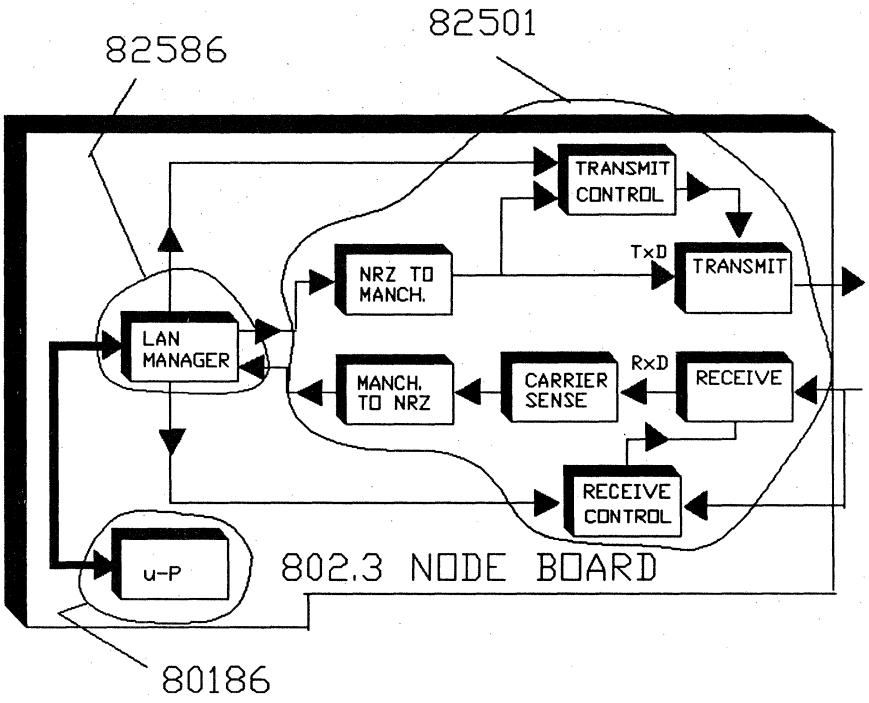


Figure 6.9 Realization of the node board.

more advanced LAN controller chips on the market. In particular, the Intel 82593 series* and the Intel 82596 series of LAN controller chips are more advanced than the 82586. We will discuss these advanced chips later in this chapter. Our implementation however starts with a detailed discussion that utilizes the 82586. This is a logical step that follows the evolution of LAN. In fact, all the material that is presented here about the 82586 remains valid for the more advanced chips, since the advanced 82596 family of LAN controllers can be made to work in a 82586 mode.* The fact that the advanced 82596 family of LAN controllers are 82586-software-compatible is a significant advantage that greatly improves development time and cost. Typically, the project engineer examines the required performance specifications of the system to be designed, and based on these, decides whether to utilize the 82586 or the 82596. Suppose, for example, that a system was designed using the 82586, and at a later time, it is decided to introduce an upgraded version that utilizes the 82596. The software development time and cost that were invested for developing the 82586 software and

*The 82593 is not software-compatible with the 82586.

the engineering experience that was gained during the development may still be utilized for the upgraded version.

6.7.3 The Manchester encoder-decoder sections and the transmit and receive sections

As shown in Fig. 6.9, the Intel 82501 is utilized for the implementation of both the Manchester encoder-decoder sections and the transmit and receive sections. The chip is called *Ethernet serial interface,* or ESI for short.

Example 6.10 Draw a block diagram for the Cheapernet node board.

Solution Figure 6.10 is a block diagram that incorporates our choice of chips for the implementation of the Cheapernet node. It is identical to the Ethernet implementation with the exception that the transceiver circuit is an integral part of the node board.

Example 6.11 How many ground planes should a Cheapernet node board have?

Solution The PCB of the Cheapernet node board must be designed with two ground planes, one for the transceiver circuitry and the other for the rest of the node. The two ground planes must be isolated from each other.

6.8 The Ethernet Serial Interface Chip

Connecting the 82501 to the 82586 LAN manager is a straight pin-to-pin connection. As illustrated in Fig. 6.10, the 82501 has two ports: one port is connected to the LAN manager, and the other port is connected to the transceiver.

The following discussion equates the various tasks that the 82501 performs with the tasks expected from the Manchester decoder-encoder, transmit, transmit control, receive, and receive control sections of Fig. 1.15.

6.8.1 Transmitting frames to the transceiver

The Intel 82501 receives single-ended NRZ frames from the 82586 via its TxD pin (Fig. 6.11), converts them to Manchester format, and transfers them as differential signals to the transceiver circuit via its TRMT pair of pins.

6.8.2 Enabling or disabling transmission to the network

As shown in Fig. 6.11, the LAN manager is responsible for enabling or disabling the transmission. This enabling or disabling of the transmis-

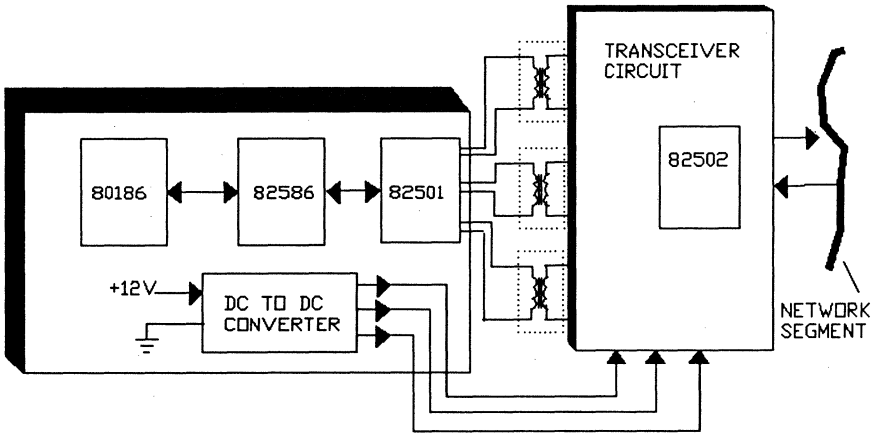


Figure 6.10 Realization of the transceiver board.

sion is accomplished by directly connecting the *request to send* (RTS) pin of the 82586 to the *transmit enable* (TEN) pin of the 82586. Transmission via the TRMT pins of the 82501 is therefore possible only if the 82586 asserts its RTS pin.

6.8.3 Implementing the jabbering procedure

As discussed in Chap. 1, jabbering is a procedure whereby a node is disabled if it transmits for more than a fixed amount of time. This procedure prevents a node from monopolizing the network cable. The jabbering procedure is automatically performed by the 82501. Once the TEN pin is asserted and transmission starts, the 82501 starts an internal timer. If the 82501 notices that the 82586 keeps transmitting for longer than 25 ms, the 82501 aborts the transmission of the frame.

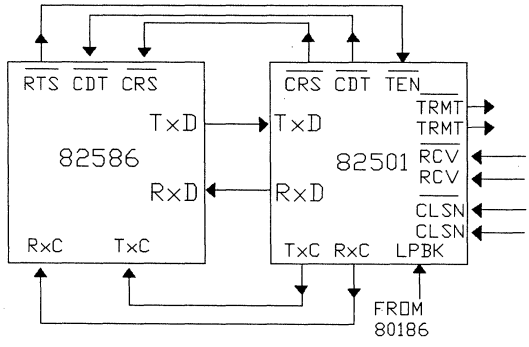


Figure 6.11 Connecting the 82586 to the 82501.

Example 6.12 Is 25 ms enough time for the transmission of a frame?

Solution From our study of the 802.3 frame in Chap. 1, we know that the maximum length of a frame is 1523 bytes. The time T(frame) that it takes to transmit 1523 bytes at a rate of 10 Mbit/s is

$$T(\text{Frame}) = \frac{1523 \times 8}{10 \text{ Mbit/s}} = 1.22 \text{ ms}$$

The 25 ms is therefore more than enough time.

6.8.4 Generating the idle signal

As discussed in Chap. 3, the last bits of the differential 802.3 frame must be 1s, and then the differential signal has to slowly reduce to zero (the process of generating the idle signal). Indeed, the 82501 generates the idle signal at the end of the frame in accordance with the network protocol. The 82501 is equipped with an input pin called the *Ethernet version 1* (ENETV1) input pin that if left open, causes the differential output to slowly reduce to zero. As shown in Fig. 6.12, we leave the ENETV1 pin open in our design.

6.9 Receiving Frames from the Transceiver

6.9.1 Manchester to NRZ conversion

The Intel 82501 receives differential Manchester frames from the transceiver circuit, converts them to NRZ frames, and transfers them

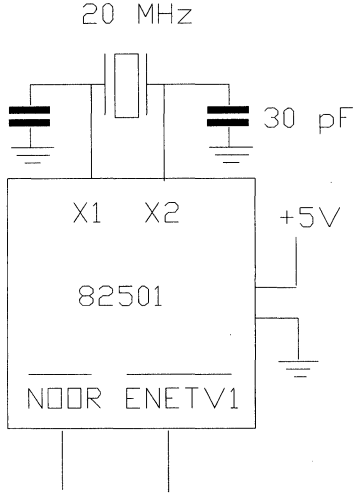


Figure 6.12 Connecting a 20-MHz crystal to the 82501.

as single-ended signals to the LAN manager circuit. As shown in Fig. 6.11, the 82501 receives the differential Manchester frames through its RCV and RCV pins, and transfers single-ended NRZ frames to the 82586 through its RxD pin.

6.9.2 Note about the conversion from Manchester to NRZ

When we studied the process of converting a Manchester signal back to an NRZ signal (Chap. 1), we indicated that this process is also called the *clock recovery* process. This process requires being as precise as possible. Indeed, performing clock recovery is an art by itself. The old analog way of performing clock recovery is to use an analog phase lock loop technique. The 82501 utilizes a sophisticated digital phase lock loop technique for performing the clock recovery which achieves excellent performance. The clock recovery section of the 82501 requires 14 bits to lock the clock. Thus, the first 14 bits of the frame are lost.

Example 6.13 Due to the operation of the clock recovery circuitry of the 82501, the first 14 bits of the frame are lost. Does this represent loss of data?

Solution As discussed in Chap. 1, the 802.3 frame starts with 56 alternating 1s and 0s (the preamble field). These bits are present in the frame to enable the clock recovery circuit to lock itself to the incoming bits. Thus, there is no loss of data, as long as the clock recovery circuit locks itself within the first 56 incoming bits.

6.9.3 Filtering and squelching

The 82501 is equipped with the appropriate circuitries to perform the filtering and squelching of the incoming signals at its RCV and CLSN pair of pins.

6.10 The Clock of the 82501

From our previous study of Manchester generation, we know that in order to convert an NRZ signal to a Manchester signal, a clock with double the frequency of the NRZ signal is required (Fig. 1.11). Thus, to generate a 10-MHz Manchester signal, the 82501 requires a 20-MHz crystal, which is shown together with its antinoise capacitors in Fig. 6.12.

6.10.1 The transmit and receive clocks

The 20-MHz clock is further divided internally by 2 to provide 10-MHz pulses to the *transmit clock* (TxC) pin and to the *receive clock* (RxC)

pin. As shown in Fig. 6.11, these pins are directly connected to the RxC and TxC input pins of the 82586.

Example 6.14 Why does the 82586 need the TxC and RxC clocks?

Solution The 82586 outputs bits into the TxD pin in accordance with the TxC clock, and it receives bits from the RxD pin in accordance with the RxC clock.

Example 6.15 The Ethernet and Cheapernet protocol specifies that the 10-Mbit/s data rate should have a minimum tolerance of 0.01 percent. What is then the required tolerance of the crystal frequency in Fig. 6.12?

Solution The 82501 generates the 10-MHz clocks (TxD and RxD) by dividing the crystal frequency by 2. Thus, the crystal should be a 20-MHz crystal with a 0.002 percent tolerance.

6.11 Informing the 82586 about Collision

Whenever the transceiver circuit detects a collision on the network cable, it transmits a differential signal to the CLSN and CLSN pins of the 82501 (Figs. 6.6 and 6.11). Upon receiving this signal, the 82501 asserts its collision-detected (CDT) output pin, which is directly connected to the CDT input pin of the 82586 (Fig. 6.11). The CDT pin is asserted for as long as the 82501 senses that its CLSN pair of pins receive the collision signal. The 82586 is thus notified whenever there is a collision on the network cable.

6.11.1 Informing the 82586 about network cable activity

When there is no activity on the network cable (signals are not traveling over the cable), the 82501 senses no valid data on its RCV pair of pins, and senses no signal on its CLSN pair of pins. Whenever the 82501 decides that there is no activity on the network cable, it responds by deactivating its carries sense (CRS) output pin, which is directly connected to the CRS input pin of the 82586 (Fig. 6.11). The 82586 is therefore informed whenever the network cable is free.

6.11.2 Two configurations

The 82501 can be configured to one of two ways. The configurations are accomplished by using the *no OR* (NOOR) input pin of the 82502 (Fig. 6.12).

In one configuration, the CRS output pin is asserted if there is an incoming valid Manchester signal to the RCV pins or if there is an incoming collision signal to the CLSN pins. As shown in Fig. 6.12, we leave the NOOR pin open, which causes the CRS signal to be asserted based on both the presence of the collision presence signal and the pres-

ence of valid data. Thus, in this configuration, the CSR is asserted as an indication that the network cable currently carries either valid data or the collision presence signal; in either case the network cable is busy.

In the other configuration, the NOOR pin is grounded, and the CRS output pin is asserted only if there is an incoming valid Manchester signal to the RCV pin (independent of the signal that is applied to the CLSN pin).

Example 6.16 Since in our design we are leaving both the ENETV1 pin and the NOOR pin open, it is recommended to connect a 0.022-μF capacitor between these two pins. This capacitor reduces the possibility of voltage fluctuation on these pins due to noise.

Example 6.17 Since in our design we leave the NOOR pin open, the CRS pin is asserted when there is either incoming valid data or collision. How is the 82586 informed about the presence of a collision?

Solution Via the CDT pin discussed above.

6.12 The Loop-Back Concept

As demonstrated, most of the available chips for implementing LAN equipment are designed by the IC manufacturers in such a way that in most cases, only a straight pin-to-pin connection is required for the implementation. Nevertheless, debugging a node board (during development or during normal operation) could become unmanageable since there are too many variables involved (e.g., did the node fail due to bad components on the node itself, a short on the AUI cable, or to the fact that the transceiver board is not functioning). To facilitate the debugging process, the *loop-back* feature is incorporated within many of the LAN chips. The loop-back feature enables the user to isolate the problem.

As shown in Fig. 6.11, the 82501 is equipped with an *internal loop-back* (LPBK) input pin. When this pin is asserted, the 82501 is in an internal loop-back mode. The internal loop-back mode is shown in Fig. 6.13. As shown, the 82501 in this mode returns the data that it receives through its TxD pin back to the RxD pin.

The loop-back feature enables the user to perform a complete test for all node operations except for the transmit and receive sections of the 82501, which are internally disconnected in this mode as shown in Fig. 6.13.

While in the loop-back mode, the 82586 sends a frame to the 82501, and upon receiving it, the 82586 is able to decide if the circuit is functioning properly. The 82501 is operating as a full duplex device in this mode, sending and receiving data at the same time.

At the end of the frame, the 82501 waits 1 μs and then starts generating the collision presence signal as if a real collision is occurring. This enables testing the node under collision conditions.

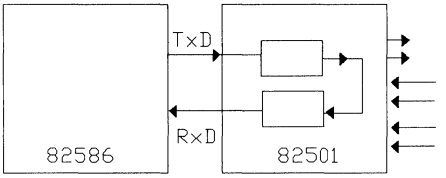


Figure 6.13 The loop-back feature.

Example 6.18 Does the 82586 receive back all the bits that it transmits in the internal loop-back mode?

Solution The internal loop-back mode emulates the exact same conditions that occur during normal operations. Thus, since it takes the clock recovery circuit 14 bit times to lock to the incoming bits, the first 14 bits of any frame are not returned to the RxD pin.

The internal loop-back mode enables the testing of the jabbering procedure as well. That is, the 82586 may send test frames that are longer than 25 ms and should observe that after 25 ms of transmission, no data are coming back to the RxD pin.

It is possible to disable the jabbering capability of the 82501 by applying 10 to 16 V through a 1-kΩ resistor to its LPBK pin.

Example 6.19 Figure 6.11 indicates that the LPBK pin of the 82501 is to be connected to the 80186 processor. Explain.

Solution As mentioned in Chap. 4, some of the peripheral chip select pins of the 80188 may be used as latches. The discussion in Chap. 4 regarding the PCS5/A1 pin (Fig. 4.2) applies to the 80186 processor as well.

As shown in Fig. 6.14, the PCS5/A1 pin of the 80186 is connected to the LPBK pin of the 82501. The program of the 80186 may therefore include a self-diagnostic section whereby the 80186 asserts its PCS5/A1 pin, causing the 82501 to enter the loop-back mode.

6.13 The Shared Memory Concept

The rest of this chapter discusses the operation of the 82586 LAN controller (Fig. 6.9). As we shall see, this LAN controller operates in a dif-

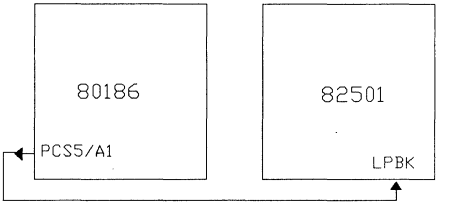


Figure 6.14 Entering the loop-back mode via software.

ferent way from the 82588 LAN controller discussed in Chap. 5. Both LAN controllers require DMA service for transferring bytes from the RAM (data to be transmitted) and to the RAM (received data). However, while the 82588 requires the host processor to prepare the DMA, the 82586 prepares the DMA by itself.

The 82586 is a sophisticated coprocessor. By *coprocessor* we mean that the 82586 operates together with another processor called the *host processor*. In our case, the host processor is the 80186 processor. The 80186 serves as the master of the 82586, issuing commands to 82586. The 80186 processor then monitors the execution of the commands and fetches the results of the executions. The 82586 is capable of executing complex commands, all by itself, without any help from the host processor.

Figure 6.15 is a block diagram showing how the 82586 and the host processor are connected. As shown, the 80186 and the 82586 share the same memory chips. Both the 82586 and the host processor have their data, address, read (RD), and write (WR) pins connected to the memory chips, so each is capable of reading bytes from the memory and writing bytes to the memory. Of course they cannot access the memory simultaneously; only one device can access the memory chips at any given time.

To avoid chaos on the data and address buses, rules are established to dictate which of the devices can access the memory. The HOLD and HOLD pins of the devices are connected as shown to accomplish these access rules.

Initially, the 82586 floats its data, address, RD, and WR pins. Being the master, the 80186 does not have to request permission to access the memory. It can access the memory without worrying about the 82586 creating conflict (contention) on the data, address, RD, and WR lines.

On the other hand, being the slave, the 82586 must request permission from the 80186 to access the memory. It does so by asserting its

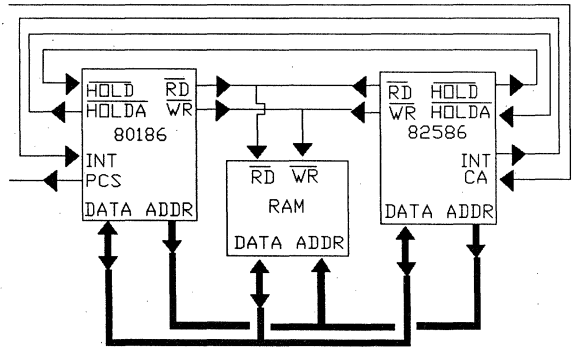


Figure 6.15 Shared memory (between the host processor and the LAN controller).

HOLD pin, which is directly connected to the HOLD pin of the host processor (Fig. 6.15). Once the host processor senses that its HOLD pin is asserted, it understands that the 82586 wishes to access the memory and may respond by granting the memory to the 82586 by asserting its HOLD acknowledgment (HOLDA) pin, which is directly connected to the HOLDA pin of the 82586, and concurrently releasing the bus. Releasing the bus means that the 80186 processor floats its own data, address, RD, and WR lines.

Once the 82586 senses that its HOLDA pin is asserted, it accesses the memory chips. The 80186 is forbidden from accessing the memory for as long as it senses that its HOLD pin is still asserted. Eventually, the 82586 completes it memory access, ceases the assertion of the HOLD pin, and floats its data, address, RD, and WR lines. The 80186 then gains control of the memory and can access it if it so desires.

6.14 The Operation

To operate the 82586, the host processor writes command bytes into a predetermined location in the shared memory and then asserts the *channel attention* (CA) pin of the 82586 (Fig. 6.15). For example, one of the commands that the 80186 issues to the 82586 is the TRANSMIT command. The command is accompanied by a set of parameters. One of these parameters indicates the address location in memory of the bytes to be transmitted to the network.

Upon sensing that the CA pin is asserted, the 82586 understands that the host processor has something for it in the shared memory. The 82586 then requests permission to access the memory and fetches the command from the predetermined location in memory. The 82586, all by itself, decodes the command and extracts from the command the address location of the data. Now that the 82586 knows the address location of the data, it fetches the data from memory, constructs a frame, does all the necessary procedures specified in the CSMA/CD 802.3 protocol, and sends the frame to the 82501. Upon completing the transmission, the 82586 writes to a predetermined location in memory, reporting whether the transmission was successful. The 82586 then asserts the INT pin of the host processor (Fig. 6.15). Upon sensing that its INT pin is asserted, the host processor fetches the results of the command execution from the predetermined location in memory.

In a similar way, if the host processor enables the 82586 to receive frames from the network, the 82586 receives and processes the frames from the network in strict compliance with the network protocol and stores the received data in a predetermined address location in memory. Once the 82586 deposits the received data into the memory, it asserts the INT pin. The 80186 responds by fetching the data from memory.

The program of the 82586 is already embedded on chip, so there is no need to write a program for the 82586. Once the host processor writes the command into the shared memory and then activates the CA pin of the 82586, the 82586 does all the rest by itself. Since the 82586 is involved with a lot of byte transfers to and from the memory, it has four DMA controllers on chip. This makes the byte-transfer process very fast and efficient. The 82586 programs its own DMA controllers in accordance with the command that it is decoding.

6.15 Writing the Program for the 80186

Writing the program for the 80186 is a fairly straightforward process that involves issuing commands to the 82586 by writing them into the predetermined location in memory and then interrupting the 82586 by asserting the CA pin of the 82586. The program then surveys the various memory locations, examining the results of the command execution and fetching the received frames from the memory.

6.15.1 Configuring and setting the 82586

The first command that the 80186 must issue is the CONFIGURE command. This command configures the 82586 with various parameters.

Example 6.20 One of the parameters that accompanies the CONFIGURE command is the parameter that determines whether the 82586 would work in the promiscuous mode. The *promiscuous mode* is a mode whereby the 82586 accepts any incoming frame without considering the destination address contained in the frame.

Once the CONFIGURE command is issued, the 82586 sets itself in accordance with the parameters that accompany the command, writes into the memory the results of the execution (e.g., execution completed successfully), and interrupts the 80186. The program of the 80186 should check that the command was executed without any problems, and if it discovers that the command was executed successfully, it should proceed to the next step of setting the 82586.

The next step is to assign the source address of the node by sending the SETUP INDIVIDUAL ADDRESS command accompanied with the assigned source address.

The next step is the assignment of the multicast address (multicasting was discussed in Chap. 1).

6.15.2 Frame reception

Frame reception is possible by writing the proper bit sequence into a certain location in memory. Every time the 82586 visits the memory,

it checks to see if that certain location contains the code of ENABLE FRAME RECEPTION or DISABLE FRAME RECEPTION. The 80186 should enable the reception of frames only if it already allocated space in memory for the received data. The allocation of space is accomplished by preparing a sequence of bytes called RECEIVE BUFFER DESCRIPTORS. These descriptors inform the 82586 of the memory location allocated for the received data. Upon receiving and processing frames from the network, the 82586 stores the received frame to the location in memory dictated by the RECEIVE BUFFER DESCRIPTORS.

6.15.3 Transmitting frames

Once the 80186 completes preparing the data for transmission, it deposits the data into memory and issues the TRANSMIT command. The 82586 responds by extracting the memory location of the data from the parameters of the TRANSMIT command, fetching the data from the memory, constructing a frame, and transmitting it to the 82501.

6.15.4 Other available commands

The other commands that may be issued by the 80186 are the NOP command (no operation command), the DIAGNOSE command, the TDR command, and the DUMP command:

The DIAGNOSE command causes the 82586 to perform a self-test procedure and to store the results of the diagnostics in memory.

The DUMP command causes the 82586 to store the contents of its internal registers in memory. This feature is provided as a debugging tool during development time.

The TDR command is the TIME DOMAIN REFLECTOMETER command, which is utilized as a diagnostic tool. Upon executing this command, the 82586 transmits 1s to the network and keeps track of how long it takes for a response to return to the node.

6.16 Command Example

This section is included to provide the reader with a feeling of what it takes to program the 82586 LAN controller.

6.16.1 The IA-SETUP command

Prior to issuing the TRANSMIT command and prior to enabling the reception of frames, the 80186 must assign a source address to the 82586. The source address is assigned by using the *individual address setup* (IA-SETUP) command.

The format of the IA-SETUP command is shown in Fig. 6.16. As shown, the command is composed of 12 bytes (6 words). The 80186 prepares these 12 bytes and stores them in RAM.

6.16.2 The seventh, eighth, ninth, tenth, eleventh, and twelfth bytes of the command

These 6 bytes contain the source address that is being assigned to the 82586. The seventh byte contains the least significant byte of the source address, and the twelfth byte contains the most significant byte of the source address.

Example 6.21 Are there any rules regarding the least significant bit of the source address?

Solution Yes. As discussed in Chap. 1, the 802.3 protocol dictates that the least significant bit of the individual source address must be 0.

6.16.3 The fifth and sixth bytes of the command

The fifth and sixth bytes of the command contain the offset address of the next command. That is, once the 82586 completes the execution of this command, it should execute the next command. The location in RAM of the next command is indicated by the *line offset* (the fifth and sixth bytes of the command).

Example 6.22 Suppose that the first byte of this command is located at address F0000, and the next command is located at address F0010. What should be the content of the line offset?

Solution The line offset should contain the value 0010.

6.16.4 Bits 2,1, and 0 of the second word of the command

As shown in Fig. 6.6, these bits contain the values 001. The 001 is the code of the IA-SETUP command.

Example 6.23 Three bits could represent a total of 8 combinations; each combination represents a different command. The 001 represents one of the eight possible commands, the IA-SETUP command.

6.16.5 Bits 3 through 12 of the second word of the command

These bits are not used. The 80188 may fill these bits with either 0s or 1s.

6.16.6 Bit 15 of the second word of the command

As shown in Fig. 16.6, bit 15 of the second word of the command is denoted as the EL bit, which is the abbreviation of *end list.* Once the 82586 completes the execution of this command, it examines the content of the EL bit. If EL is 0, the 82586 fetches the next command from RAM. (As discussed above, the 82586 calculates the address of the first byte of the next command by adding the address of the first byte of this command to the offset address given in the fifth and sixth bytes of this command.)

If the 82586 finds that the EL bit is 1, the 82586 concludes that there are no more commands to be executed.

Example 6.24 How does the 82586 know the address of this current command?

Solution There are two ways for the 82586 to know about the address of this current command:

1. It is possible that the 82586 completed the execution of a previous command, the previous command had its EL bit equal to 0, and its line offset pointing to this current command.
2. Prior to issuing commands, the 80186 stores certain instructions in the RAM. The RAM location of these instructions is called the system control block, or the *mail box.* One of these instructions is the instruction to start executing commands. This instruction contains the address of the first command. Once the 80186 completes preparing the mail box, it stores the address location of the mail box into location FFFFF6.

Upon reset, the 82586 always automatically fetches the address of the mail box from address FFFFF6. Now that the 82586 knows the address of the mail box, it proceeds and fetches the instruction stored in the mail box. As mentioned, one of the instructions stored in the mail box might be an instruction to start executing commands, and the address of the first command is contained within the instruction. The above process is shown in Fig. 6.17.

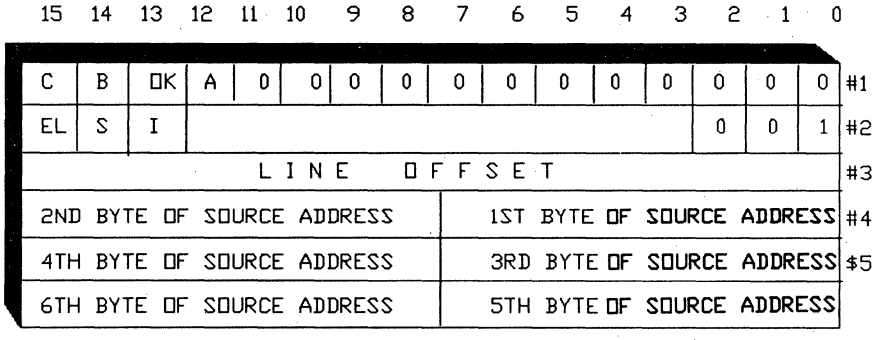


Figure 6.16 Programming the 82596 LAN controller.

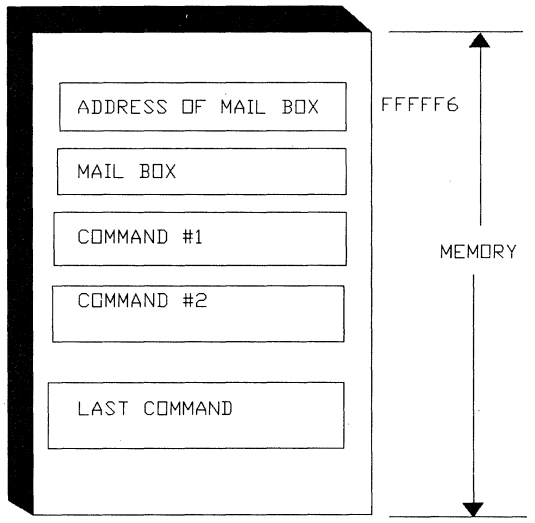


Figure 6.17 Placing information in the share memory.

6.16.7 Bit 14 of the second word of the command

As shown in Fig. 6.16, bit 14 of the second word of the command is denoted as the S bit, which is the abbreviation of *suspend.* Once the 82586 completes the execution of this command, it examines the content of the S bit. If the S bit is 1, the 82586 suspends further command executions (after completing executing this current command).

6.16.8 Bit 13 of the second word of the command

As shown in Fig. 6.16, bit 13 of the second word of the command is denoted as the I bit, which is the abbreviation of *interrupt.* Once the 82586 completes the execution of this command, it examines the content of the I bit. If the I bit is 1, the 82586 interrupts the 80186 (after completing the execution of this current command).

6.16.9 The first word of the command

The 80186 should write all 0s into the first word of the command. Upon executing the command, the 82586 writes into the location of the first word of the command, reporting about the execution of the command.

The 80186 examines the first word of the command to learn whether the 82586 "visited" this command already and whether the execution of the command was successful. Bits 11 through 0 of the first word of the command are not used.

6.16.10 Bit 15 of the first word of the command

As shown in Fig. 6.16, bit 15 of the first word of the command is denoted as the C bit, which is the abbreviation of *command completed.* Once the 82586 completes executing the command (successfully or otherwise), it writes a 1 to this bit location.

The 80186 may read this byte, and if it finds that the C bit is 1, it concludes that the 82586 completed the execution of this command.

6.16.11 Bit 14 of the first word of the command

As shown in Fig. 6.16, bit 14 of the first word of the command is denoted as the B bit, which is the abbreviation of *busy executing.* Once the 82586 starts executing the command, it writes a 1 to this bit location.

The 80186 may read this byte, and if it finds that the B bit is 1, it concludes that the 82586 is currently executing the command.

6.16.12 Bit 13 of the first word of the command

As shown in Fig. 16.6, bit 13 of the first word of the command is denoted as the OK bit. The 80186 may read this byte, and if it finds that the OK bit is 1, it concludes that the 82586 completed the execution of this command without encountering any errors during the execution.

6.16.13 Bit 12 of the first word of the command

As shown in Fig. 16.6, bit 12 of the first word of the command is denoted as the A bit, which is the abbreviation of *aborted.* The 80186 may read this byte, and if it finds that the A bit is 1, it concludes that the 82586 aborted the execution of this command (for a variety of reasons).

6.17 The High Performance 32-Bit LAN Coprocessors

As discussed in Sec. 6.7.2, the Intel 82596 series of chips are more advanced than the 82586. The three chips in the 82596 series are

- 82596DX (supports a 32-bit bus)
- 82596SX (supports 16-bit bus)
- 82596CA (supports 32-bit burst bus).

The two chips in the 82593 series are

- 82593SX (an 8- or 16-bit bus).
- 82593BX (an 8-bit bus).

Readers who studied the material presented in this chapter about the 82586 will have little difficulty understanding the terms and terminology used in the Intel 82596 and 82593 data sheets.

There are basically two reasons for utilizing the 82596 instead of the 82586. One obvious reason is the fact that the 82596 is able to process data faster than the 82586. The other reason is that the 82596 was designed in such a way that integrating the LAN onto a motherboard that contains the 386 (or 486) CPU is an easy task. Of course, this is to be expected, since after all, it is the Intel corporation that designed the 386 and 486 CPUs. When designing a motherboard that has a LAN controller on board, the designer should match the CPU to the appropriate LAN controller chip, i.e., the 82596DX is the appropriate LAN controller for a motherboard that has a 32-bit 386DX CPU, the 82596SX is the perfect match for a motherboard that has a 16-bit 386SX CPU, and so on.

Until recently, node boards were purchased as I/O cards, which the LAN integrator (i.e., the person who is in charge of installing the hardware and software of the LAN system) has to insert into one of the I/O slots of the PC. By having the node as an actual integral part of the motherboard, the end users do not have to worry about making a separate purchase for a node board. By purchasing a PC that has a LAN circuitry on the motherboard, the end user does not waste an I/O slot and does not have to worry whether the plugged-in node board would or would not be compatible with the PC.

As can be testified to by many LAN integrators, the main nightmare of implementing node boards is the "simple" task of plugging the node board into the I/O slot. Many things can go wrong during this easy process of just plugging the node board into an I/O slot. When plugging in the node board, the housing of the PC has to be opened, a free slot has to be allocated, and the address decoding and the interrupt switches need to be set to the proper address and interrupt number. Once a new node board has been plugged into the I/O slot, isolating a problem with the node may require several hours of trial and error adjustments. The difficulty in isolating a node problem is due to the fact that usually the LAN software is not installed yet. If for some reason the network does not function well, the user has to determine whether the cause of the problem is an improper setting of either the software or the node board (e.g., a simple address-interrupt setting that conflicts with another I/O setting in the system, or malfunctioning of the node board due to mis-

handling of the board during shipping of the product or during the installation of the board). One common nightmare is the case where the user happily finds that the node board just installed indeed functions properly, but another I/O board in the system (such as a graphic processor card, hard-card controller, etc.) does not function properly whenever the LAN is in operation. This scenario is particularly unpleasant for the LAN integrator who is not supposed to get involved with the system's other portions and tasks.

The ideal solution is to have the LAN board as an integral part of the motherboard. This frees an I/O slot for other peripherals, saves the cost of the node board and its installation, and assures the end user that all the hardware problems have already been taken care of by the manufacturer of the motherboard. The end user expects the node to be tested and to be in functional condition just like all the other components on the motherboard (e.g., the CPU, RAM, ROM, etc.). The end user may immediately proceed with the software installation of the LAN without having to be concerned about the node.

The reader should be aware that although the 82596 is an ideal candidate for integrating a LAN onto the motherboard of the PC, it may also be utilized in designing a plugged-in node card. Such node cards with greater capabilities of processing data are needed by file servers and workstations that are connected to a LAN. The CPU of the PC that has such an advanced node is able to perform higher level functions (such as MS windows), while leaving all the LAN tasks to be performed by the advanced node without CPU intervention.

6.18 Summary

This chapter discusses the Ethernet and Cheapernet topologies and implementations. These LANs are designed to transfer data at a rate of 10 Mbit/s. However, unlike the Starlan, these LANs require the use of coaxial cables. The next chapter discusses the latest CSMA/CD LAN standard, the 10BASE-T LAN. This LAN (also called twisted pair Ethernet LAN) is capable of transferring data at a rate of 10 Mbit/s over twisted pairs of unshielded wires.

Chapter

7

Twisted Pair Ethernet

7.1 The Need for an Additional 802.3 LAN

As discussed in Chap. 1, the 802.3 LANs are *probabilistic;* that is, there is no guarantee of when a node will be able to access the network cable. On the other hand, the IEEE 802.4 and 802.5 LANs are *deterministic,* since in these LANs it is known in advance when a node will be able to access the network cable. Deterministic-type LANs are usually used in real-time applications such as in automatic manufacturing environments.

Generally speaking, the probabilistic LANs are more suitable for office applications, where the cost of the LAN (together with its installation cost) plays an important role. The IEEE 802.3 1BASE5 Starlan utilizes twisted pairs of unshielded regular telephone wires, which are far less expensive than the cables utilized in the 802.3 10BASE5 Ethernet or the 10BASE2 802.3 Cheapernet LANs. In addition, the Starlan topology enables the end user to take advantage of the existing unused telephone wiring. However, the 1BASE5 Starlan specifies a maximum data rate of only 1 Mbit/s (while the Ethernet and Cheapernet specify a data rate of 10 Mbit/s).

The next logical step in developing probabilistic LANs is therefore a LAN that still utilizes the regular telephone wires and takes advantage of unused telephone wiring, but has a data rate of 10 Mbit/s.

The LAN should utilize the CSMA/CD access method for the following main reasons:

- The CSMA/CD access method specified in the IEEE 802.3 protocol is a method that has proved itself to be reliable.
- By using the CSMA/CD as the access method, the newly developed LAN would be able to interface with already purchased Ethernet and Cheapernet equipment.

7.1.1 The Twisted Pair Ethernet (TPE) LAN, IEEE 802.3 10BASE-T

TPE is the abbreviation of *Twisted Pair Ethernet* LAN. As implied by the name, this LAN is implemented by using unshielded twisted pair of wires (regular telephone wires), and it allows the transfer of data at a rate of 10 Mbit/s. As we shall soon see, the TPE topology is such that the already installed unused telephone wires in buildings may be used for installing the TPE LAN. The IEEE protocol that defines this LAN is the IEEE 802.3 10BASE-T.

7.2 Connecting Nodes to the Multiport Repeater Board

Figure 7.1 shows an example of a Twisted Pair Ethernet network. The network has a star topology and is composed of a *multiport repeater board,* or MPR for short. The MPR shown has a provision for connecting 12 nodes (11 MAUs are on the bottom of the MPR, and the twelfth MAU is on the right side of the MPR).

The node boards in the network of Fig. 7.1 are connected directly to the MPR board via a twisted pair of wires. Each twisted pair has a maximum length of 100 m.

To be able to connect a node directly to the multiport repeater board via a twisted pair of wires, each node must contain a medium attachment unit (MAU) circuitry. These MAU circuits are designed to drive twisted pairs of wires. The node boards in Fig. 7.1 have their MAU circuits embedded within the node boards.

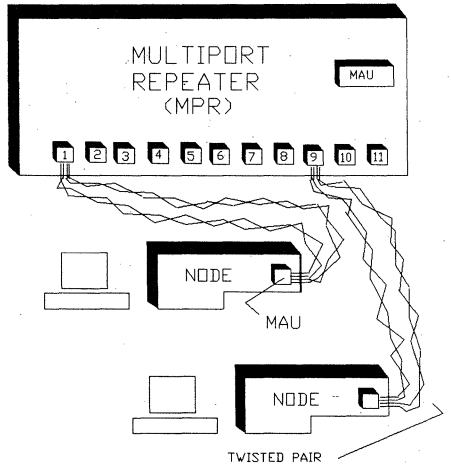


Figure 7.1 The multiport repeater (MPR).

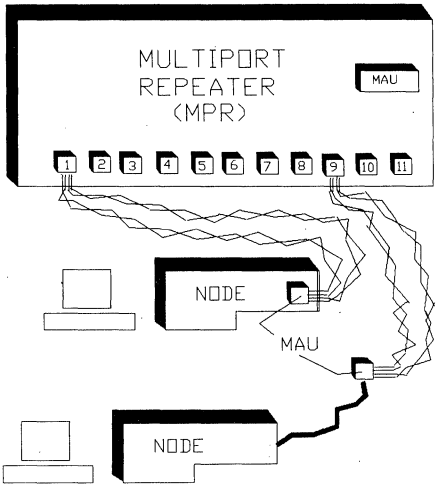


Figure 7.2 Connecting node boards to the MPR.

7.2.1 Connecting nodes that do not have embedded twisted-pair-MAU to the MPR

Figure 7.2 shows how a regular Ethernet node board is connected to the MPR. The Ethernet node board (the lower node board of Fig. 7.2) does not have a twisted-pair-MAU circuit on board. Rather, the Ethernet node board has a circuitry that is designed to drive AUI (attachment unit interface) cables. As discussed in Chap. 6, the AUI cable should not exceed 50 m. The AUI cable connects the Ethernet node board to a special external MAU board. The other side of the external MAU board is equipped with the proper circuitry to connect the board to the MPR via a twisted pair of wires that has a maximum length of 100 m.

By using these special external MAU boards, the end user is able to utilize already purchased Ethernet node boards in TPE LANs. The Ethernet node may be located as far as 150 m away from the MPR (100 m is the maximum length of the twisted pair wires, and 50 m is the maximum length of the AUI cable).

7.2.2 Connecting an additional multiport repeater

The network of Fig. 7.2 may be further extended to the one shown on Fig. 7.3, where an additional multiport repeater is now added to the network. The two multiport repeaters are connected via a twisted pair of wires that should not exceed 100 m.

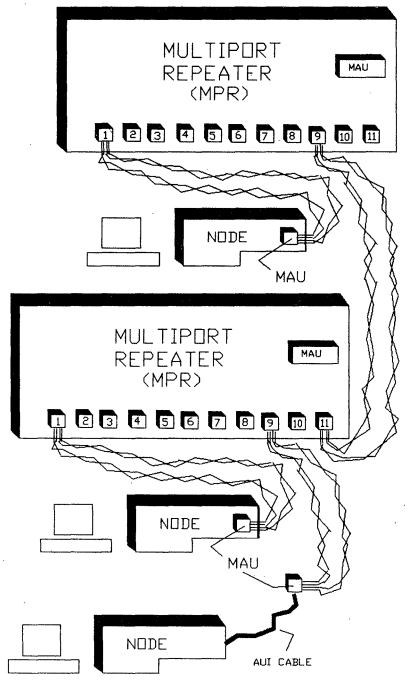


Figure 7.3 Adding additional MPR levels.

7.2.3 Connecting an Ethernet network to the Multiport repeater

Each multiport repeater contains an MAU circuitry (shown unconnected on the right side of each of the multiport repeaters in Figs. 7.1, 7.2, and 7.3). The multiport repeater is equipped with a MAU circuitry to allow a connection to an Ethernet network as shown in Fig. 7.4.

Example 7.1 What is the difference between the 11 ports (MAU circuits) shown on the bottom of the multiport repeater and the MAU circuit shown on the right side of the multiport repeater?

Solution The 11 MAU circuits that are shown on the bottom of the MPR are designed to accommodate data transfer over a twisted pair of wires. These twisted pairs can be extended to a maximum length of 100 m.

The MAU of the multiport repeater that is shown on the right side of the MPR is designed to accommodate data transfer over an AUI cable. The AUI cable can be extended to a maximum length of 50 m and can be connected to the MAU (transceiver) board of an Ethernet segment.

Example 7.2 What is the advantage of utilizing the multiport repeater concept?

Solution The advantage of utilizing the multiport repeater concept is illustrated in Fig. 7.4. Different networks with different topologies may be combined to form

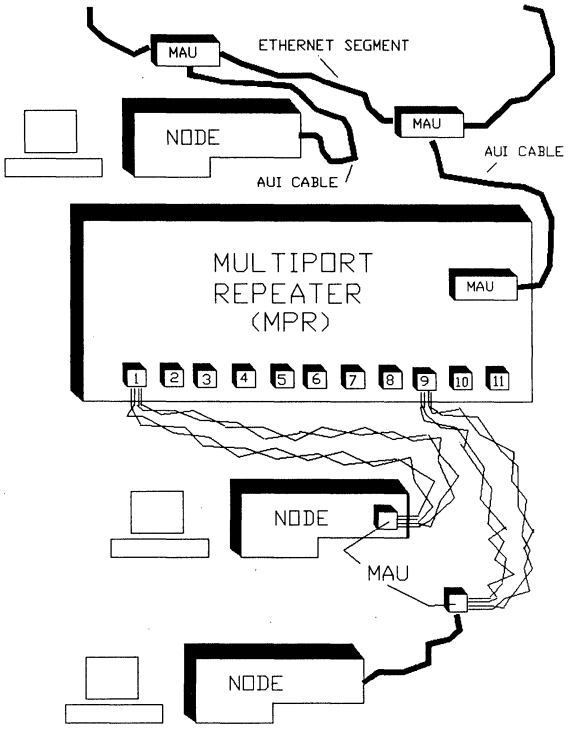


Figure 7.4 Adding Ethernet segment to the MPR.

one giant network. Of course, whatever is connected to the network must comply with the 802.3 CSMA/CD protocol.

Example 7.3 What are the requirements of a node that is connected to a multiple port repeater?

Solution The multiple port repeater board expects each node that is connected to it to be able to transmit and receive frames at a rate of 10 Mbit/s. All the other node requirements discussed in previous chapters apply as well (e.g., each node should be able to transmit and receive Manchester differential signals, be able to detect and handle a collision, etc.).

7.2.4 Maximum configuration of the Twisted Pair Ethernet

The Twisted Pair Ethernet defines the maximum configuration as one that has a maximum of four levels of connected multiport repeater boards. A maximum of 1024 nodes is permitted in the Twisted Pair Ethernet network.

Example 7.4 Show how the Twisted Pair Ethernet network may utilize the already installed unused telephone wires.

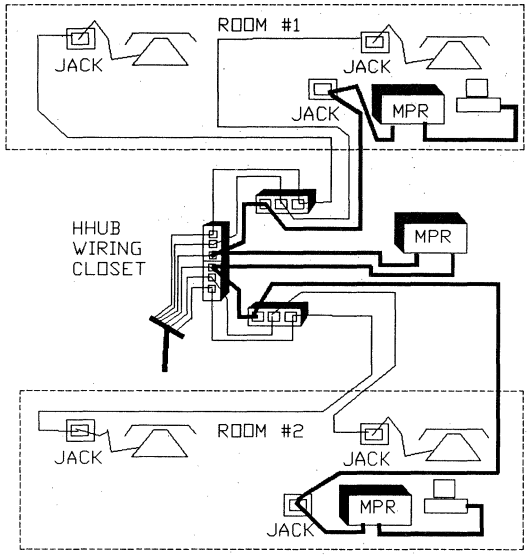


Figure 7.5 Utilizing existing unused telephone wires.

Solution Chapter 2 includes a detailed discussion of telephone wiring. Figure 2.15 is an example illustrating how the telephone company installs telephone service for a typical building. (The two heavy lines of Fig. 2.15 represent installed unused telephone wires. Each heavy line consists of two pairs of twisted wires.)

Figure 7.5 shows an arrangement of a Twisted Pair Ethernet network that takes advantage of the already installed telephone wires. The arrangement conforms to the required topology and wiring specifications of the Twisted Pair Ethernet network.

As stated in Chap. 5, although Fig. 2.15 shows only two extra unused telephone cables, typically, the telephone company installs many more extra connections.

7.3 The Medium Dependent Interface Connector

The twisted pairs that connect a multiport repeater to a node board (Fig. 7.2), or a multiport repeater to another multiport repeater (Fig. 7.3), may be extended to a maximum length of 100 m, and are called the *link segment* (Fig. 7.6).

The connector that connects the link segment to the MAU circuit is called the *medium dependent interface connector,* or MDI connector for short. Each link segment has two MDI connectors attached to it, one at each end of the link.

The MDI connector is the RJ-45 eight-pin connector (Fig. 7.7). Although there are eight pins to the connector, only four pins are required for connecting the two pairs of twisted wires.

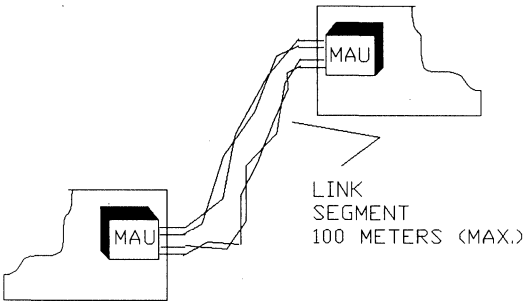


Figure 7.6 The link segment.

The following example illustrates how two twisted-pair-MAU circuits are connected by using the RJ-45 connectors.

Example 7.5 Show how the two twisted-pair-MAU circuits of Fig. 7.6 are connected when two RJ-45 connectors are utilized.

Solution Figure 7.8 ilustrates how an RJ-45 connector is connected to the twisted-pair-MAU. The transmitter of the MAU has two wires: one is denoted as the TD+ wire, and the other is denoted as the TD– wire. The receiver of the MAU also has two wires: one is denoted as the RD+ wire, and the other is denoted as the RD– wire.

Pins 1 and 2 of the RJ-45 connector are connected to the transmitter of the MAU, and pins 3 and 6 of the RJ-45 connector are connected to the receiver of the MAU.

When attaching the MAU circuit on either side of the link segment, care must be exercised to ensure that the transmitters are connected to the receivers (i.e., do not connect receiver to receiver and transmitter to transmitter). In addition, the connections must be accomplished with the proper differential polarities. That is, the TD+ wire on one side of the link segment must be connected to the RD+ wire of the other side of the link segment, and the TD– were on one side of the link segment must be connected to the RD– wire of the other side of the link segment (see Table 7.1).

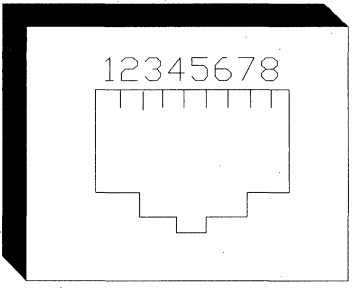


Figure 7.7 The RJ-45 connector.

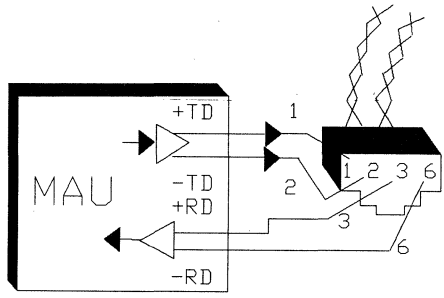


Figure 7.8 Connecting the MAU to the RJ-45 connector.

7.4 The Node Board in a Twisted Pair Ethernet

The node board in a Twisted Pair Ethernet network is a "regular" 802.3 node board of the type discussed in previous chapters, and thus the 802.3 node board model of Fig. 1.15 is utilized for the design.

Of course, the transmitter and receiver sections of the node must be designed to accommodate signal transmitting and receiving via a twisted pair of wires for a maximum distance of 100 m. The Twisted Pair Ethernet node board shown in Fig. 7.9 is identical to the Ethernet node board, except for its transmitter and receiver sections.

Example 7.6 The lower node of Fig. 7.2 is a regular Ethernet node and cannot be connected directly to the multiport repeater, since it does not have the proper twisted pair MAU. As shown, the node is connected to an external MAU circuit via an AUI cable. This MAU board is assumed to contain the proper twisted pair transceiver circuit on its other end and thus may be connected to the multiport repeater via a 100-m twisted pair of wires.

Example 7.7 The top node of Fig. 7.2 is a Twisted Pair Ethernet node and can be connected directly to the multiport repeater, since it does contain the proper twisted pair MAU.

7.5 Design of a Twisted Pair Ethernet Node Board with an Analog Front End

Figure 7.10 illustrates a design of a Twisted Pair Ethernet node board. That is, the node board is a regular Ethernet node board, but it contains the proper transmitter and receiver circuits, which allow the connection of this board directly to the multiport repeater via twisted pairs.

As shown in Fig. 7.10, the design takes advantage of existing LAN chips:

- *The microprocessor section.* The 80186 is utilized as the processor for the microprocessor section.

TABLE 7.1

One side of the link segment	The other side of the link segment
TD+	RD+
TD–	RD–
RD+	TD+
RD–	TD–

- *The LAN manager.* The LAN controller 82586 is utilized for implementing the LAN manager section (the same LAN controller chip utilized in the Ethernet node of Chap. 6).
- *The Manchester encoder-decoder and carrier sense sections.* The Manchester encoder-decoder and carrier sense sections are implemented by using the Intel 82504.
- *The transmit and receive sections.* The transmit and receive circuits are designed to meet the requirements of a twisted pair interface. In a Twisted Pair Ethernet, these circuits are called *analog front end.*

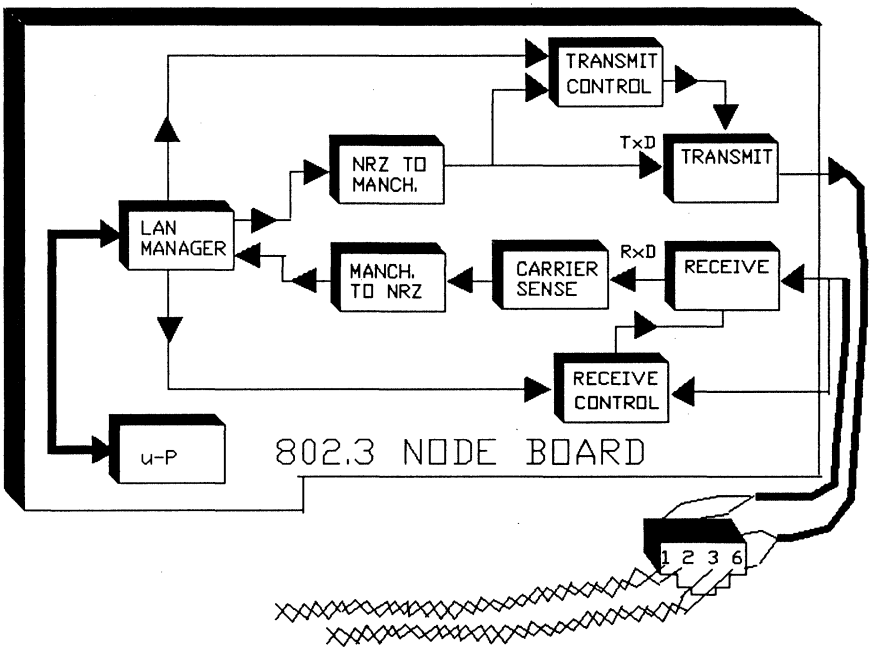


Figure 7.9 The twisted pair 802.3 node board.

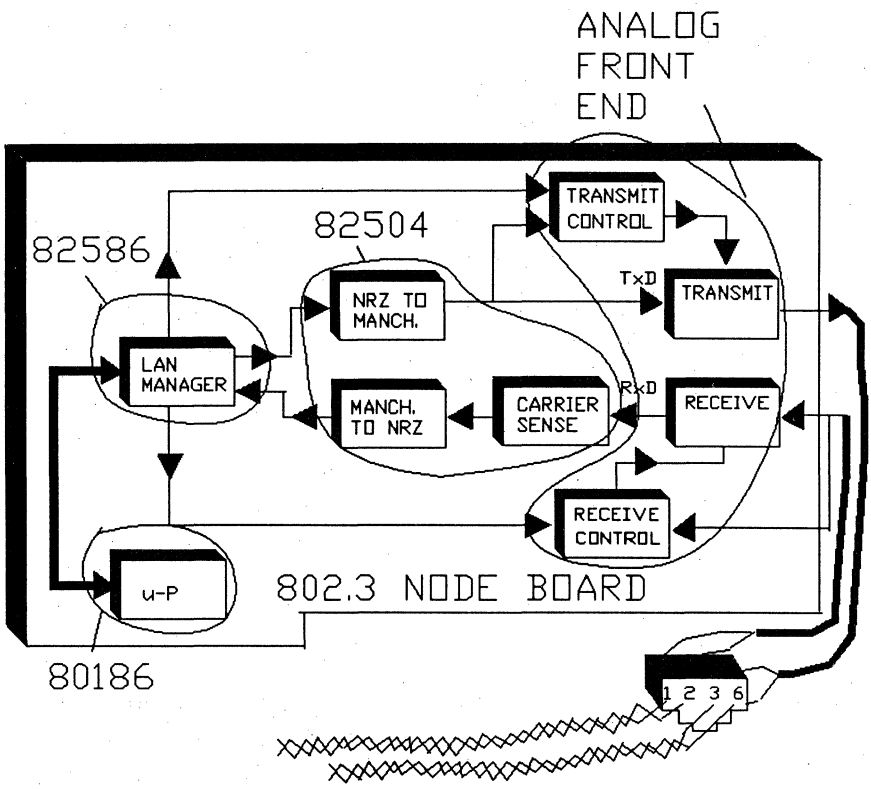


Figure 7.10 Realization of the 802.3 twisted pair node board.

7.6 The Analog Front End

The various components of the analog front end section of Fig. 7.10 are shown in Fig. 7.11. The analog front end section is more complicated and composed of more components than the transmitter and receiver of a regular Ethernet node. Of course, this is to be expected, since in a Twisted Pair Ethernet network, the signal generated by the node has to travel over unshielded wires and therefore must be generated more carefully.

The analog front end consists of

- *The transmitter section.* The transmitter section (Fig. 7.11) accepts Manchester single-ended signals from the 82504 and converts them to differential signals. The 82504 is equipped with an output pin called a *twisted pair enable,* or TPEN for short. The 82504 enables and disables the transmitter by asserting the TPEN pin (Fig. 7.12).
- *The receiver section.* The receiver section (Fig. 7.11) accepts Manchester differential signals from the previous section, converts

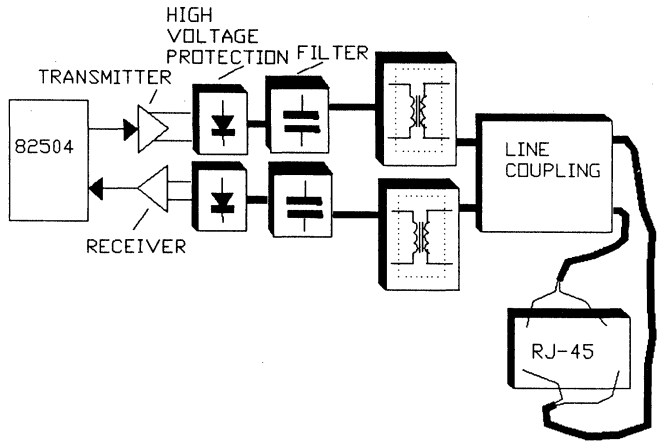


Figure 7.11 The analog front end section of the twisted pair node board.

the signals to NRZ format, and feeds the signals to the 82504. This section has to incorporate in it a squelch circuit to allow only valid data to pass to the 82504 (squelch circuit design is discussed in Chap. 3). The 82504 is equipped with a pin called twisted pair sense, or TPS for short (Fig. 7.12). The output of the squelch circuit is fed to the TPS input pin, indicating the presence of a valid incoming signal.

- *The high voltage protection circuits.* The high voltage protection sections of Fig. 7.11 protect the node from high voltage that may occur on the twisted pairs. It is implemented by using clamping diodes. Two high voltage protection circuits are used: one is connected to the transmitter (Fig. 7.13), and the other is connected to the receiver (Fig. 7.14).

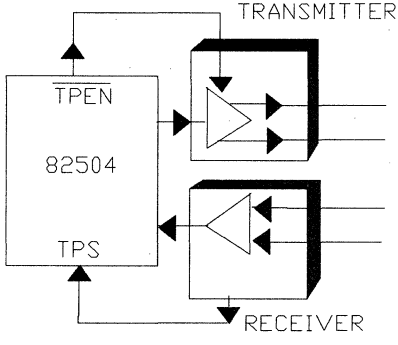


Figure 7.12 The 82504.

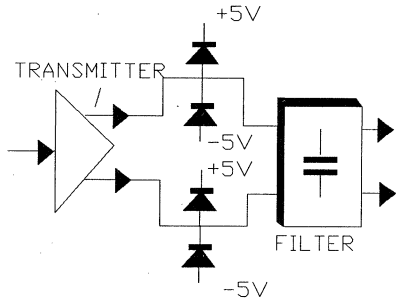


Figure 7.13 Connecting clamping diodes and filter to the transmitter of the twisted pair node board.

- *The filters.* The filters of Fig. 7.11 filter out the high frequency noise. There are two filters, one on the transmitter side and one on the receiver side. Figure 7.15 illustrates the filter topology.
- *The transformers.* The transformers of Fig. 7.11 provide the dc isolation between the twisted pair of wires and the node.
- *The line coupling.* The line coupling of Fig. 7.11 is implemented by using chokes, and it is used to further remove noise generated by the fast rise and fall time of the data transmitted and received.

7.7 The Transceiver Serial Interface

The 82504 used in the design of the Twisted Pair Ethernet node (Fig. 7.10) is also called the *transceiver serial interface,* or TSI for short. The 82504 accepts an NRZ signal from the 82586, performs the NRZ-to-Manchester conversion, and feeds the Manchester signal to the analog front end section. Upon receiving a Manchester signal from the analog front end section, the 82594 converts the signal to an NRZ signal and

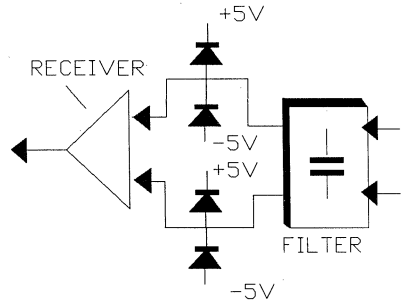


Figure 7.14 Connecting clamping diodes and filter to the receiver of the twisted pair node board.

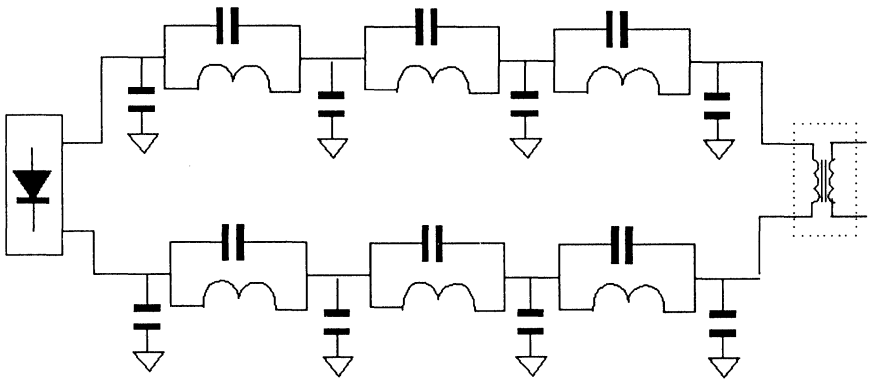


Figure 7.15 The filter topology.

feeds the signal to the 82586. The 82504 has the proper pins to control the analog front end (i.e., the TPEN and TPS pins shown in Fig. 7.12). Connecting the 82504 with the 82586 is almost a straight pin-to-pin connection.

Based on the above components, the Twisted Pair Ethernet node is implemented as shown in Fig. 7.16. The 80186-RAM-82586 connection is the "shared memory" connection discussed in Chap. 6.

7.8 A Twisted Pair Ethernet Serial Supercomponent

Figure 7.17 is identical to Fig. 7.16, except that as shown, we circled the 82504 chip and the whole analog front end section and denoted these sections as 82521. Yes, a component known as the 82521 Twisted Pair Ethernet Serial Super component is available from the Intel Corporation. This item is capable of performing all the tasks denoted

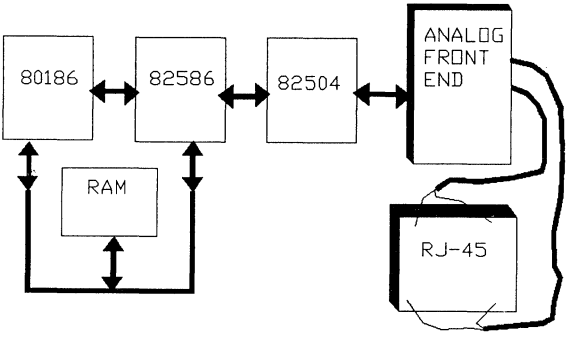


Figure 7.16 Realization of the twisted pair node board with off-the-shelf ICs.

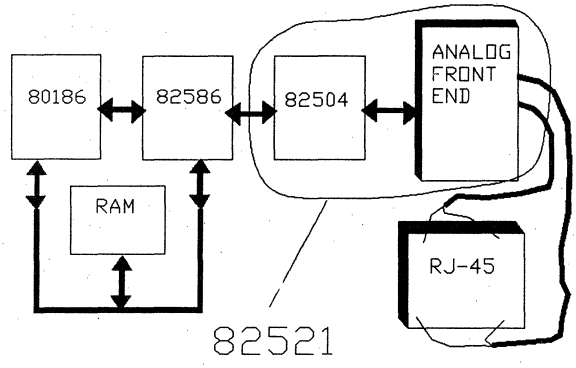


Figure 7.17 Using the 82521 for minimum chip count.

in Fig. 7.18 (including the transformers). Figure 7.19 shows the Twisted Pair Ethernet node when the supercomponent is used. Connecting the 82521 to the 82586 is a straight pin-to-pin connection.

7.9 No Change to the Existing Software of an Ethernet Node

The fact that the very same Ethernet node board (less the transmit and receiver sections) can be utilized in TPE networks is the major attractiveness of the TPE and the multiport repeater concept. That is, the developed Ethernet node software remains as is, regardless of whether it is used in Ethernet, Cheapernet, or Twisted Pair Ethernet networks. The node might have to undergo minor software changes as explained in the following example.

Example 7.8 Figure 6.10 shows an Ethernet/Cheapernet node that is implemented by using the 82586 and the 82501. As discussed, the 82501 is capable of performing the loop-back test (Fig. 6.14).

When implementing a Twisted Pair Ethernet node by using the 82504 (Fig. 7.16) or the 82521 (Fig. 7.19), the section of the 80186 software that implements the loop-back diagnostic has to be removed, since the 82504 and the 82521 are currently not equipped with the loop-back feature.

7.10 The Multiport Repeater (MPR)

The MPR specifications are outlined in the IEEE 802.3c protocols. Figure 7.20 is a block diagram illustrating the tasks performed by the MPR. The multiport repeater board has the following sections:

- *Ports.* The MPR shown in Fig. 7.20 is equipped with two twisted ports and one AUI port. Typically, a multiport repeater has a total of 12 ports, 11 twisted ports, and 1 AUI port.

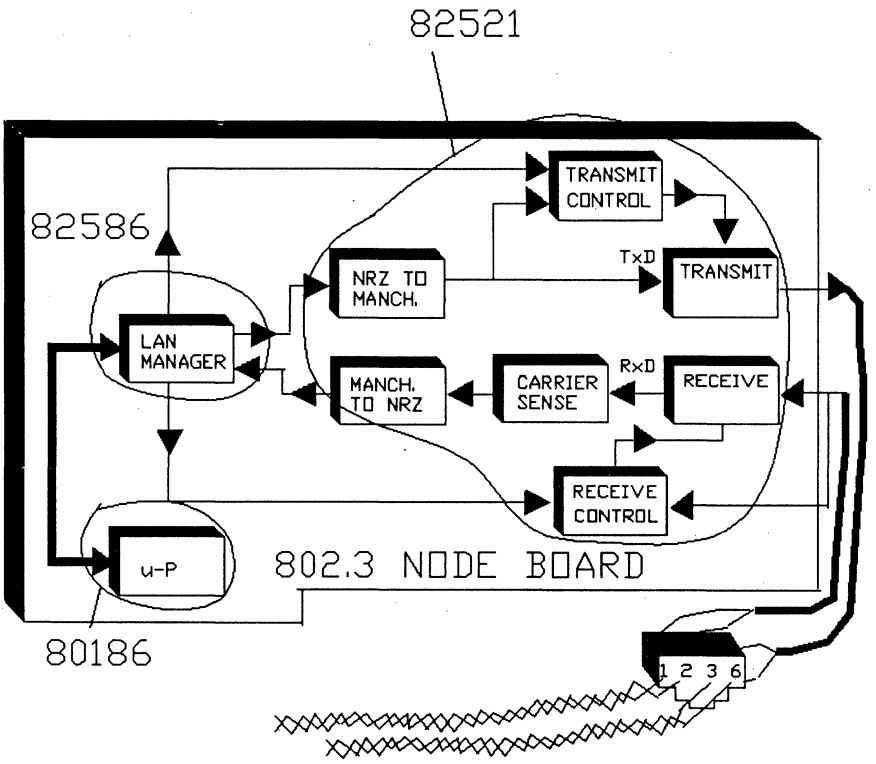


Figure 7.18 Complete realization of the twisted pair node board.

- *Controller.* The controller is responsible for managing all the MPR board tasks.
- *LEDs and LED control section.* The LEDs are included to provide visual indication regarding the status of the MPR board. The controller instructs the LED controller when and which LED should be turned ON and OFF.
- *The Manchester-to-NRZ and NRZ-to-Manchester sections.* These sections perform the Manchester encoding-decoding.
- *Port disable section.* This section is able to disable the twisted ports and the AUI ports upon receiving the appropriate signal from the controller to do so.

7.10.1 Principle of operation of the MPR board

The controller monitors its receive ports, and if it detects that it is receiving a signal from more than one port, it concludes that there is a collision going on.

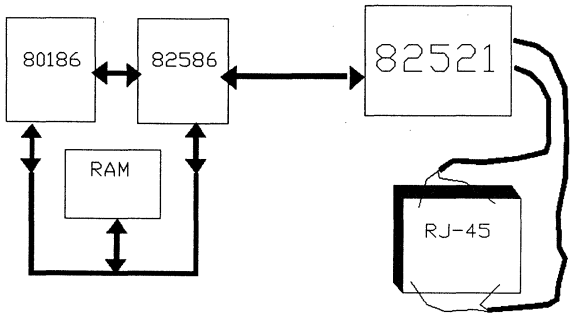


Figure 7.19 Using the 80186-82586 shared memory for the realization of the twisted pair node board.

If however the MPR controller detects that it is receiving a signal from only one port, it transfers the incoming bits to the Manchester-to-NRZ converter, and at the same time generates preamble bits to all the other ports.

The preamble bits are sent to all the other ports, alerting the devices connected to the ports that soon enough they are about to receive a frame. This feature enables the connected devices to prepare their clock recovery circuitries. When the frame eventually arrives at these devices, their clock recovery circuits are already trained.

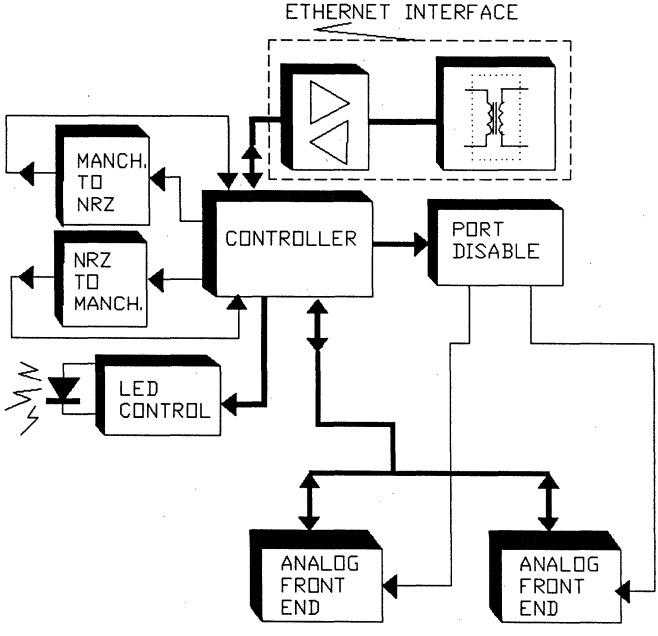


Figure 7.20 The multiport repeater (MPR) block diagram.

The incoming Manchester signal that was routed to the Manchester-to-NRZ converter returns as an NRZ signal to the controller. These NRZ bits are stored in the buffer of the controller.

Eventually, the buffer of the controller is filled up. The MPR controller stops the generation of the preamble bits and transfers the NRZ bits to the NRZ-to-Manchester converter. The output of the NRZ-to-Manchester is a Manchester signal that is returned back to the controller. The MPR controller transmits the Manchester signal to all the devices that are connected to its ports.

If the controller senses a collision during the transmission, it stops the transmission of the frame and instead starts transmitting the jam pattern, making sure that everybody else in the network detects the collision.

While the MPR controller transmits the jam pattern, it keeps monitoring the incoming bits from the devices connected to its ports. It continues transmitting the jam pattern until it discovers that there is no more collision in the network. Once the MPR controller detects that there is no more collision in the network, it stops transmitting the jam pattern.

Another function performed by the controller is the execution of the jabbering procedure, disabling a node that transmits for too long. As shown in Fig. 7.20, the controller sends a signal to the port disable circuit, informing it which port is to be disabled. Thus, a faulty node can-

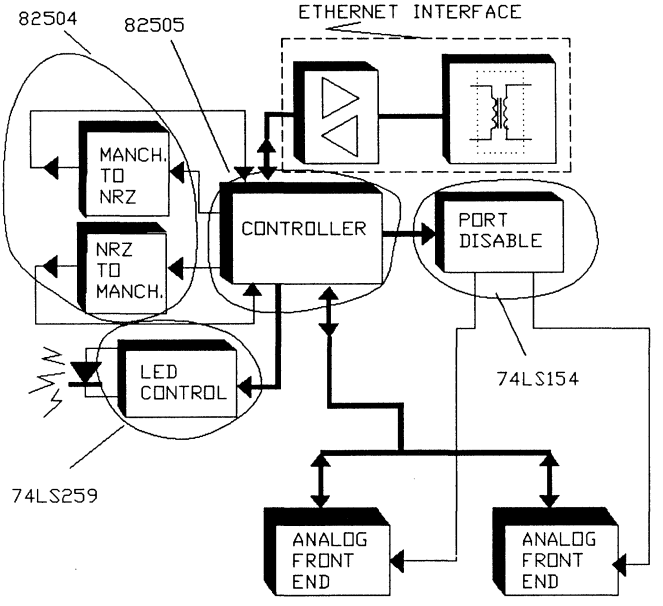


Figure 7.21 Realization of the multiport repeater with off-the-shelf ICs.

not cause the failure of the whole network. The controller keeps monitoring the faulty port, and if at a later time it discovers that the port is fixed (not transmitting any more), it instructs the port disable section to enable that port.

7.11 Implementing a Multiport Repeater Board

The multiport repeater board of Fig. 7.20 is shown again in Fig. 7.21, indicating the components utilized for the implementation.

The controller is the Intel 82505. This chip accomplishes all the required tasks, and its connection to the various sections is almost a straight pin-to-pin connection.

The Manchester encoding-decoding is implemented by using the 82504 that was discussed previously (Fig. 7.10). Of course, the 82504 of Fig. 7.21 is utilized merely for accomplishing the Manchester encoding-decoding, while in Fig. 7.10, the 82504 is utilized to perform the Manchester encoding/decoding and to provide the serial interface.

The LED controller is implemented by using two 74LS259 8-bit addressable latches. A total of 16 LEDs are used.

The 74LS154 4-to-16 decoder is used for the implementation of the port disable section.

7.12 Implementing the Twisted Pair MAU

As shown in Fig. 7.2, an external MAU board is required to connect a "regular" Ethernet node to the multiport repeater board. This MAU

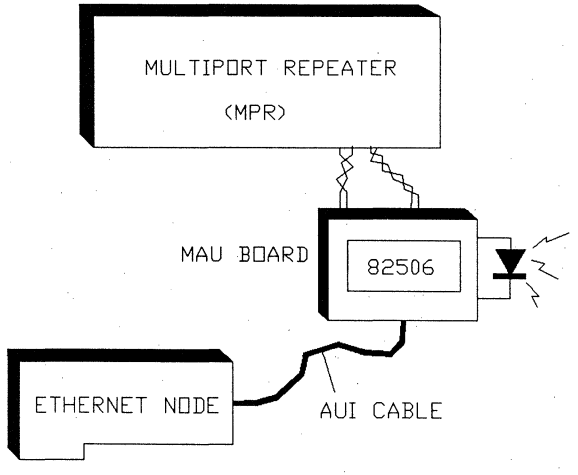


Figure 7.22 Using the 82506.

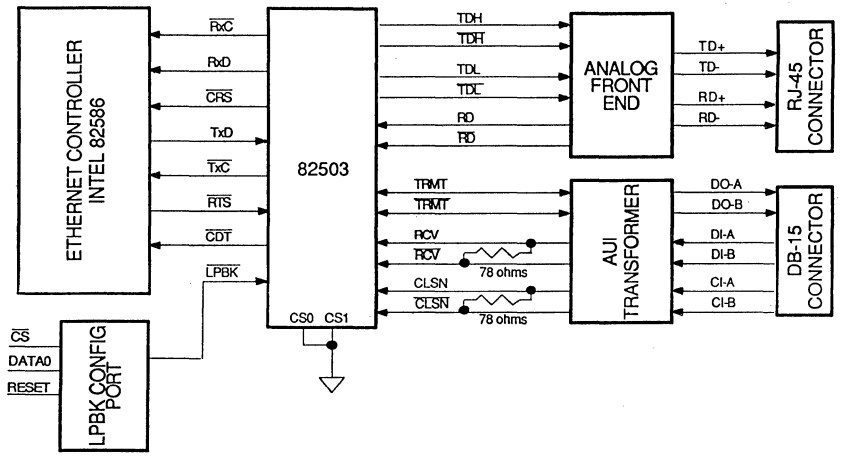


Figure 7.23 Interfacing 82503 to Intel 82586 Ethernet LAN controller. This figure demonstrates how the 82503 is utilized as an interface between the 82586 and a 10BASE-T cable (through the RJ-45 connector). (*Courtesy of Intel Corporation.*)

board must have a "regular" Ethernet AUI cable interface on one side and a twisted pair interface on the other side.

This board is conveniently implemented by using the Intel 82506TB chip. This chip is especially designed for this application, and it has three ports, the AUI cable port, the twisted port, and a LED port to provide visual indication (Fig. 7.22).

7.13 The 82503

This section discusses another important chip, the Intel 82503. The 82503 is a newly announced chip that is used in the implementation of

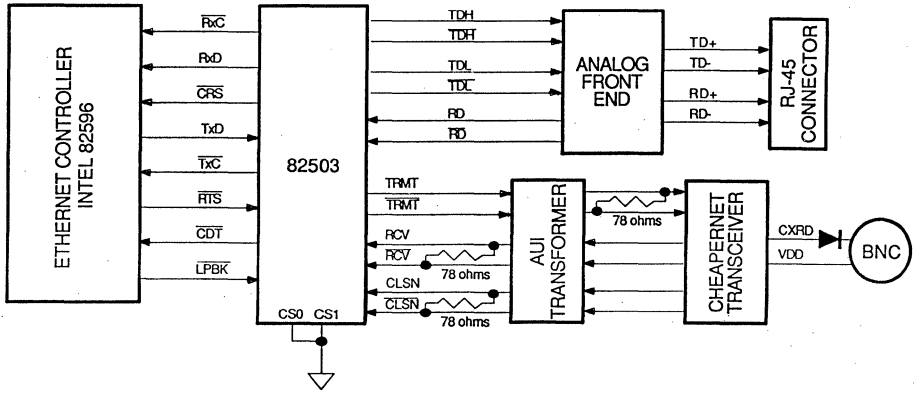


Figure 7.24 Interfacing 82503 to 10BASE-T and Cheapernet. This figure demonstrates the interface between the 82586 and a Cheapernet cable (through the BNC connector). (*Courtesy of Intel Corporation.*)

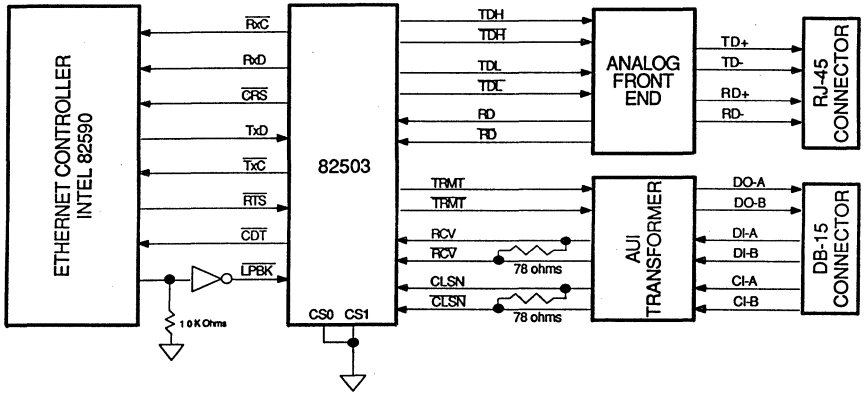


Figure 7.25 Interfacing 82503 to Intel 82590 Ethernet LAN controllers. This figure demonstrates the use of the 82503 as an interface between the Intel 82590 controller and both the 10BASE-T connector (through the RJ-45 connector) and an Ethernet cable (through the DB-15 connector). (*Courtesy of Intel Corporation.*)

10BASE-T nodes. To understand the importance of this device, it is important to understand the "polarity dilemma" of the 10BASE-T protocol.

As previously discussed, the 10BASE-T protocol specifies the usage of unshielded "regular" telephone wires. Altogether, there are four wires connecting one node to another node. As can be seen in Fig. 7.12, the polarity of the wires is important. That is, one pair of wires is designated as the transmitting pair of wires, and the other pair is designated as the receiving pair of wires. Morever, one of the wires in each pair is the "plus" wire, and the other wire is the "negative" wire. This polarity must be preserved since the signal which is being transmitted is a differential signal that is converted to a single-ended signal (see, for example, Fig. 3.9 and the corresponding discussion in Chap. 3).

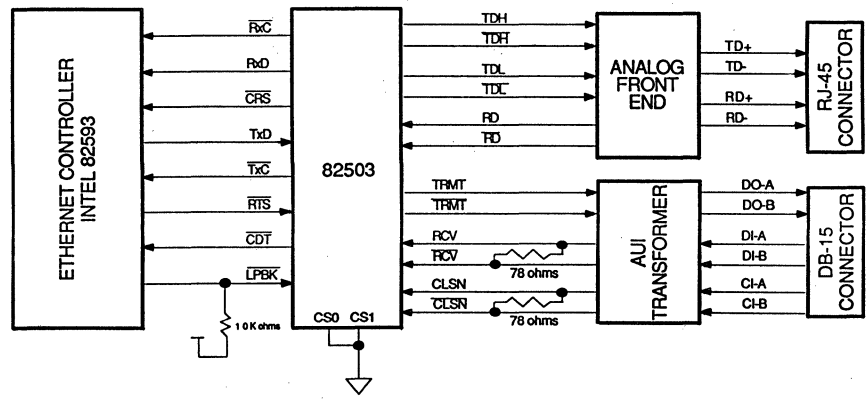


Figure 7.26 Interfacing 82503 to Intel 82593 Ethernet LAN controllers. (*Courtesy of Intel Corporation.*)

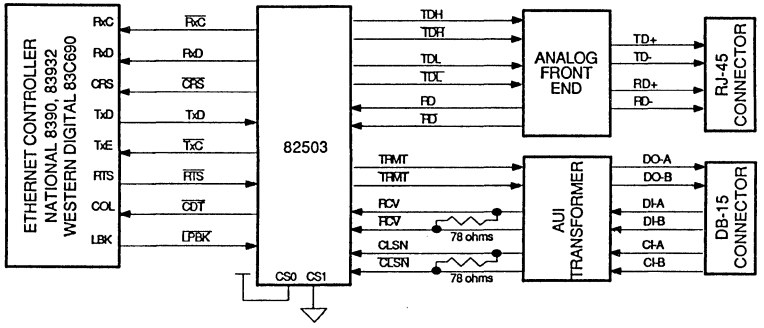


Figure 7.27 Interfacing to National 8390, 83932 and Western Digital 83C690 Ethernet LAN controllers. This figure demonstrates the vast flexibility of the 82503; that is, it may be used with non-Intel LAN controllers as well. (Note that the connections of pins CS0 and CS1 of the 82503 dictate which mode is used.) (*Courtesy of Intel Corporation.*)

Obviously, connecting the wires in reverse order will cause improper operation of the node. Reversing the polarity of the wires is a very common error that occurs even under laboratory conditions (during development) when a node is placed only a few feet from the other node. As you can imagine, reversing wires becomes even more of a problem when the nodes are placed on different floors in a building. Unfortunately, the original 10BASE-T protocol has no provisions that address this issue. Fortunately, the Intel 82503 does address this issue. In particular, the chip detects the fact that the pair of wires were connected to it with reversed polarity. Moreover, the chip automatically corrects the connection (by switching the wires electronically

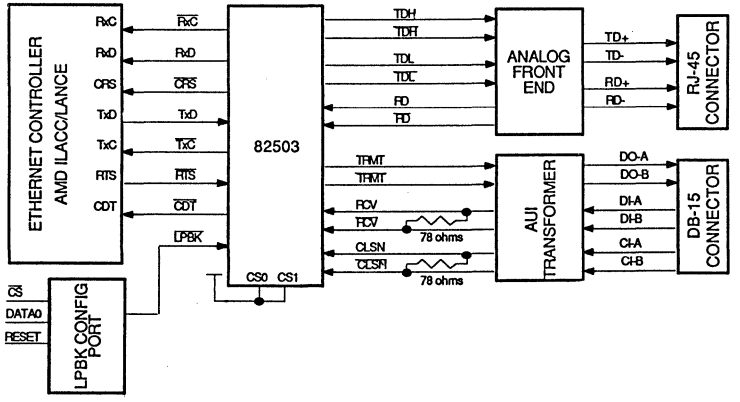


Figure 7.28 Interfacing 82503 to AMD 79C900 (ILACC) and AMD 7990 (LANCE) Ethernet LAN Controllers. This figure also demonstrates the vast flexibility of the 82503; that is, it may be used with non-Intel LAN controllers as well. (Note that the connections of pins CS0 and CS1 of the 82503 dictate which mode is used.) (*Courtesy of Intel Corporation.*)

inside the chip). These functions of the 82503 are referred to as *polarity detection* and *polarity correction.* Thus, even if the wires are reversed, there is no need to physically reverse the wires, since the 82503 will do it by itself without loss of performance.

Another important feature of the 82503 is its ability to serve as the serial interface to a 10BASE-T network and to an Ethernet/Cheapernet network. Figure 7.23 shows how easy and straightforward the interface between the 82586 and the network cable is. The network cable is shown to be either a 10BASE-T cable (through the RJ-45 connector) or an Ethernet cable (through the DB-15 connector). This flexibility of the 82503 enables us to use it as an adapter between 10BASE-T cables and Ethernet cables without changing the driver software of the node (Figs. 7.24 through 7.28). Recall that until the 10BASE-T was announced, the Ethernet was (and still is) a very popular LAN implementation, and there are millions of Ethernet systems already in existence. An 82503 implementation enables the user to be able to incorporate both types of LANs in the very same system at minimum cost.

7.14 Summary

This chapter discusses the 10BASE-T LAN, which is also known as the Twisted Pair Ethernet LAN. This LAN requires multiport repeater boards and node boards that are connected to the multiport repeater via twisted pairs of unshielded wires. The maximum length of the twisted pairs that is allowed to connect a node to a repeater is 100 m. The data rate of the 10BASE-T network is 10 Mbit/s. Thus, the 10BASE-T LAN offers the high data rate of Ethernet/Cheapernet as well as the inexpensive wires (and topology) of Starlan.

Chapter

8

Serial Communication

8.1 Connecting LANs with WANs

The local area network is designed to communicate data within a local geographical area. There is however a need sometimes to communicate data over distances beyond the local area. In such cases, the network is called *wide area network,* or WAN for short. In order to communicate over such great distances, the existing telephone network is being used. Figure 8.1 illustrates the concept of connecting several LANs via the telephone network. Note that the data might be transmitted via a range of complicated telephone processing equipment such as central offices, satellite equipment, and telephone poles.

Example 8.1 What are the advantages and disadvantages of utilizing the telephone network for establishing communication between two remote locations?

Solution One obvious advantage is the fact that the communication link already exists and is readily available.

One disadvantage is the fact that each time a communication session is established, the phone company views it as making a regular phone call, and charges the user.

The main disadvantage is the fact that the telephone system is not designed to transmit data; it is designed primarily for the purpose of transmitting voice. As we shall soon see, voice transmission is equivalent to the transmission of data (1s and 0s) at a data rate of 64,000 bits per second (64 kbit/s). Recall that the Ethernet is able to transmit data at 10,000,000 bits per second (10 Mbit/s). Of course this makes the communication sessions over the telephone system much longer. As an example, the amount of time it takes to transmit a 1-Mbyte file over the Ethernet cable is

$$\frac{1{,}000{,}000 \times 8 \text{ bits}}{10{,}000{,}000 \text{ bit/s}} = 0.8 \text{ s}$$

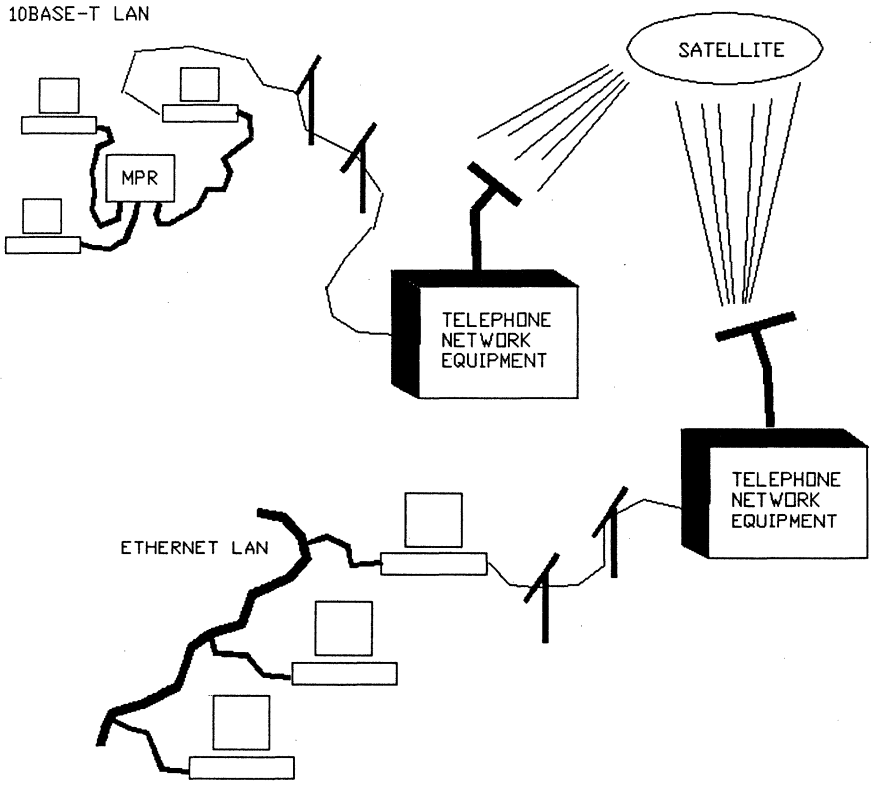


Figure 8.1 Connecting several LANs via the telephone network.

The amount of time it takes to transmit the same file at a rate of 64 kbit/s is

$$\frac{1{,}000{,}000 \times 8 \text{ bits}}{64{,}000 \text{ bit/s}} = 125 \text{ s} = 2.08 \text{ min}$$

8.2 Serial Communication Circuitry

This chapter deals with a very basic communication circuit, a circuit that enables transmitting and receiving data over the telephone network. As we shall see, such a circuit requires only a few components. The design illustrated in this chapter is based on Intel's USART chip, the 8251A. Figure 8.2 is a block diagram describing the principle of operation of the serial communication circuit. Circuit A and circuit B each contain two main components, a microprocessor and a USART device. Assume that circuit A is sending a byte to circuit B. The microprocessor of circuit A transfers the byte to its USART in a parallel form (8 bits simultaneously). The USART accepts the byte, converts it to a

sequence of bits, and then transmits the data in a serial form (sequence of bits, bit after bit) to modem A. Modem A accepts the serial data, converts the serial digital data to analog signal, and transmits the analog signal to the telephone line.

The analog signal travels through the telephone network and eventually reaches modem B. Modem B accepts the incoming analog signal and converts it to a serial digital signal (a sequence of 1s and 0s). The modem then transfers the digital serial data to its USART. The USART accepts the incoming bits, converts them to a byte, and transfers the byte to its microprocessor.

In the same manner, circuit B is able to send bytes to circuit A.

8.3 Modems

As shown in Fig. 8.2, the modem accepts a serial digital signal and converts it to an analog signal compatible with the telephone network. This function of the modem is called *modulation.*

The other function of the modem is to accept the analog signal from the telephone network and to convert it to a serial digital signal. This function is called *demodulation* (hence the name *modem*).

The modem is designed to output data at the same rate at which it receives data. That is, the modem outputs the analog signal to the telephone lines at the same frequency of the incoming bits from its USART. Similarly, the modem outputs the bits to its USART at the same frequency of the incoming analog signal from the telephone lines.

Most modems are capable of performing full-duplex transmission. In a full-duplex transmission the modem receives and transmits simultaneously.

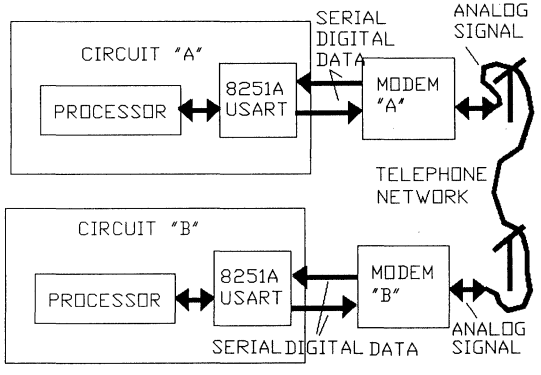


Figure 8.2 Serial communication over the telephone network.

8.4 USART

USART is an abbreviation of *universal synchronous/asynchronous receiver/transmitter.* USART is pronounced "yoos-art." The USART is universal in the sense that it can be programmed to implement a variety of communication protocols.

USART is a device that accepts a data byte and converts it to a serial data (a sequence of 1s and 0s). In addition, the USART accepts serial data and converts it to bytes. USART design is the subject of this chapter. As we shall see, the USART is not just a simple parallel-to-serial and parallel-to-serial converter, but rather a sophisticated device capable of implementing complex communication tasks.

The Intel's USART, the 8251A, is capable of receiving and transmitting serial bits simultaneously. This type of operation is called full-duplex operation.

Once the microprocessor transfers a byte to the 8251A, the 8251A clocks the data, bit after bit to the modem. Upon completing the transfer of the bits, the 8251A signals the microprocessor that it is ready to accept a new byte. In a similar manner, upon receiving bits from the modem, the 8251A signals the microprocessor that data were received and are available.

In most applications, the rate at which bits are clocked from the modem into the 8251A is the same as the rate at which bits are clocked out from the 8251A to the modem. This is called the *baud rate.* The maximum allowable baud rate is determined by the modem, and it is much slower than the rate at which bytes are being transferred between the microprocessor and the 8251A. Thus, there is a considerable time between the time the microprocessor writes a byte into the 8251A and the time that the 8251A signals the microprocessor that it is ready to receive the next byte from the microprocessor. Even during a full-duplex operation, the microprocessor spends a small amount of time transferring and fetching bytes to and from the 8251A.

Prior to the announcement of the 8251A, the chip used in the design of such circuits was the Intel 8251. The 8251A is an upgraded version of the 8251. The 8251A maintains compatibilities with the old 8251, but it includes some extra features.

8.4.1 Communication protocols

To establish communication between circuit A and circuit B (Fig. 8.2), users at both circuits must agree in advance on the communication protocol that will be used during the communication session. The protocol defines the format of the data being transmitted and the data rate. There are several communication protocols practiced in the industry which are all supported by the 8251A.

8.4.2 Other applications where the 8251A is utilized

In addition to serving the purpose of communicating data via the telephone network, the 8251A chip is serving other communication functions such as connecting a CRT to a system.

8.4.3 The 8251A as a peripheral device

As shown in circuits A and B, each of the circuits contains a microprocessor and an 8251A device. There are a few suitable microprocessors to be used with the 8251A. Such processors include

- Processors from Intel's MCS-48 family of processors.
- Processors from Intel's MCS-80 family of processors.
- Processors from Intel's MCS-85 family of processors.
- Processors from Intel's iAPX-86 and 88 family of processors.

8.5 USART-Processor Interface, Hardware Design

The 8251A is a 28-pin chip. It has a *dual-in-line* (DIP) package. The pinout of the chip is shown in Fig. 8.3.

Figure 8.4 shows the internal logic structure of the 8251A. The 8251A is divided to seven internal blocks:

1. Data buffer block
2. Control logic block
3. Transmit buffer block
4. Transmit control block

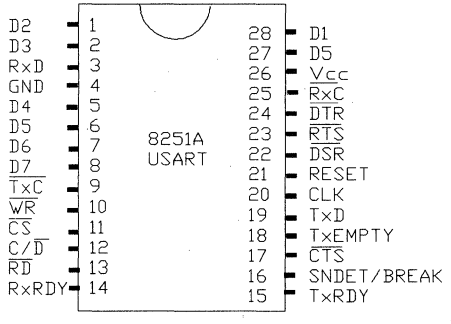


Figure 8.3 The 8251A USART.

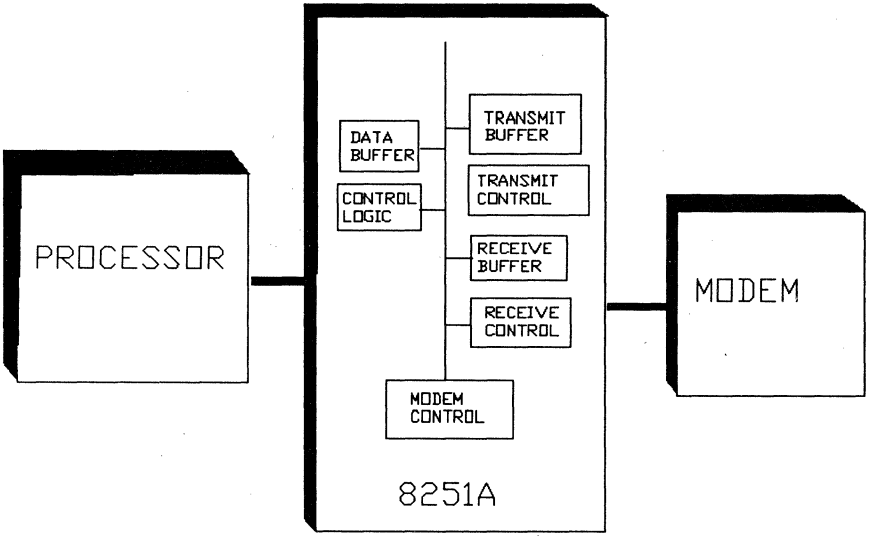


Figure 8.4 8251A based system.

5. Receive buffer block
6. Receive control block
7. Modem control block

Note that Fig. 8.4 shows the 8251A positioned between the processor and the modem. It is the function of the 8251A to interface between the processor and the modem.

8.6 The Data Buffer Block of the 8251A

Figure 8.5 illustrates the connection between the processor and the data buffer block of the USART. The data buffer block has eight external pins:

1. The D0 bidirectional pin
2. The D1 bidirectional pin
3. The D2 bidirectional pin
4. The D3 bidirectional pin
5. The D4 bidirectional pin
6. The D5 bidirectional pin
7. The D6 bidirectional pin
8. The D7 bidirectional pin

The D0 to D7 pins of the USART are connected directly to the D0 to D7 pins of the processor.

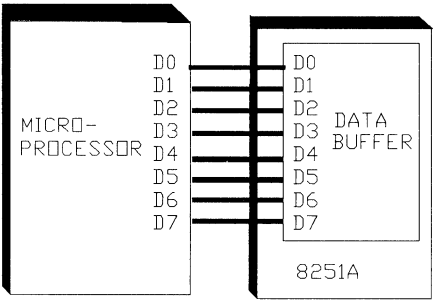


Figure 8.5 The data bus.

There are four types of bytes transferred on the data bus (D0 to D7):

Type 1: Characters to be transmitted

Type 2: Received characters

Type 3: Instruction bytes

Type 4: Status bytes

8.6.1 Type 1, characters to be transmitted

The processor writes bytes to the USART via the D0 to D7 data bus. The USART serializes these bytes and transmits them to the modem bit after bit. We refer to these bytes as the character to be transmitted (i.e., the actual information to be transmitted).

8.6.2 Type 2, received characters

The USART accepts a sequence of bits from the modem, assembles the bits to form a byte, and transfers the byte to the processor via the D0 to D7 data bus. We refer to these bytes as the received characters.

8.6.3 Type 3, instruction bytes

The processor has to send instructions to the 8251A, instructing it to perform various operations (e.g., start the transmission process, type of communication protocol to be implemented, etc.). These instructions are transferred to the USART via the D0 to D7 data bus. We refer to these bytes as the instruction bytes.

8.6.4 Type 4, status bytes

Whenever the USART receives bits from the modem, it examines the incoming bits and decides whether the bits are in accordance with the defined communication protocol being used. If the USART finds a pro-

tocol violation, it sets error flags in its internal register. In addition, the USART updates its internal register regarding the status of the communication process (e.g., a complete character received, etc.). This status information is read by the processor via the D0 to D7 data bus. We refer to these bytes as the status bytes.

8.7 The Control Logic Block of the USART

The next block to be considered is the control logic block shown in Fig. 8.6. The control logic block of the 8251A has six external pins:

1. The RESET input pin
2. The WR input pin
3. The RD input pin
4. The CLK input pin
5. The C/D input pin
6. The CS input pin

8.7.1 The RESET pin of the 8251A

One of the signals that the processor sends to the control logic block is the reset signal. As shown in Fig. 8.6, the reset pin of the 8251A is directly connected to the P0 pin of the processor. We assume here that the processor uses one of its latched output ports to assert the reset of the USART.

Whenever the reset pin of the USART is asserted, the USART enters the idle state. During the idle state, the USART does not send any characters to the modem and does not accept any characters from the modem.

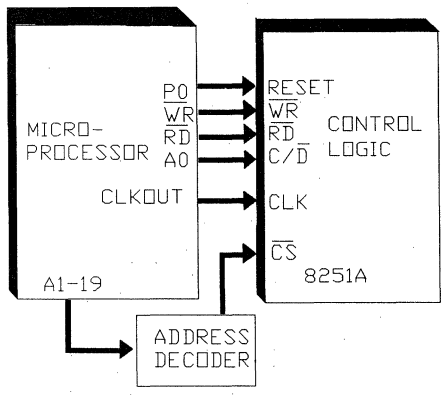


Figure 8.6 Control logic circuitry.

The USART stays in the idle mode until the processor writes an instruction to the USART, instructing it to get out of the reset (idle) state into another state.

To assert the reset pin, the processor should apply a 1 to the pin. This 1 should be applied for a minimum time of T(reset).

The reset operation described above is a hardware reset. Alternatively, the processor may instruct the USART to enter the reset (idle) state by sending an instruction byte to the USART via the D0 to D7 data bus.

8.7.2 The WR (write) pin of the 8251A

Another signal being supplied to the control logic of the 8251A is the WR (write) signal. This pin is shown in Fig. 8.6 to be connected directly to the WR (write) pin of the processor. Whenever the processor writes to the 8251A (or to any other device), the processor asserts this pin. The 8251A samples its WR pin, and when sensing 0 on the pin, it realizes that the processor is attempting to write bytes into its data buffer.

8.7.3 The RD (read) pin of the 8251A

Another signal being supplied to the control logic block of the 8251A is the RD (read) signal. This signal is shown in Fig. 8.6 to be directly connected to the RD (read) pin of the processor. Whenever the processor reads bytes from the 8251A (or from any other device), the processor asserts this pin. The 8251A samples its RD pin, and when sensing a 0 on the pin, it realizes that the processor is attempting to read bytes from its data buffer.

8.7.4 The CLK (clock) pin of the 8251A

Another signal being supplied to the control logic block of the 8251A is the CLK (clock) signal. This pin is shown in Fig. 8.6 to be directly connected to the CLKOUT pin of the processor. Most processors have a CLKOUT pin (or an equivalent pin) which generates a clock to be used by external devices. This pin of the processor is usually designated as the phase 2 clock pin, or CLKOUT pin.

We shall soon learn that there are two additional clocks to be supplied to the 8251A. One of these additional clocks determines the rate at which bits are being outputted from the 8251A to the modem. The other clock determines the rate at which bits are being supplied to the 8251A from the modem. We mention these two additional clocks now, because it is important to realize that these other two clocks have nothing to do with this CLK pin.

All operations in the 8251A are being performed in accordance with the CLK clock. The 8251A needs the CLK clock to transfer data

among its internal blocks, to decode instructions, and to perform other operations.

Example 8.2 The minimum time that the 8251A has to stay in the reset state is called T(reset) and is equal to 6 CLK cycles.

8.7.5 The C/D (control/data) pin of the 8251A

Another signal being supplied to the control logic block of the 8251A is the C/D (command/data) signal. This signal is shown in Fig. 8.6 to be directly connected to the A0 pin of the processor.

The 8251A must know whether the processor is attempting to write character bytes or instruction bytes. The 8251A must also know whether the processor is attempting to read received characters or status bytes.

Whenever the processor writes a character byte, it sends a 0 to the C/D pin of the 8251A. Whenever the processor writes an instruction byte, it sends a 1 to the C/D pin of the 8251A.

Similarly, whenever the processor reads received characters from the 8251A, it sends a 0 to the C/D pin of the 8251A. Whenever the processor reads status bytes from the 8251A, it sends a 1 to the C/D pin of the 8251A. The reason for connecting the A0 pin of the processor to the C/D pin of the 8251A will be explained soon.

8.7.6 The CS (chip select) pin of the 8251A

Another signal being supplied to the control logic block of the 8251A is the CS (chip select) signal. This signal is shown in Fig. 8.6 to be connected to the output of the address decoder.

Whenever the processor wants to write a byte into the 8251A, or to read a byte from the 8251A, it must supply a 0 to the CS pin of the USART. If the CS pin of the 8251A is not asserted, the 8251A floats its D0 to D7 pins.

Example 8.3 Assume that the processor has 20 address lines. The address decoder may be designed to decode the following address:

```
A19                                   A0
 0000    0001    0010    0011    000x

   0       1       2       3     0 or 1
```

Thus, the USART is decoded to address 01230(hex) and 01231 (hex). That is, the CS pin of the 8251A is asserted whenever the processor reads or writes the address 01230(hex). The CS pin of the 8251A is also asserted whenever the processor reads or writes the address 01231 (hex).

As shown in Fig. 8.6, the A0 pin of the processor is directly connected to the C/D pin of the 8251A, and thus, the A0 is used to determine the type of byte that is being read or written.

The program of the processor should be written for the above decoding example in the following manner:

- Write character to be transmitted to address 01230(hex)
- Read received characters from address 01230(hex)
- Write instructions to the 8251A to address 01231(hex)
- Read status bytes from the 8251A from address 01231(hex)

8.8 Protocols

8.8.1 Serial communication

Serial communication is a form of transmitting data in which the individual bits of data are being transmitted sequentially one bit after the other.

The 8251A follows the common practice of transmitting the LSB (least significant bit) first.

Example 8.4 The byte 4F (hex) is to be transmitted serially. The byte 4F (hex) is written in binary form as 0100 1111. Thus, the 8251A outputs the LSB bit (the 1) as the first bit.

8.8.2 Synchronous-asynchronous communication

There are two categories of serial communication protocols: (1) synchronous protocols and (2) asynchronous protocols. The 8251A is capable of supporting all the standard serial communication protocols used in various industries. Before attempting to use the 8251A in our design, we must understand the serial protocols that the chip supports.

The 8251A must know in advance which type of transmission protocol is being used by the modem, synchronous transmission or asynchronous transmission.

8.8.3 Synchronous transmission

In synchronous transmission, characters are being sent continuously (string of characters). There are no bits separating the characters. The format of data in synchronous transmission is shown in Fig. 8.7.

While there are no bits separating the characters in synchronous transmission, the 8251A needs some type of indication as to where is the starting point of the incoming string of characters. Most protocols insert a character called the SYNC character at the beginning of the string. Thus, whenever the 8251A detects the SYNC character, it realizes that a string of characters follows.

While most protocols insert one SYNC character in front of the string, one particular protocol, the IBM bi-sync protocol, requires that two consecutive SYNC characters are inserted in front of the string.

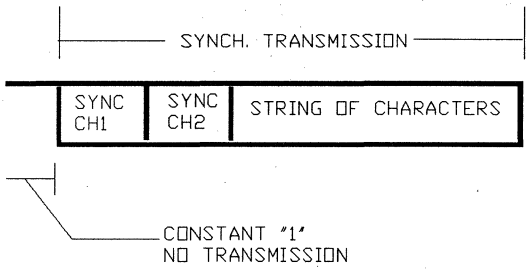


Figure 8.7 Synchronous transmission.

Figure 8.7 shows two SYNC characters preceding a string of characters (i.e., the figure shows the IBM bi-sync synchronous protocol). For a single SYNC character protocol, the format is the same except that there is a single SYNC character preceding the string of characters.

Whenever the 8251A does not have any character to transmit, it transmits a continuous string of 1s.

8.8.4 Asynchronous transmission

In asynchronous transmission, data are being sent character after character. The receiver has to recognize which bit is the first bit of the character and which bit is the last bit of the character. In asynchronous transmission, each character is separated from the following character by special bits that enable the receiver to determine the boundaries of each character.

Example 8.5 The format of the data in asynchronous transmission is shown in Fig. 8.8.

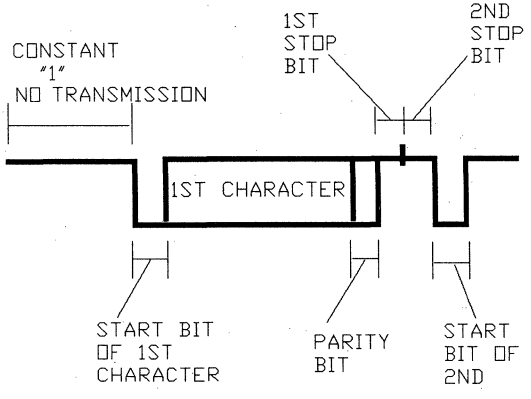


Figure 8.8 Asynchronous transmission.

Whenever there is no transmission of data, the receiver detects a continuous 1. The receiver recognizes the starting point of a character by recognizing the start bit. The start bit is defined as a 0 bit.

Next in the sequence of bits are the bits of the character itself. Asynchronous protocols may consist of 8, 7, 6, or 5 bits per character.

The parity bit Following the bits of the character itself is the parity bit. The parity bit may be an even parity bit, an odd parity bit, or no parity bit at all.

The stop bits The last section of the data consists of the stop bits. A stop bit is always a 1, and it is an indication that this is the end of the data block. Different protocols require different stop bit durations.

Example 8.6 The duration of the stop bit may be 1, 2, or 1½ bit time.

In a protocol that requires a 1½ stop bit, the duration of the first stop bit is one bit time, and the duration of the second stop bit is half the duration of the first stop bit.

Following the stop bits is either a continuous 1, indicating that there is no transmission after the last stop bit, or a 0, which is the start bit of the next character.

The break character Most asynchronous protocols use the concept of the *break* character. A break character is all 0 bits.

Example 8.7 Suppose that the asynchronous protocol used is

- Five bits per character
- Even parity bit
- Two stop bits

A character with

D0 = 1

D1 = 1

D2 = 0

D3 = 1

D4 = 1

The waveform of the transmission signal under the above protocol is shown in Fig. 8.9.

- The data shown start with no transmission (a continuous 1).
- The next bit is a 0, the start bit.

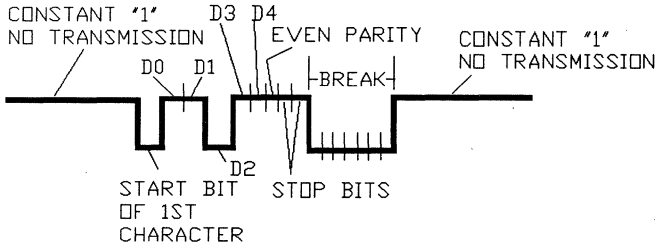


Figure 8.9 Waveform of the transmission signal.

- The next 5 bits are the character itself. In this example, the 5 bits of the character are

$$D0 = 1$$

$$D1 = 1$$

$$D2 = 0$$

$$D3 = 1$$

$$D4 = 1$$

- The next bit is the even parity bit, which should be a 0. However, the receiver received a 1. The 8251A will detect the fact that there is a parity error and will respond by raising a flag known as the parity flag.
- The next 7 bits are all 0s, that is, one start bit, five 0s for the character, one 0 for the even parity.
- Following the parity bit are two stop bits followed by a continuous 1.

The second character is the break character (since it is composed of only 0 bits).

The receiver and the sender have to agree in advance what the break character means. Most asynchronous communication protocols use the break character to attract the immediate attention of the receiver. In asynchronous transmission, the 8251A evaluates each incoming character. Upon detecting the break character, the 8251A raises an internal flag to indicate that a break character was received. This enables the processor to respond immediately.

8.8.5 Protocol parameters

The 8251A is capable of implementing all the above protocols. As can be seen from the above discussion, there are dozens of possible protocol combinations. The parameters of the synchronous protocol are

- Number of SYNC characters (one or two SYNC characters)
- Parity type (even, odd, or no parity)
- Number of bits per character (5, 6, 7, or 8 bits per character)

The parameters of the asynchronous protocol are

- Number of stop bits (1, 2, or 1½ stop bits)
- Parity type (even, odd, or no parity)
- Number of bits per character (5, 6, 7, or 8 bits per character)

Prior to transmitting, the receiver and the sender must agree on the type of protocol to be used.

8.9 Transmitter, Hardware Design

8.9.1 The transmitting process

To transmit a character from the processor to the modem, the following sequence of steps takes place (refer to Fig. 8.10).

Step 0. The processor configures the 8251A to make it work in accordance with a particular communication protocol.

Step 1. The processor sends an instruction to the 8251A, instructing it to set the flag TxEN. The 8251A has an internal transmit enable flag, the TxEN flag, which enables or disables the 8251A from trans-

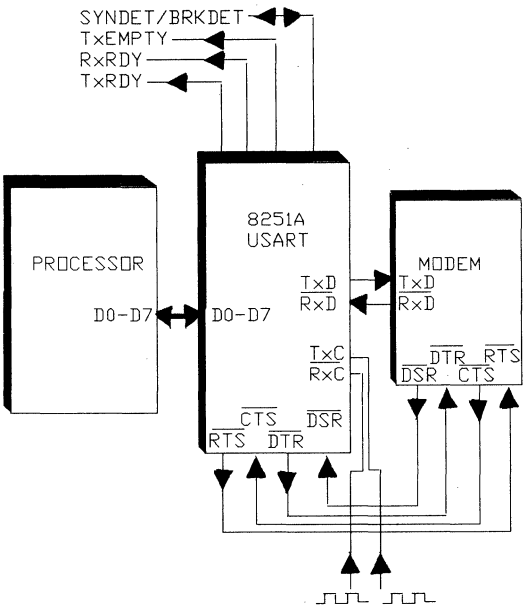


Figure 8.10 Modem-8251A circuitry.

mitting bits to the modem. It is the processor that sets or resets this flag by sending to the 8251A an instruction.

Step 2. The processor sends the character to be transmitted to the data buffer of the 8251A. The 8251A transfers the character from the data buffer to its transmit buffer and prepares the character for transmission. This includes inserting the start bit, the parity bit, the stop bit(s), the SYNC character(s), all in accordance with the configured protocol.

Once the 8251A completes transferring the character from the data buffer to the transmit buffer, the data buffer is empty and ready to receive a new character to be transmitted. The 8251A raises its internal flag, the TxRDY flag, to indicate that the data buffer is empty and ready to accept a new character. The processor has the means to read this flag. If the processor writes a new character to the data buffer, the TxRDY flag is automatically reset.

The 8251A also has an output pin called TxRDY. The 8251A asserts this pin whenever the data buffer is empty. Thus, instead of reading the TxRDY flag via software, it is possible to connect the TxRDY pin to one of the interrupt pins of the processor. In Fig. 8.10 we show the pin not connected, since in our design we read the RxRDY status via the software. For completeness, we should mention that the condition for setting the TxRDY pin is different from the condition for setting the TxRDY internal flag. The TxRDY flag is set whenever the data buffer is empty, while the TxRDY pin is asserted whenever the data buffer is empty and the CTS pin is asserted (the CTS pin is discussed in step 4).

Step 3. Now the modem needs to be alerted that bits are about to be transmitted to it. This is done via the RTS pin. The RTS (ready to send) pin is an output pin of the 8251A. The RTS pin of the 8251A and the RTS pin of the modem are tied together. It is through this pin that the 8251A informs the modem that it has bits to be transmitted to the modem.

The 8251A does not assert the RTS pin automatically. It is the processor that sends an instruction to the 8251A to assert the RTS pin.

Step 4. Once the modem realizes that it is about to receive bits from the 8251A, it could confirm that it is OK to go ahead with the transmission. The modem could also indicate NOT to go with the transmission. This confirmation or denial is accomplished via the CTS pin.

The CTS (clear to send) pin is an input pin of the 8251A. The CTS pin of the modem and the CTS pin of the 8251A are tied together. It is through this pin that the modem informs the 8251A that it is ready to

accept bits. When the modem asserts the CTS pin, the 8251A automatically transmits bits to the modem. The 8251A transmits the bits through the TxD output pin. Whenever the 8251A is not outputting bits onto the TxD pin, the TxD pin is at a continuous 1 level. The transmission rate is determined by the TxC input pin. The TxC pin is called the transmitter clock pin. The 8251A outputs the bits onto the TxD pin on the falling edge of the TxC clock.

When the 8251A completes the transmission of the character to the modem, it sets an internal flag called the TxEMPTY as an indication that it has no more bits to transmit in its internal transmit buffer. The processor may read this flag.

The 8251A also has an output pin called the TxEMPTY pin. Whenever the 8251A raises the TxEMPTY flag, it also asserts the TxEMPTY pin. Thus, it is possible to connect the TxEMPTY pin to one of the interrupt pins of the processor. In Fig. 8.10 we show the pin not connected, since in our design we read the TxEMPTY status via the software.

8.10 Receiver, Hardware Design

8.10.1 The receiving process

To receive a character (from the modem to the processor), the following sequence of steps takes place.

Step 0. The processor configures the 8251A to make it work in accordance with a particular communication protocol.

Step 1. The processor sends an instruction to the 8251A, instructing it to set the flag RxE. The 8251A has an internal receive enable flag, the RxE flag, which enables or disables the 8251A from receiving bits from the modem. It is the processor that sets or resets this flag by sending to the 8251A an instruction.

Step 2. Now the 8251A needs to be alerted that bits are about to be transmitted to it. This is done via the DSR pin.

The DSR (data set ready) pin is an input pin of the 8251A. The DSR pin of the 8251A and the DSR pin of the modem are tried together. The modem informs the 8251A that it has bits to be transmitted to the 8251A through this pin.

The 8251A has an internal flag called the DSR flag. The DSR flag is raised whenever the 8251A senses that its DSR pin is asserted. The processor can read this flag and thus realize that the modem has bits to be transmitted to the 8251A.

Step 3. Once the processor realizes that the modem has bits to send (via reading the DSR flag), it could confirm that it is OK to go ahead with the transmission. The processor could also indicate *not* to go ahead with the transmission. This confirmation or denial is done via the DTR pin.

The DTR pin (data terminal ready) pin is an output pin of the 8251A. The DTR pin of the modem and the DTR pin of the 8251A are tied together. It is through this pin that the 8251A informs the modem that it is ready to accept bits.

The 8251A does not assert the DTR automatically. It is the processor that sends an instruction to the 8251A to assert the DTR pin. Once the DTR pin is asserted, the modem automatically transmits bits to the 8251A.

The modem transmits the bits through the RxD pin. Whenever the modem is not outputting bits onto the RxD pin, the 8251A senses a continuous 1 level on this pin. The transmission rate is determined by the RxC input pin. The RxC pin is called the receiver clock. The 8251A accepts the bits onto the RxD pin on the rising edge of the RxC clock.

Once the 8251A receives the character, it assembles the character. This includes stripping the start bit, the parity bit, and the stop bit(s), all in accordance with the configured protocol.

Once the 8251A completes assembling the character, it sets an internal flag called the RxRDY as an indication that it has completed the receiving of a character. The processor is able to read this flag.

The 8251A also has an output pin called the RxRDY pin. Whenever the 8251A raises the RxRDY flag, it also asserts the RxRDY pin. Thus, it is possible to connect the RxRDY pin to one of the interrupt pins of the processor. In Fig. 8.10 we show the pin not connected, since in our design we read the RxRDY flag via the software.

Step 4. The processor realizes that there is a character ready to be fetched from the 8251A by reading the RxRDY flag. The processor then reads the character from the 8251A.

8.11 Synchronous Modems and Asynchronous Modems

Figure 8.11 shows that there are two clocks supplied to the 8251A, the TxC clock and the RxC clock. We will now show how to connect these clocks.

8.11.1 Synchronous modem

When a synchronous modem is used, the modem dictates the rate at which the 8251A outputs and inputs bits. Figure 8.12 shows that the

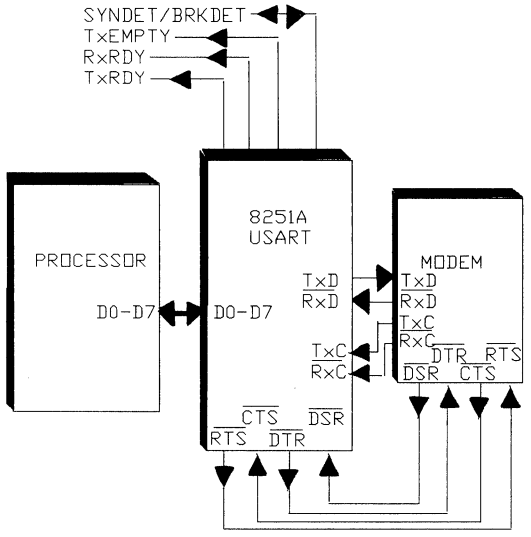


Figure 8.11 Synchronous modem.

TxC and RxC clocks are supplied by the external clock generator. (The external clock generator could be a 555 or a crystal circuit.) Note from the figure that the clock generator supplies the clock to both TxC and RxC. This is applicable where the 8251A has to transmit and receive bits at the same rate.

There are however some applications where the 8251A is receiving bits from one modem at one baud rate and sends bits to another

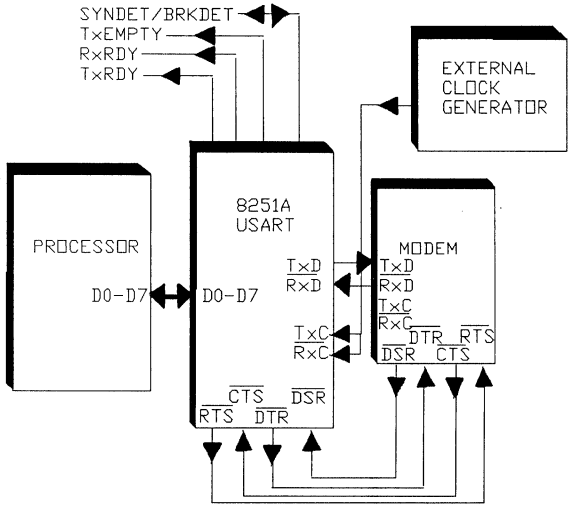


Figure 8.12 Asynchronous modem.

modem at another baud rate. In this case there is a need to use two external generators.

8.11.2 Synchronous mode

When the 8251A is working in the synchronous mode, the baud rate is simply the TxC and RxC frequencies. This is called the 1x baud rate.

8.11.3 Asynchronous mode

When the 8251A is working in the asynchronous mode, the baud rate is the frequency of the external clock generator multiplied by a constant. This constant could be 1, 1/16, or 1/64. We shall see later how the processor instructs the 8251A which constant to use.

8.11.4 The baud rate

If the 8251A is working in the synchronous mode, you do not have to worry about the baud rate, since the baud rate is totally dictated by the modem. The 8251A is able to accommodate any rate dictated by the modem.

If, however, the 8251A is configured to work in the asynchronous mode, you do have to consider the baud rate. As discussed, the baud rate in this case is equal to the frequency of the external clock generator multiplied by a constant. However, you should not set the external clock generator to an arbitrary frequency. The reason is that most modems work at particular baud rates. The most common baud rates used are

110 baud
300 baud
600 baud
1200 baud
2400 baud
4000 baud
9600 baud

Other, higher baud rates exist, but a regular telephone network link is not able to transmit a higher baud rate without severely distorting the signal.

Example 8.8 Suppose that the 8251A is transmitting at a baud rate of 110 baud in asynchronous mode. What should be the frequency of the external clock generator?

Solution If the 8251A is configured with constant = 1x, the frequency of the external clock generator should be

$$1 \times 110 = 110 \text{ Hz}$$

If the 8251A is configured with constant = 16x, the frequency of the external clock generator should be

$$16 \times 110 = 1760 \text{ Hz}$$

If the 8251A is configured with constant = 64x, the frequency of the external clock generator should be

$$64 \times 110 = 7040 \text{ Hz}$$

8.12 The SYNDET/BRKDET Pin of the 8251A

8.12.1 The SYNDET/BRKDET (SYNC DETECT/BREAK DETECT) pin of the 8251A

Note from Figs. 8.10, 8.11, and 8.12 that the SYNDET/BRKDET pin of the 8251A can be configured as either an input or as an output. (The pin is shown with double arrows pointing into and out of the 8251A.)

Upon reset, this pin is configured automatically as an output pin, and it is at 0. After reset, the processor can configure the pin as input or output.

8.12.2 The SYNDET/BRKDET as an output pin in synchronous mode

Assume that the processor instructs the 8251A to be in a synchronous mode. Also assume that the processor instructs the 8251A to configure the SYNDET/BRKDET as an output pin. Under these conditions (synchronous mode and SYNDET/BRKDET is an output pin), the 8251A is working in the so-called internal SYNC DETECT mode.

The 8251A does not accept any incoming characters unless it detects the SYNC character(s). Once the 8251A detects the SYNC character(s), it outputs a 1 to the SYNDET/BRKDET pin. The 8251A also has an internal flag that is being raised automatically whenever these same conditions occur.

Thus, the processor may examine this pin and is able to conclude that the 8251A received the SYNC character(s). In Figs. 8.10, 8.11, and 8.12 we show the pin not connected, since in our design the processor reads the SYNDET/BRKDET flag via the software.

The IBM bi-sync protocol (which the 8251A supports) requires two SYNC characters. In this case, the 8251A searches for two consecutive SYNC characters. The 8251A outputs a 1 on the pin and raises the flag if it detects two consecutive SYNC characters.

8.12.3 The SYNDET/BRKDET as an input pin (only in synchronous mode)

Assume that the processor instructs the 8251A to be in synchronous mode. Also assume that the processor instructs the 8251A to configure the SYNDET/BRKDET as an input pin. Under these conditions (synchronous mode and SYNDET/BRKDET is an input), the 8251A is working in the so-called external SYNC DETECT mode.

Here, an external device supplies a signal to the pin, an indication to the 8251A not to search for the SYNC character(s), but rather to start receiving characters from the modem [even if the SYNC character(s) is not detected]. In most applications where synchronous transmission is used, the internal SYNC DETECT mode is used.

8.12.4 The function of the SYNDET/BRKDET pin in an asynchronous transmission

Assume that the processor instructs the 8251A to work in the asynchronous mode. In this case the SYNDET/BRKDET is automatically configured to be an output pin. Whenever the 8251A detects the break character, it asserts the SYNDET/BRKDET pin.

The processor is able to examine the value of the pin by connecting the pin to one of the input port pins of the processor. Alternatively, the processor may read an internal SYNDET/BRKDET flag of the 8251A that holds the value presented on the pin.

8.12.5 Final hardware notes

So far this chapter has discussed the functions of all the pins of the 8251A and how to connect these pins. For completeness we mention here that the V_{cc} pin of the 8251A has to be connected to a +5 V_{dc}, and the GND pin should be connected to ground.

8.12.6 Summary of the 8251A's hardware

From the discussion, you see that connecting the 8251A to the processor and to the modem is a straightforward pin-to-pin connection. There are occasions where LEDs are connected to some of the pins of the 8251A. These LEDs are used in cases where the user likes to monitor the operation of the 8251A by observing the LEDs. Adding these LEDs proves to be very useful during the development period. For production, many designers prefer to have at least some of these LEDs as a permanent part of the circuit. This enables ease of repairing and enables a repair-service technician to perform visual diagnostics of the circuit.

8.13 Software Implementation

This section deals with the software design of the processor-8251A circuit. The 8251A is capable of performing many complex tasks. It was designed to be interfaced as a peripheral device to a microprocessor. The fact that the 8251A is serving as a peripheral device means that the 8251A does not fetch its instructions from a RAM or ROM, but rather the instructions are being supplied to it by the processor.

Once the 8251A receives the instructions from the processor, it decodes the instructions and executes them with no help from the device that sent to it the instructions.

The processor software consists of the following main sections:

- Configuring the 8251A.
- Sending instructions to the 8251A.
- Monitoring the operation of the 8251A.
- Reading characters from the 8251A. These are characters that the 8251A received from the modem.
- Sending characters to the 8251A. These are characters that the 8251A sends to the modem.

We now examine each of the above sections.

8.13.1 Configuring the 8251A

The 8251A has to be configured to work in accordance with the desired serial communication protocol. As mentioned, there are two categories of protocols, the asynchronous protocol and the synchronous protocol. An integral part of the configuration process is to program the 8251A with the parameters of the protocol (e.g., one or two SYNC characters, number of bits per character, etc.).

We now list all the parameters—eight of them—with which the processor configures the 8251A. The 8251A must be supplied with these eight parameters in order to be able to implement the protocol.

8.13.2 The parameters

Parameter 0: Synchronous or asynchronous protocol. The 8251A must be told whether the protocol to be implemented is a synchronous or asynchronous protocol.

Parameter 1: Number of bits per character. The number of bits per character could be 5, 6, 7, or 8. This parameter is applicable to both synchronous and asynchronous protocols.

The processor sends a character to the 8251A via the 8-bit data bus. If the 8251A is configured to consider a character with fewer than 8

bits per character, the 8251A ignores the unused bits. The most significant bits are the ones being ignored.

Example 8.9 Suppose that the 8251A is configured to work with 5 bits per character. The processor sends the following byte to the 8251A:

```
  A    8
1010 1000
```

The 8251A ignores the three most significant bits and actually sends the following sequence of bits to the modem:

```
0 1000
```

When the processor fetches a received character from the 8251A, the 8251A outputs 8 bits on the data bus regardless of the number of bits per character. The processor software (knowing the configuration) extracts the useful bits.

Example 8.10 Suppose that the 8251A is configured to work with 5 bits per character. The processor reads the following byte from the 8251A:

```
  0    8
0000 1000
```

The processor ignores the three most significant bits and understands that the received sequence of bits from the modem is

```
0 1000
```

The 8251A sends 0s for each unused bit.

Parameter 2: Parity bit implementation. The 8251A has to be told whether a parity bit is to be implemented in the protocol. This parameter is applicable to both synchronous and asynchronous protocols.

If the 8251A is configured to implement the parity bit, it examines the parity bit for each incoming character from the modem. If the 8251A finds that indeed there is a parity error, it sets the parity error flag. In addition, if the 8251A is configured to implement the parity bit, it automatically inserts the parity bit in the characters that it is sending to the modem.

Note that the parity implementation is handled completely by the 8251A. The processor sends characters to the 8251A without the parity bit. Similarly, when the processor reads a received character from the 8251A, the 8251A supplies the character without the parity bit.

It is important to realize that when configuring the number of bits per character, the parity bit is not included in the count.

Example 8.11 Suppose that a protocol supports

- A parity bit.
- The fact that each character is represented by 7 bits.

When specifying number of bits per character, specify 7 bits per character (not 8 bits per character).

Parameter 4: Number of stop bits. This parameter is applicable to the asynchronous protocol only. That is, stop bits are implemented in asynchronous protocol but are not part of the synchronous protocol.

The 8251A has to be told the number of stop bits. There can be 1, 2, or 1½ stop bits. The asynchronous protocol must use at least one stop bit. Thus, it is invalid to configure the 8251A to work with asynchronous protocol and no stop bit.

When the 8251A encounters the stop bit(s) in the receiving character from the modem, it knows that this is the end of the character. Part of the 8251A's job in processing the incoming characters is to strip off the stop bits.

When transmitting characters to the modem, the 8251A inserts the stop bit(s) automatically.

Parameter 5: Baud rate. This parameter is applicable only to the asynchronous protocol. Recall that the baud rate is determined by multiplying the frequency of the TxC clock by a certain constant. This constant can be 1, 1/16, or 1/64. This parameter configures the 8251A to use one of these constants.

This parameter is not used in the synchronous protocol since the baud frequency in the synchronous protocol is always the same frequency of the TxC clock.

Parameter 6: External or internal SYNC DETECT. This parameter is applicable only to the synchronous protocol. Recall that in a synchronous transmission, the modem sends to the 8251A a string of characters preceded by one or two SYNC characters.

The 8251A examines each of the incoming characters and searches for the SYNC character(s). If the 8251A discovers that the first character is not the SYNC character, it ignores the character. Eventually, the 8251A detects the SYNC character(s), and processes the characters that follow the SYNC character(s).

In order to force the 8251A to search for the SYNC character(s), the processor must configure the 8251A to be in the internal SYNC DETECT mode. In this mode, the 8251A automatically configures the pin SYNDET/BRKDET to be an output pin. The 8251A automatically outputs a 1 to the SYNDET/BRKDET pin upon detecting that the SYNC character(s) had been received from the modem.

However, the processor may configure the 8251A to be in an external SYNC DETECT mode. In this mode, the following occur:

1. The pin SYNDET/BRKDET becomes an input pin.

2. The 8251A no longer waits for a SYNC character(s) from the modem, but rather waits for a positive-going signal to be applied at the SYNC/BRKDET pin.
3. Once the 8251A senses that a positive-going signal is applied to its SYNC/BRKDET, it continues with processing the incoming characters as though it had detected a regular SYNC character(s).

Parameter 7: One or two SYNC characters. This parameter is applicable only to the synchronous protocol. Recall that in a synchronous transmission, the modem sends to the 8251A a string of characters preceded by one or two SYNC characters. This parameter configures the 8251A to search for either one or two SYNC characters. The 8251A automatically inserts the proper number of SYNC characters when sending characters to the modem.

8.14 Sending the Mode Instruction to the 8251A

When the processor wishes to configure the 8251A, it sends to the 8251A a single byte that configures all eight parameters of the protocol. This byte is called the *mode instruction.* Figure 8.13 shows the format of the mode instruction when the 8251A is to be configured to work in the asynchronous protocol. Figure 8.14 shows the format of the mode instruction when the 8251A is to be configured to work in the synchronous protocol.

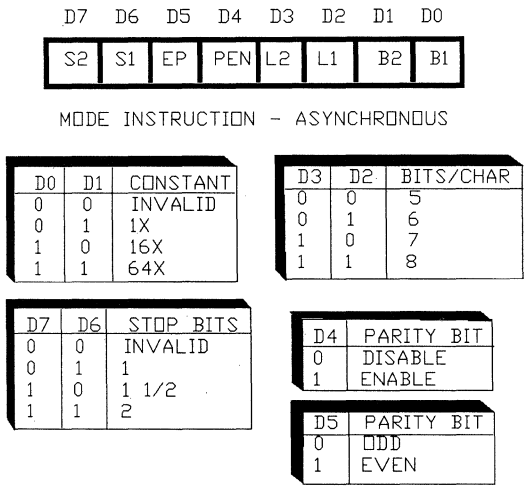


Figure 8.13 The mode instruction for asynchronous protocol.

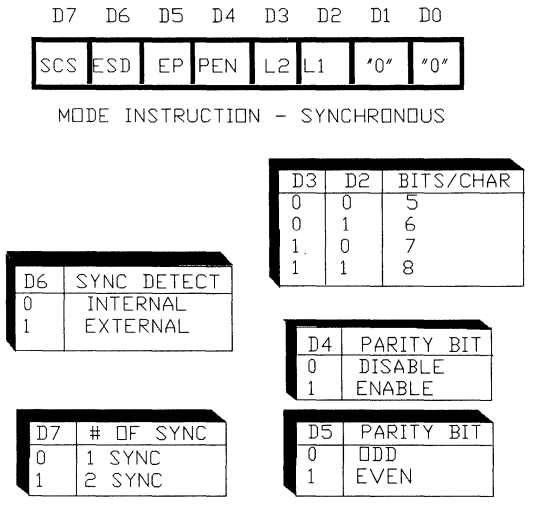


Figure 8.14 The mode instruction for synchronous protocol.

Example 8.12 Suppose that the 8251A is to be configured in accordance with the following protocol:

- Protocol: Asynchronous
- Baud rate: The same as the TxC clock
- Number of bits per character: 6
- Parity generation and checking: To be implemented
- The parity method is: Even
- Number of stop bits: 1

(Refer to Fig. 8.13.)

D7 = 0, D6 = 1: Number of stop bits = 1

D5 = 1: Even parity bit

D4 = 1: Enable parity bit generation and checking

D3 = 0, D2 = 1: Six bits per character.

D1 = 0, D0 = 1: Baud rate = 1x TxC

The processor sends the following mode instruction to the 8251A:

```
0111 0101 = 75 (hex)
```

Example 8.13 Suppose that the 8251A is to be configured in accordance with the following protocol:

- Protocol: Synchronous
- Number of bits per character: 7
- Parity generation and checking: To be implemented
- The parity method is: Odd
- SYNC DETECT mode: Internal
- Number of SYNC characters: 2

(Refer to Fig. 8.14.)

D7 = 1: Two SYNC characters
D6 = 0: Internal SYNC DETECT mode
D5 = 0: Odd parity
D4 = 1: Enables parity bit generation and checking
D3 = 1, D2 = 0: Seven bits per character
D1 = 0, D0 = 0: Must be 0 for synchronous protocol

The processor sends the following mode instruction to the 8251A:

```
1001 1000 = 98 (hex)
```

8.14.1 How to send the mode instruction to the 8251A

Once the mode instruction byte is calculated by the processor, the processor sends the byte to the 8251A. To do that, the processor simply writes the byte to the decoded address of the 8251A.

Example 8.14 Suppose that the 8251A address is decoded to

```
01230 (hex) and 01231 (hex)
```

(Recall that the 8251A has to be decoded to two addresses.) Address 01230 is used to read and write characters, and address 01231 is used to read and write status and instructions.

To send the mode instruction 98 (hex), the processor uses a command such as

```
Write 98 (hex) to address 01231
```

8.15 Sending Instructions to the 8251A

Another section of software executed by the processor is the section that has the task of sending instructions to the 8251A.

8.15.1 Eight possible instructions

The instruction set of the 8251A is rather limited; there are only eight instructions that the processor can issue to the 8251A. A list of the eight instructions follows.

Instruction 0: The TRANSMIT ENABLE/DISABLE instruction. The processor issues the TRANSMIT/DISABLE instruction to either enable or disable the 8251A's transmission function.

Instruction 1: The DTR ENABLE/DISABLE instruction. The processor issues the DTR ENABLE/DISABLE instruction to cause the 8251A to output a 0 onto the DTR pin of the 8251A.

Instruction 2: The RECEIVE ENABLE/DISABLE instruction. The processor issues the RECEIVE ENABLE/DISABLE instruction to either enable or disable the 8251A's receiving function.

Instruction 3: The SEND BREAK instruction. The processor issues the SEND BREAK instruction to force the 8251A to transmit a BREAK character to the modem. This instruction is valid only if the 8251A is working in the asynchronous mode.

Instruction 4: The ERROR RESET instruction. The processor issues the ERROR RESET instruction to force all the error flags to be reset.

The 8251A has a register called the *status read* register. This register contains, among other things, three error flags. Whenever there is a transmission error, the 8251A automatically records the error in its status read register.

The processor can read the status read register and analyze the errors that are reported in the register. Once the processor finishes analyzing the errors, it sends the ERROR RESET instruction to reset the three error flags.

Instruction 5: The REQUEST TO SEND instruction. The processor issues the REQUEST TO SEND instruction to the 8251A, causing the 8251A to output a 0 onto the RTS pin of the 8251A.

Instruction 6: The INTERNAL RESET instruction. The processor issues the INTERNAL RESET instruction to force the 8251A to enter the reset state (software reset).

Instruction 7: The ENTER HUNT instruction. The processor issues the ENTER HUNT instruction to force the 8251A to enter the HUNT state. This instruction is valid only if the 8251A is in the synchronous mode.

In the synchronous protocol, there are SYNC characters preceding the string of characters. It is via the SYNC character(s) that the 8251A identifies the beginning of a string character.

While in the HUNT state, the 8251A examines each incoming character and compares the incoming character to the SYNC character. If the 8251A is configured to work with two SYNC characters, the 8251A continues to be in the HUNT state until the 8251A receives two consecutive SYNC characters. Whether the 8251A searches for one SYNC character or two SYNC characters is dependent on how the 8251A is configured.

Note that in order for the 8251A to search for the SYNC character(s), it is not enough to configure the 8251A to be in the internal

SYNC DETECT mode. The processor must issue the ENTER HUNT instruction.

8.15.2 Sending the instruction to the 8251A

When the processor wishes to send an instruction to the 8251A, it sends a single byte to the 8251A. This byte is called the *command instruction.* Figure 8.15 shows the format of the command instruction.

Example 8.15 What is the value of the command instruction when the processor issues the instruction: INTERNAL RESET?

Solution Since all eight instructions are combined into one byte as shown in Fig. 8.15, there is no way to send just the INTERNAL RESET instruction. That is, by issuing the command instruction, all eight instructions are issued. Thus, let us assume that the processor wishes to issue the following eight instructions:

```
DISABLE TRANSMITTER
NEGATE THE DTR PIN
DISABLE RECEIVER
DO NOT SEND THE BREAK CHARACTER
ERROR RESET
NEGATE THE RTS PIN
INTERNAL RESET
DISABLE HUNT MODE
```

From Fig. 8.15 it follows that the corresponding bits of the command instruction are

D0 = 0: DISABLE TRANSMITTER
D1 = 1: NEGATE THE DTR PIN
D2 = 0: DISABLE RECEIVER
D3 = 0: DO NOT SEND THE BREAK CHARACTER
D4 = 0: ERROR RESET
D5 = 0: NEGATE THE RTS PIN

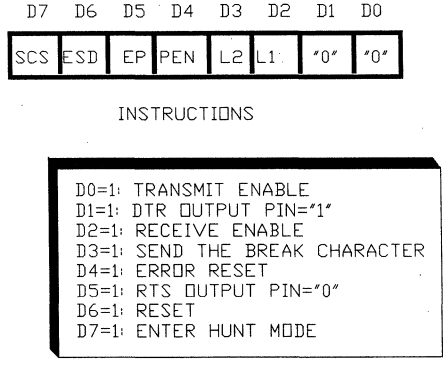


Figure 8.15 The instruction.

D6 = 1: INTERNAL RESET
D7 = 0: DISABLE HUNT MODE

The command instruction byte is

```
0100 0010 = 42 (hex)
```

8.15.3 How to send the command instruction to the 8251A

Once the command instruction byte is calculated by the processor, the processor needs to send the byte to the 8251A. To do that, the processor simply writes the byte to the decoded address of the 8251A.

Example 8.16 Suppose that the 8251A address is decoded to address 01230 and 01231 (recall that the 8251A is decoded to two addresses). Address 01230 is used to read and write characters, and address 01231 is used to read and write status and instructions.

To send the command instruction 42 (hex), the processor uses a command such as

```
Write 42 (hex) into address 01231
```

8.16 Monitoring the Operation of the 8251A

Following is a list of status information that the processor can retrieve from the 8251A. There are a total of eight pieces of status information.

Item 0: The TxRDY flag. This flag is set automatically by the 8251A whenever the 8251A senses that its data buffer is empty. This is used by the processor as an indication that the 8251A is ready to accept a new character from the processor. The 8251A resets this flag upon sensing that the processor writes a character into its data buffer.

Item 1: The TxEMPTY flag. This flag is set automatically by the 8251A whenever the 8251A senses that its transmit buffer is empty. The 8251A provides this feature only if the TRANSMIT ENABLE instruction was previously issued.

Item 2: The RxRDY flag. The 8251A automatically sets this flag upon finishing processing a received character from the modem. This flag is an indication to the processor that a character in the 8251A is ready to be fetched.

Item 3: The SYNDET/BRKDET flag. In synchronous mode, the 8251A sets this flag whenever the SYNC character(s) are detected. In synchronous mode, the 8251A sets this flag whenever the break character is detected.

Item 4: The DSR flag. This flag is set whenever the DSR input pin of the 8251A is asserted.

Item 5: The parity-error flag. If the parity error check is enabled (via configuration) and the 8251A received a character with a parity error, then the 8251A sets the parity error flag.

Note that once the 8251A detects a parity error, everything else continues to work. That is, even if a parity error were detected, the 8251A continues transmitting bits to the modem as well as receiving bits from the modem.

The processor can reset the parity error flag by issuing the ERROR RESET instruction. If one character is received with parity error while the next character is received with no parity error, the error flag remains set. It remains set until the processor explicitly issues the ERROR RESET instruction.

Item 6: The overrun error flag. The processor should fetch an incoming character from the 8251A upon receiving an indication from the 8251A (via the RxRDY flag) that a character was received, assembled, and ready to be fetched. Suppose however that the processor does not fetch a ready character. Suppose further that the 8251A receives a new character from the modem. The new character is being processed by the 8251A, and eventually this new character is placed in the received buffer of the 8251A. If by that time the processor still did not fetch the old character, the old character is being written over and lost forever.

In such a case, the 8251A sets the overrun error flag. This enables the processor to detect that it missed a character. (This feature does not enable the processor to retrieve the lost character, but at least the processor software will be able to detect that a miss occurred.)

The processor can reset the flag by issuing the ERROR RESET instruction. If one character is overwritten while the next character is fetched in time, the error flag remains set.

Item 7: The framing error flag. This error flag is applicable only if the 8251A is configured to work in the asynchronous protocol. If while receiving characters from the modem, the 8251A discovers that a stop bit is missing, it sets the framing error flag.

Example 8.17 The reader might wonder how it is possible to detect something that is missing. The answer is that a stop bit is always a 1. The 8251A knows that the first bit is a start bit, the next N bits are the character bits, the next bit is the parity bit, followed by the stop bit(s). Therefore, the 8251A calculates the location of the stop bit(s) (in the same way that we calculated it here) and expects to see 1 at the location of the stop bit(s).

Note that the 8251A continues its operation even if a framing error occurred. To reset the framing error flag, the processor must explicitly issue the ERROR RESET instruction.

8.17 The Read Status Byte

All of the above eight flags have the format shown in Fig. 8.16.

8.17.1 How to read the read status byte

The processor reads the read status byte by simply reading the address of the 8251A. Recall that the 8251A has to be decoded to two addresses. One address has the A0 bit equals to 0, and the other address has the A0 bit equal to 1. The read status byte is read by reading the address with bit A0 = 1.

Example 8.18 Suppose that the 8251A address is decoded to address 01230 and address 01231. To read the read status byte, the processor uses a command such as

```
Read the value from address 01231
```

8.18 Fetching Received Characters from the 8251A

The processor fetches a character from the 8251A by simply reading from the decoded address of the 8251A. Recall that the 8251A is

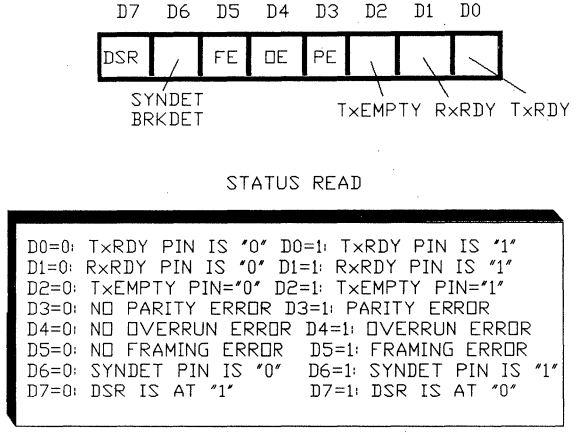


Figure 8.16 The status read byte.

decoded to two addresses. One address has A0 = 0, and the other address has A0 = 1. The character is read by reading from the address that has A0 = 0.

Example 8.19 Suppose that the 8251A addresses are the decoded addresses: 01230 and 01231. To fetch a character from the 8251A, the processor software utilizes an instruction such as

```
Read a value from address 01230
```

8.19 Writing Characters to Be Transmitted to the 8251A

The processor writes a character to be transmitted to the 8251A by simply writing to the decoded address of the 8251A. Recall that the 8251A has two decoded addresses. One address has A0 = 0, and the other address has A0 = 1. The character to be transmitted is written to the decoded address that has A0 = 0.

Example 8.20 Suppose that the 8251A address is decoded to address 01230 and address 01231. To write character-to-be-transmitted = 04 (hex) to the 8251A, the processor software utilizes an instruction such as

```
Write the value 04 (hex) to address 01230.
```

8.19.1 The software of the processor

At this point we are equipped with all the software components that enable the processor to use the 8251A for telephone network communication (serial communication). These components include

- Configuring the 8251A (sending the mode instruction byte).
- Sending instructions to the 8251A (sending the command instruction byte).
- Reading the status of the 8251A (reading the read status byte).
- Fetching received characters from the 8251A.

To establish communication, the processor software must follow these steps (refer to the flowchart diagram shown in Fig. 8.17):

Step 1: Resetting the 8251A. Before the processor can access the 8251A, the 8251A must be reset.

Step 2: Configuring the 8251A. Once the 8251A is properly reset, it is ready to accept the first byte from the processor. This first byte must be the mode instruction byte, which configures the 8251A. The 8251A

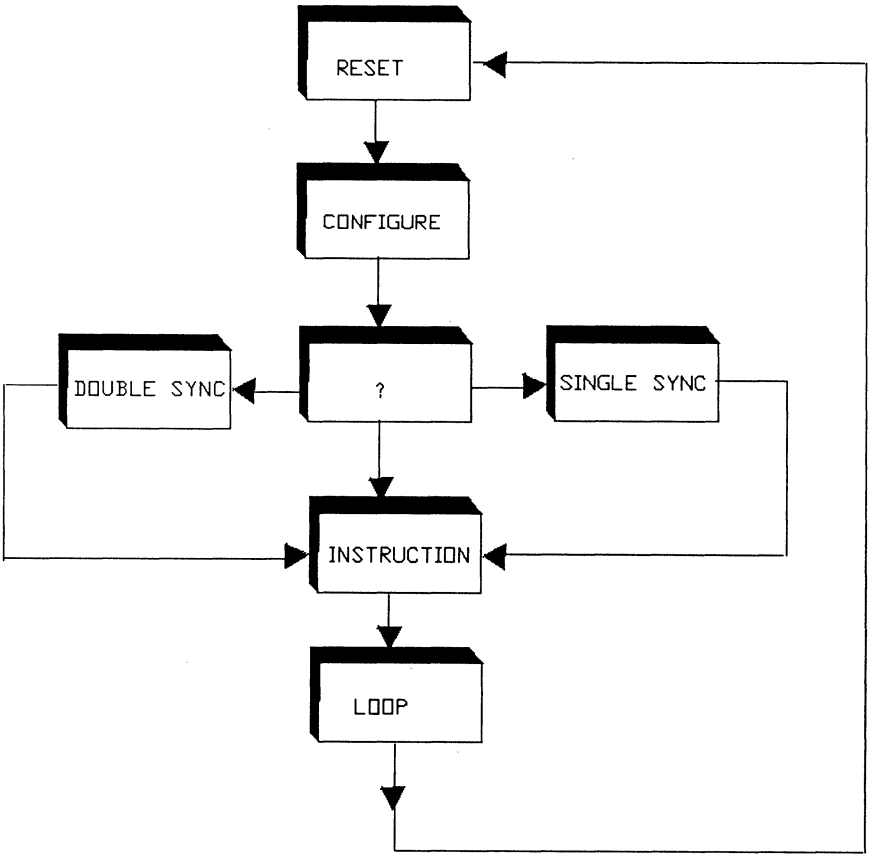


Figure 8.17 The processor software flow chart.

does not default to any particular configuration, and it expects the first byte to be the mode instruction.

Step 3: The first SYNC character. If the 8251A is configured to work in the asynchronous mode, this step is skipped.

Assume that the 8251A is configured to work in the synchronous mode with a single SYNC character or double SYNC characters. This being the case, the 8251A knows that it must search for the SYNC character(s) among the incoming bits from the modem. When transmitting to the modem, the 8251A is responsible for the insertion of the SYNC character(s).

In both operations, the 8251A must be told in advance the value of the SYNC character to be searched for and to be inserted. In this step the processor sends the SYNC character to the 8251A. Note that the processor sends the SYNC character to the 8251A only once after each reset.

To write a SYNC character to the 8251A, the processor software utilizes the following instruction:

```
Write the SYNC character to the 8251A decoded address
that has A0 = 1.
```

Step 4: The second SYNC character. If the 8251A is configured to work in the asynchronous mode, this step is skipped.

If the 8251A is configured to work in the synchronous mode with a single SYNC character, this step is skipped.

If the 8251A is configured to work in the synchronous mode with double SYNC characters, the 8251A must be told the value of the second SYNC character. In this step, the processor informs the 8251A of the value of the second SYNC character by writing the value to the 8251A decoded address that has A0 = 1.

Step 5: Sending the first instruction command to the 8251A. At this point, the 8251A is expecting to receive the first command instruction from the processor.

The rest of the steps (the loop). In the rest of the steps the processor writes characters to be transmitted to the 8251A, fetches received characters from the 8251A, reads the status of the 8251A, and writes command instructions to the 8251A. In Fig. 8.17, this step is shown as the loop-block.

The order in which the rest of the steps are implemented depends on the objective of the software (e.g., full-duplex, half-duplex, receiving only, etc.).

Note from the flowchart of Fig. 8.17 that if the processor wishes to reconfigure the 8251A (sending a new mode instruction), the processor *must* first reset the 8251A.

Example 8.21 Suppose that the 8251A is decoded to address 01230 and 01231. Suppose also that it is required to put the 8251A in a full-duplex synchronous mode. Following is the list of steps that the processor software is executing to implement the requirement.

Step 1 Reset the 8251A.

Step 2 (Refer to Fig. 8.14). Send the mode instruction:

```
D0 = 0, D1 = 0:
This configures the 8251A to work in the synchronous mode.

D2 = 0, D3 = 0:
Five bits per character
```

```
D4 = 1:
Enable parity checking-generation.

D5 = 1:
Even parity.

D6 = 0:
Internal SYNC DETECT mode.

D7 = 0:
Double SYNC character.
```

Thus, the processor software executes the instruction:

```
Write 0011 0000 into address 01231
```

Step 3 Write the first SYNC character. The processor software executes the instruction:

```
Write SYNC#1 into address 01231
```

Step 4 Write the second SYNC character. The processor software executes the instruction:

```
Write SYNC#2 into address 01231
```

Step 5 (Refer to Fig. 8.15.) In this step the command instruction is calculated and written to the 8251A.

```
D0 = 1:
Enable the transmitter.

D1 = 1:
Assert the DTR pin.
Informing the modem that it is OK to send bits to the 8251A.

D2 = 1:
Enable the receiver.
The system is ready to accept bits from the modem.

D3 = 0:
Do not send the break character.
(Break character is applicable in asynchronous mode only.)

D4 = 0:
Do not reset the error flags.
These flags are reset during the RESET state.

D5 = 1:
Assert the RTS pin.
Alert the modem to be ready to receive bits soon.

D6 = 0:
Do not reset the 8251A.

D7 = 1: Enter the hunt mode.
```

The processor software executes the instruction:

```
Write 1010 0110 into address 01231
```

The rest of the steps Now the processor should be in a loop, executing the following:

```
LOOP: Read the status byte. If the TxEMPTY flag indicates
      that the data buffer is empty, write a character to
      be transmitted. (The character is transmitted
      automatically when the modem asserts the CTS pin.)

      Read the status byte from the 8251A to examine the
      RxRDY flag to see if a character was received from
      the modem, and if the character is ready to be
      fetched.

      If there is a received character in the 8251A, fetch
      it.

      Go back to LOOP.
```

The processor should exit the loop when it discovers that the received bits are all 1s. The processor may read the status bytes to determine if any of the received characters were received with a parity error or if an overrun error occurred. (If so, the processor may transmit a message, asking the sender to retransmit the last string of characters.)

Once a string is received or transmitted, the processor may reset all the flags, issue the ENTER HUNT instruction, and start the loop again to receive (and transmit) the next string of characters.

Software development. Writing the processor software for the purpose of 8251A communication is similar to writing software for other applications. The programmer has to practice and become familiar with each of the instructions and features that the 8251A offers. One popular method of developing such software is to experiment with two known-to-work stations (e.g., configure one station to receive bits, configure the other station to transmit bits, then examine whether the data sent from the transmitter is received by the receiver).

It is also possible to experiment with a single station. Connect the transmitter of the 8251A (the TxD pin) to the receiver pin of the 8251A (the RxD pin). This method enables the examination of full-duplex operation (the system transmits and receive at the same time).

Appendix A

A.1 Choosing the Right Network

Once an organization decides that it requires a LAN, the people who construct the LAN have to make the decision of which type of network should be procured. Basically, the LAN designers have to choose between a probabilistic network (i.e., 802.3) and a deterministic network.

Among the available deterministic types are networks such as token bus and token ring. While it is difficult to predict the future of the LAN market, at this point in time, it does look as though many organizations prefer the probabilistic networks over the deterministic networks. The main reason for the success of the 802.3 network is that it proved itself to be reliable.

The 10BASE-T network is anticipated to dominate the LAN market since it contains all the ingredients that make a LAN popular. That is, it uses the very reliable probabilistic 802.3 protocol, its data rate is 10 Mbit/s, it requires inexpensive twisted pairs of wires, and the topology of the existing telephone wiring may be utilized.

While the probabilistic network is most suitable for office applications, the deterministic networks are most suitable for applications such as automatic manufacturing where it is desired that each station be able to transmit for a predetermined period of time in a predetermined priority scheme.

A.2 The Line Integrity Test Feature of the 10BASE-T

The 10BASE-T protocol specifies the implementation of a *line integrity test.* The line integrity test is a procedure whereby the node attached to the multiport repeater periodically sends short pulses to the repeater. Upon receiving these pulses, the repeater concludes that indeed a node is attached to the port. If the repeater does not receive the integrity

pulses at any of its twisted ports, it treats that port as if no node is attached to the port. Some multiport repeaters are designed with a disable line integrity test mechanism, in which the multiport repeater responds to nodes even if no line integrity pulses are received from the node.

A.3 Minimally Configured Station

A station in a LAN is composed of data terminal equipment (DTE) and a node board. In many cases, the use of a personal computer is the most economical way to construct an inexpensive station. For example, a PC XT and a node board are all the hardware components that are required to implement a station. The PC XT may have minimal configuration. That is, a minimally configured XT consists of an XT motherboard, keyboard, monochrome driver card, monochrome monitor, and 256K bytes RAM. Note that such a station does not have a hard drive or a floppy drive.

Example A.1 An inexpensive station that is built by plugging a node board into a minimally configured PC XT is able to function without having a hard drive or a floppy drive. This is accomplished by incorporating a boot PROM into the design of the node board. Upon applying power to the PC XT, the ROM BIOS of the XT (i.e., the software that resides on a PROM chip on the motherboard of the XT and is being executed upon power up) is looking for a hard drive (or a floppy drive) for the purpose of booting up the DOS files. If the program finds that the system does not contain a hard drive or a floppy drive, the program automatically searches for a program located in a predetermined address location and executes that program. The node board thus must contain a PROM which resides on the node board and is decoded to that address location. This PROM contains the proper initialization routines that enable the node to send a request via the network cable to a remote location, asking that remote location to send DOS files and other related network files. Once these files are received by the node, these files are loaded into the RAM memory of the XT. Upon completing loading all the required files, the XT is able to function as a station. The above sequence of events is illustrated in Fig. A.1.

A.4 The File Server

The previous example assumes that there is a station in the network which has the responsibility of sending boot files to minimally configured stations in the network. This station is called the *file server.* The file server is responsible not only for booting up the various stations but also for containing the various files that are required for normal operation of the various stations. As an example, the file server might contain a word processor program, databases, and other programs and files that are accessed by the station (recall that in a minimally configured station, the station has no means of storing programs or files

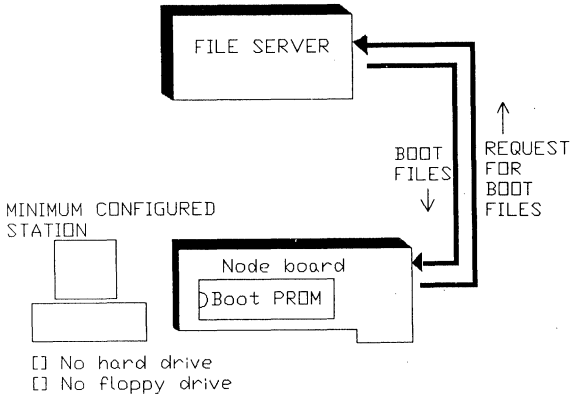


Figure A.1 Constructing a station by using a minimum configured XT with a boot PROM.

since it does not have floppy or hard drive). A 10BASE-T network that incorporates a file server is shown in Fig. A.2. Note that the multiport repeater treats the file server as a regular station.

A.4.1 The file server as a means for storing databases

The minimally configured station is an excellent solution for applications where the user at the station site does not require its own mass storage devices. An example of such an application is a reservation station where data are extracted from a large database residing in the hard drive of the file server. The station is capable of requesting data from the file server, as well as updating the database of the file server. Other applications that require a single focal point for storing a database are inventory databases.

There are applications where it is desired to have a nonminimally configured station. Such a station might be an advanced PC (e.g., PC AT 286/386/486, PS/2) that contains its own hard drive. For example, such stations might be used as engineering work stations where the purpose of connecting the stations to the network is to share resources

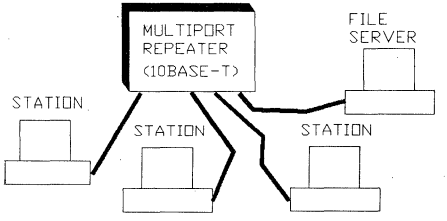


Figure A.2 The file server in a 10BASE-T environment.

such as plotters and large expensive programs. In such cases, the station already contains all the proper boot files in its hard drive, and the boot PROM of the node board is therefore not used.

Example A.2 From the above discussion it may be concluded that there are two types of node boards demanded by the market. One type is a node board that is basically intended for use by a minimally configured station that requires the transfer (transmission and reception) of relatively short files between the file server and the station. Since a minimally configured station might consist of an XT (which has only 8 data bits, D0, D1, . . . , D0, on its I/O slot), an inexpensive node board may well accommodate such an application. These node boards are usually inexpensive since they do not contain a microprocessor on board and have a small amount of RAM on board. The ability of a minimally configured station to capture incoming frames is limited by the limited performances of the 8-bit XT machine. Nevertheless, in most applications an XT machine is adequate. Note that if for some reason the station did not capture all the incoming frames, it is the responsibility of the higher level software to detect the fact that acknowledgment had not been received and to resend the corresponding frames.

On the other hand, a station that is involved in the transmission and reception of large files might require a more sophisticated node board that is able to accommodate a 16-bit data bus that exists on the I/O slot of the more advanced PCs. These node boards might contain a microprocessor on board and additional RAM. Some of the node manufacturers claim that their node boards are able to perform back-to-back capture of frames. That is, the node is designed so that even if frames are sent to the node one after the other, the node is capable of capturing all the frames.

Example A.3 While currently most of the nodes are manufactured as I/O boards to be plugged into the motherboard of the PC, there is a growing trend to incorporate the node on the motherboard itself. For such designs, the Intel 82590 family of chips is most suitable.

A.5 Intelligent Repeaters

There are applications where it is required to monitor the operation of the network in great detail. An *intelligent repeater* is a combination of hardware and software, usually incorporated within the design of the repeater, which monitors and keeps count of the number of collisions that occur and makes a statistical analysis such as the average length of frames and variety of other traffic information on the network cable. The information is being analyzed, and based on it, the people who are in charge of maintaining the network are able to recommend ways and means of improving the efficiency of the network. An intelligent repeater usually includes features that allow the remote disconnection of stations from the repeater and, thus, ease the traffic load on the cable. The intelligent repeater equipment also allows performing diagnostics of the network, discovering whether there is a faulty node in the network, and disconnecting such a node automatically.

A.6 Combining Several Networks

Large organizations might have a need to combine several LANs (Fig. A.3). The backbone of the network is an FDDI (Fiber Optic). As discussed in Chap. 1, the fiber optic cable is capable of carrying data at a very high data rate for long distances with very high immunity to noise. As shown in Fig. A.3, one local network is capable of communicating with another local area network via the FDDI. To be able to achieve such communication, a *gateway circuitry* is required. The gateway circuitry accepts data from the local area network and formats it to a data stream in accordance with the protocol of the FDDI. This data stream includes the address of the destination gateway. The data are rotated on the FDDI, and eventually are recognized and accepted by the destination gateway. The data are then converted from an FDDI format to the format of the destination local area network.

A.7 Node Drivers

The driver software is a software executed by the PC to allow the interfacing of the node board to a particular high level network software. Usually the manufacturer of the high-level software sells a library of software drivers as an integral part of the software. The manufacturer of the high-level software also supplies a routine (usually called a shell

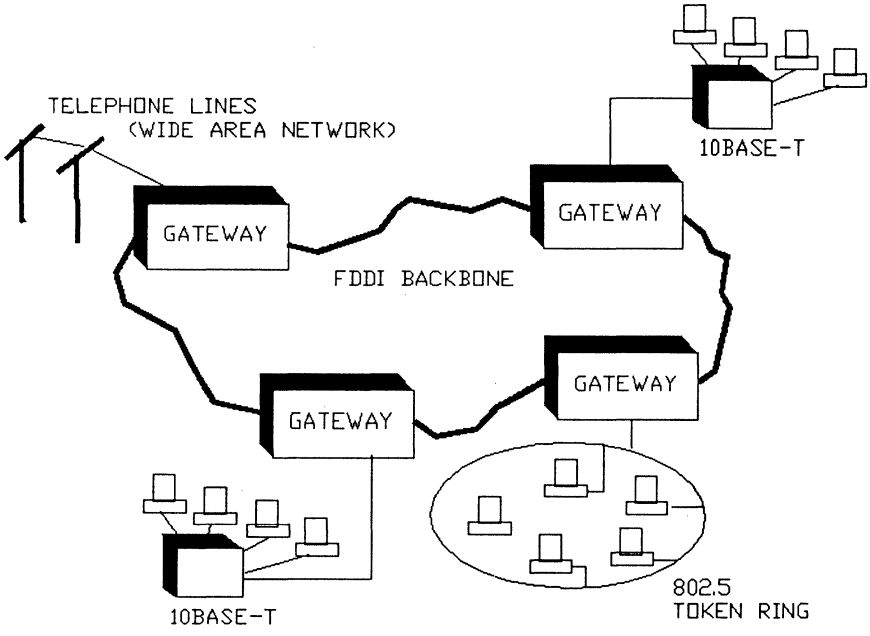


Figure A.3 FDDI backbone.

generator) which sets up the driver software for the particular node card. As an example, the shell generator asks the user to indicate the interrupt level that the node card will use to interrupt the PC. The shell generator is also required to know what I/O address and memory address are reserved for the node card. Based on these values the shell generator sets up the proper parameters in driver software.

Example A.4 In the Novell environment, a user at the station site may type

```
F: {Enter}
```

which means: CHANGE DIRECTORY TO THE ROOT DIRECTORY OF THE FILE SERVER. (Once this command had been executed successfully, the user may type DIR to examine the files of the root directory of the file server.)

Upon executing the F: command, the driver software at the station site sends the proper bytes to be transmitted to the node board. The node board formats the bytes in accordance with the frame protocol and transmits it over the network wires.

On the file server side, the node board recognizes that the frame is intended for itself (by recognizing the destination address of the frame). The node interrupts the file server and loads the received data into the file server. The higher level software then analyzes the received data and interprets the request. The file server acknowledges the request by formatting a certain sequence of bytes and then instructs its node to transmit the bytes to the requesting station.

On the station side, the node receives the acknowledgment frame and transfers the acknowledgment to the PC. The acknowledgment is being translated to the user by displaying

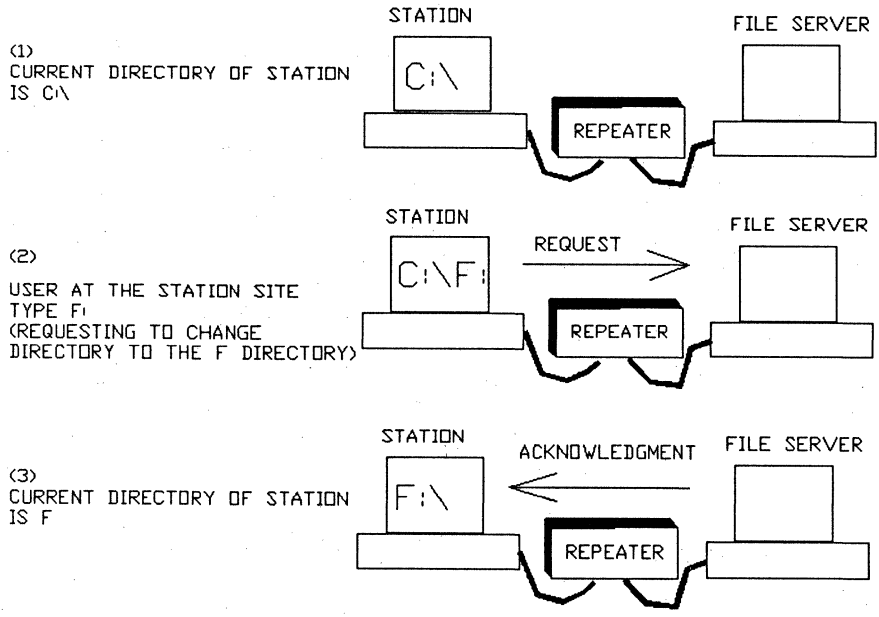


Figure A.4 Requesting the F: drive.

```
F:\
```

on the monitor. (This process is shown in Fig. A.4.)

A.8 Interfacing to the I/O of the PC

As mentioned above, many node cards are designed to be plugged into the I/O slot of the PC. As such, it is important not to create an address conflict with other existing I/O cards in the PC. As a matter of a fact, installing node cards is an easy process; simply plug the card into an available I/O slot. However, the installer must make sure that the address of the node card is not already used by other I/O cards. The following is a standard I/O port address map used in the industry.

I/O Hex address	I/O card
000–1FF	Used by the motherboard.
200	Not used.
201	Game port.
202–277	Not used.
278–27F	Second printer port.
280–2F7	Not used.
2F8–2FF	COM2 (serial port).
300–377	Not used.
378–37F	Printer port.
380–3AF	Not used.
3B0–3BF	Monochrome and printer.
3C0–3CF	Not used.
3D0–3DF	Color and graphics.
3E0–3EF	Not used.
3F0–3F7	5.25-in diskette drive.
3F8–3FF	COM1 (serial port).

The node card should be designed with DIP switches (or jumpers) to allow the user to program the node card to any desired I/O address.

Example A.5 Is it legal to set the node card jumpers so that the node card is decoded to address 201(hex)?

Solution Address 201(hex) is reserved for the game port. If no game port exists in the system, address 201 would be legal. Yet it would not be a good practice to set the node card to address 201(hex) even if there is no game port in the system. The reason is that at a later time, somebody might decide to attach a game port to the system, not realizing that the I/O address is already used by the node card. A better choice would be to set the jumpers of the node card to address 300 (provided that no other I/O card uses this address).

A.8.1 Other settings in the node card

In addition to setting the I/O address decoding, the node card has to be set (via the setting of jumpers) to a particular memory address and to a particular interrupt level (IRQ). Again, before setting these jumpers, the installer must make sure that no other I/O card would create a con-

flict in the system. The IRQ setting determines the IRQ channel that the node would use to interrupt the PC, and the memory address setting determines the addresses of RAM on the node board that the PC will use to write bytes to be transmitted and to read received bytes.

Once the I/O address, the memory address, and the IRQ jumpers are set, the installer has to run the shell generator software, which incorporates these settings into the driver software.

A.8.2 The source address of an Ethernet node card

As mentioned in previous chapters, the source address of a node in an 802.3 network is composed of 6 bytes. The range represented by 6 bytes is 000000 to FFFFFF (hex) which is equivalent to 0 to 16,777,216 (decimal). Most manufacturers assign a unique fixed source address to each of the node cards manufactured. This is done by incorporating a PROM on the node card that contains the predetermined source address. Upon booting, the source address is extracted from this PROM. It is the responsibility of the manufacturer to ensure that each of its node cards will carry a unique embedded source address. The following example illustrates how the decoding of the I/O address of a node card is accomplished.

Example A.6 Figure A.5 demonstrates how the I/O address is decoded by using the 74LS688 comparator. The P = Q pin of the 74LS688 is asserted (equal to 0) only if P0 = Q0, P1 = Q1, P2 = Q2, P3 = Q3, P4 = Q4, P5 = Q5, P6 = Q6, and P7 = Q7. The signal AEN is connected directly from the I/O connector (on the motherboard of the PC) to pin P7 pin of the 74LS688. The AEN (address enable) signal is an output signal generated by the DMA control circuit on the motherboard. The AEN is asserted (equal to 1) whenever there is a DMA bus cycle in progress. During a DMA cycle (e.g., during the refreshing of the dynamic RAM on the motherboard), the PC devotes its address bus and data bus for the DMA cycle, and no I/O read or write should occur during the DMA cycle. Since Q7 of the 74LS688 is directly tied to ground, P7 is not equal to P7 whenever there is a DMA cycle in progress. During the time when there is no DMA cycle in progress, AEN is 0, and the P = Q pin is asserted provided that P0 = Q0, P1 = Q1, P2 = Q2, P3-Q3, P4 = Q4, P5 = Q5, and P6 = Q6.

For the setting shown in Fig. A.5 (all switches are open), Q0 = Q1 = Q2 = Q3 = Q4 = Q5 = Q6 = 1. This means that the P = Q pin is asserted whenever the PC is not performing a DMA cycle and the address bus contains A9 = 1, A8 = 1, A7 = 1, A6 = 1, A5 = 1, A4 = 1, and A3 = 1. A read or write to address: XX11 1111 1XXX (X = don't care) would cause the P = Q pin to be asserted. For example, writing to address 3F8 would cause the P = Q pin to be asserted.

As shown in Fig. A.6, the P-Q pin is connected to the G2 pins (enable pins) of the two 74LS138 chips. Therefore, these chips are enabled only if the PC reads or writes to an address set by the DIP switches. The G3 pin (enable pin) of the upper 74LS138 chip is connected directly to the IOR signal of the I/O connector. The IOR signal is asserted by the PC whenever the PC executes an I/O read instruction. Thus, the upper 74LS138 is enabled whenever the PC is executing

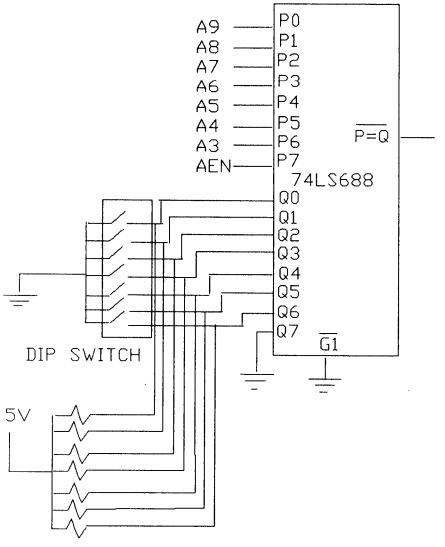


Figure A.5 Decoding I/O address for the node board.

an I/O read instruction from an address decoded by the DIP switches. Since the A0, A1, and A3 signals of the I/O connector are directly connected to the A, B, and C pins of the 74LS138, the 74LS138 asserts the Y pin, which corresponds to the A0, A1, A2 signals. For example, whenever the PC is executing an instruction to read address 3F8, the Y0 pin is asserted. If the PC reads the address 3F9, the Y1 pin is asserted.

Similarly, the G3 pin of the lower 74LS138 in Fig. A.6 is connected to the IOW signal of the I/O connector. The IOW signal is asserted whenever the PC is exe-

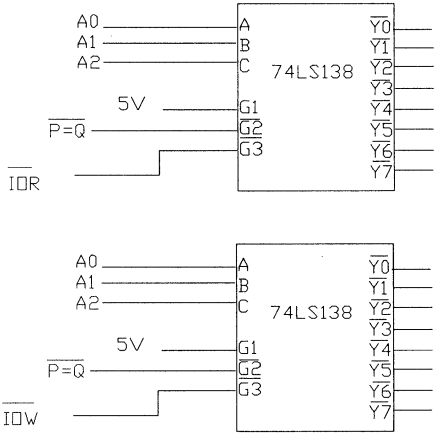


Figure A.6 The I/O read and write circuitry of the node board.

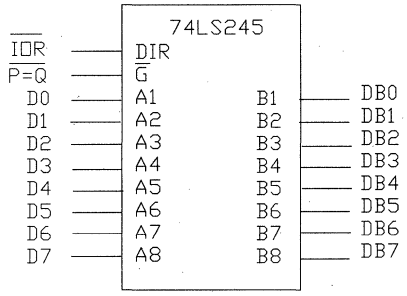


Figure A.7 Connecting the I/O data bus.

cuting an I/O write instruction. Thus, whenever the PC writes to an I/O address decoded by the DIP switches, the lower 74LS138 asserts the Y pin that corresponds to that address. For example, for the switch setting of Fig. A.5, the Y0 pin is asserted whenever the PC is writing to address 3F0. The Y1 pin is asserted whenever the PC is writing to the I/O address 3F9.

Figure A.7 illustrates how the bidirectional data bus (D0, D1, D2, D3, D4, D5, D6, and D7) from the I/O connector is connected to the 74LS245. The 74LS245 is bidirectional and is able to transfer the data signals in the direction A to B (from the I/O connector to the node board) as well as in the direction of B to A (from the node board to the I/O connector). The DIR pin of the 74LS245 determines the direction of data flow. When a 0 is applied to the DIR pin, the 74LS245 transfers the data in the B to A direction.

Since the IOR signal is connected to the DIR pin, data would flow from the B side to the A side whenever the IOR signal is 0. Thus, whenever the PC reads an I/O address, the data are able to flow into the PC. As shown, the P = Q signal is connected to the G (enable) pin of the 74LS245, which enables the 74LS245 only if the PC is executing an instruction that involves reading an I/O address decoded by the DIP switch.

In addition to I/O decoding, it is also required to decode memory addresses on the node board (e.g., to enable the PC to read and write bytes into the RAM and PROM of the node board). The memory decoding circuitry is similar to that of the I/O decoding circuitry. The only difference is that instead of utilizing the IOR and IOW signals, the MEMR and MEMW signals are used. The MEMR and MEMW signals are available on the I/O connector; the MEMR is asserted by the PC whenever the PC is reading a memory address, and the MEMW signal is asserted by the PC whenever the PC is writing to a memory address.

In addition to the above I/O decoding and memory decoding, most node boards design utilize one of the IRQ (interrupt request) lines that are available on the I/O connector.

Appendix

B

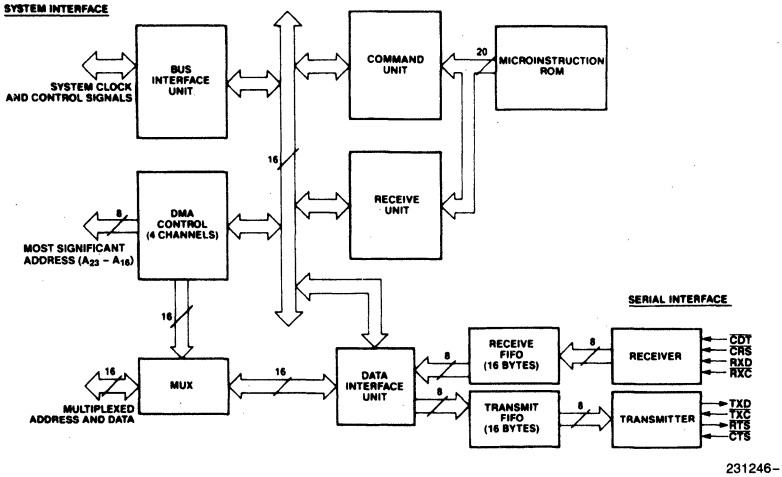
82586 IEEE 802.3 Ethernet LAN Coprocessor*

*Reprinted by permission of Intel Corporation. Copyright/Intel Corporation 1990.

- **Performs Complete CSMA/CD Medium Access Control Functions Independently of CPU**
  - **— High-Level Command Interface**
- **Supports Established and Emerging LAN Standards**
  - **— IEEE 802.3/Ethernet (10BASE5)**
  - **— IEEE 802.3/Cheapernet (10BASE2)**
  - **— IEEE 802.3/StarLAN (1BASE5)**
  - **— Proposed 10BASE-T**
  - **— Proposed 10BASE-F**
  - **— Proprietary CSMA/CD Networks up to 10 Mb/s**
- **On-Chip Memory Management**
  - **— Automatic Buffer Chaining**
  - **— Buffer Reclaim After Receipt of Bad Frames**
  - **— Save Bad Frames, Optionally**
- **Interfaces to 8-Bit and 16-Bit Microprocessors**
- **48-Pin DIP and 68-Pin PLCC**

(see "Intel Packaging" Document, Order Number: 231369-001)

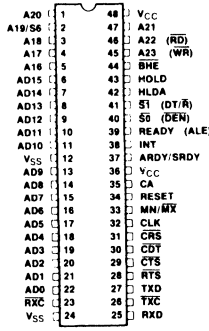
- **Supports Minimum Component Systems**
  - **— Shared Bus Configuration**
  - **— Interface to 80186 and 80188 Microprocessors Without Glue**
- **Supports High-Performance Systems**
  - **— Bus Master, with On-Chip DMA**
  - **— 5-MB/s Bus Bandwidth**
  - **— Compatible with Dual-Port Memory**
  - **— Back-to-Back Frame Reception at 10 Mb/s**
- **Network Management**
  - **— CRC Error Tally**
  - **— Alignment Error Tally**
  - **— Location of Cable Faults**
- **Self-Test Diagnostics**
  - **— Internal Loopback**
  - **— External Loopback**
  - **— Internal Register Dump**
  - **— Backoff Timer Check**



231246-1

Figure 1. 82586 Functional Block Diagram

*IBM is a trademark of International Business Machines Corporation.



231246-2

NOTE:
The symbols in parentheses correspond to minimum mode.

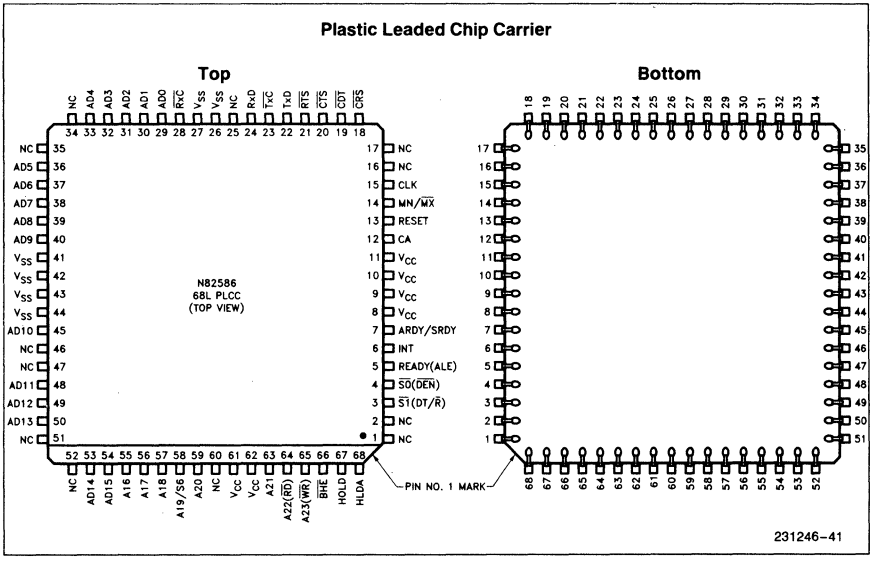


Figure 2. 82586 Pinout Diagrams

The 82586 is an intelligent, high-performance Local Area Network coprocessor, implementing the CSMA/CD access method (Carrier Sense Multiple Access with Collision Detection). It performs all time-critical functions independently of the host processor, which maximizes performance and network efficiency.

The 82586 performs the full set of IEEE 802.3 CSMA/CD Medium Access Control and channel interface functions including: framing, preamble generation and stripping, source address generation, destination address checking, CRC generation and checking, short frame detection. Any data rate up to 10 Mb/s can be used.

The 82586 features a powerful host system interface. It automatically manages memory structures with command chaining and bidirectional data chaining. An on-chip DMA controller manages four channels transparently to the user. Buffers containing errored or collided frames can be automatically recovered. The 82586 can be configured for 8-bit or 16-bit data path, with maximum burst transfer rate of 2 or 4 MB/s respectively. Memory address space is 16 megabytes maximum.

The 82586 provides two independent 16-byte FIFOs, one for receiving and one for transmitting. The threshold for block transfer to/from memory is programmable, enabling the user to optimize bus overhead for a given worst case bus latency.

The 82586 provides a rich set of diagnostic and network management functions including: internal and external loopbacks, exception condition tallies, channel activity indicators, optional capture of all frames regardless of destination address, optional capture of errored or collided frames, and time domain reflectometry for locating faults in the cable.

The 82586 can be used in either baseband or broadband networks. It can be configured for maximum network efficiency (minimum contention overhead) for any length network operating at any data rate up to 10 Mb/s. The controller supports address field lengths of 1, 2, 3, 4, 5, or 6 bytes. It can be configured for either the IEEE 802.3/Ethernet or HDLC method of frame delineation. Both 16-bit and 32-bit CRCs are supported.

The 82586 is fabricated in Intel's reliable HMOS II 5-V technology and is available in a 48-pin DIP or 68-pin PLCC package.

Table 1. 82586 Pin Description

Symbol	48 Pin DIP Pin No.	68 Pin PLCC Pin No.	Type Level	Name and Function
V_{CC}, V_{CC}	48, 36	8, 9, 10, 11, 61, 62		System Power: +5V Power Supply.
V_{SS}, V_{SS}	12, 24	26, 27, 41, 42, 43, 44		System Ground.
RESET	34	13	I TTL	RESET is an active HIGH internally synchronized signal, causing the 82586 to terminate present activity immediately. The signal must be HIGH for at least four clock cycles. The 82586 will execute RESET within ten system clock cycles starting from RESET HIGH. When RESET returns LOW, the 82586 waits for the first CA to begin the initialization sequence.
TxD	27	22	0 TTL	Transmitted Serial Data output signal. This signal is HIGH when not transmitting.
$\overline{TxC}$	26	23	I *	Transmit Data Clock. This signal provides timing information to the internal serial logic, depending upon the mode of data transfer. For NRZ mode of operation, data is transferred to the TxD pin on the HIGH to LOW clock transition.
RxD	25	24	I TTL	Received Data Input Signal.
$\overline{RxC}$	23	28	I *	Received Data Clock. This signal provides timing information to the internal shifting logic depending upon the mode of data transfer. For NRZ data, the state of the RxD pin is sampled on the HIGH to LOW clock transition.

*See D.C. Characteristics.

Table 1. 82586 Pin Description (Continued)

Symbol	48 Pin DIP Pin No.	68 Pin PLCC Pin No.	Type Level	Name and Function
$\overline{RTS}$	28	21	0 TTL	Request To Send signal. When LOW, notifies an external interface that the 82586 has data to transmit. It is forced HIGH after a Reset and while the Transmit Serial Unit is not sending data.
$\overline{CTS}$	29	20	I TTL	Active LOW Clear To Send input enables the 82586 transmitter to actually send data. It is normally used as an interface handshake to $\overline{RTS}$. This signal going inactive stops transmission. It is internally synchronized. If $\overline{CTS}$ goes inactive, meeting the setup time to $\overline{TxC}$ negative edge, transmission is stopped and $\overline{RTS}$ goes inactive within, at most, two TxC cycles.
$\overline{CRS}$	31	18	I TTL	Active LOW Carrier Sense input used to notify the 82586 that there is traffic on the serial link. It is used only if the 82586 is configured for external Carrier Sense. When so configured, external circuitry is required for detecting serial link traffic. It is internally synchronized. To be accepted, the signal must stay active for at least two serial clock cycles.
$\overline{CDT}$	30	19	I TTL	Active LOW Collision Detect input is used to notify the 82586 that a collision has occurred. It is used only if the 82586 is configured for external Collision Detect. External circuitry is required for detecting the collision. It is internally synchronized. To be accepted, the signal must stay active for at least two serial clock cycles. During transmission, the 82586 is able to recognize a collision one bit time after preamble transmission has begun.
INT	38	6	0 TTL	Active HIGH Interrupt request signal.
CLK	32	15	I MOS	The system clock input from the 80186 or another symmetrical clock generator.
MN/$\overline{MX}$	33	14	I TTL	When HIGH, MN/$\overline{MX}$ selects $\overline{RD}$, $\overline{WR}$, ALE $\overline{DEN}$, DT/$\overline{R}$ (Minimum Mode). When LOW, MN/$\overline{MX}$ selects A22, A23, READY, $\overline{S0}$, $\overline{S1}$ (Maximum Mode). Note: This pin should be static during 82586 operation.
AD0–AD15	6–11, 13–22	29–33, 36–40, 45, 48, 49, 50, 53, 54	I/O TTL	These lines form the time multiplexed memory address (t1) and data (t2, t3, tW, t4) bus. When operating with an 8-bit bus, the high byte will output the address only during T1. AD0–AD15 are floated after a RESET or when the bus is not acquired.
A16–A18 A20–A23	1, 3–5 45–47	55–57, 59, 63–65	0 TTL	These lines constitute 7 out of 8 most significant address bits for memory operation. They switch during t1 and stay valid during the entire memory cycle. The lines are floated after RESET or when the bus is not acquired. Address lines A22 and A23 are not available for use in minimum mode.
A19/S6	2	58	0 TTL	During t1 it forms line 19 of the memory address. During t2 through t4 it is used as a status indicating that this is a Master peripheral cycle, and is HIGH. Its timing is identical to that of AD0–AD15 during write operation.

Table 1. 82586 Pin Description (Continued)

Symbol	48 Pin DIP Pin No.	68 Pin PLCC Pin No.	Type Level	Name and Function
HOLD	43	67	0 TTL	HOLD is an active HIGH signal used by the 82586 to request local bus mastership at the end of the current CPU bus transfer cycle, or at the end of the current DMA burst transfer cycle. In normal operation, HOLD goes inactive before HLDA. The 82586 can be forced off the bus by HLDA going inactive. In this case, HOLD goes inactive within four clock cycles in word mode and eight clock cycles in byte mode.
HLDA	42	68	I TTL	HLDA is an active HIGH Hold Acknowledge signal indicating that the CPU has received the HOLD request and that bus control has been relinquished to the 82586. It is internally synchronized. After HOLD is detected as LOW, the processor drives HLDA LOW. Note, CONNECTING V_{CC} TO HLDA IS NOT ALLOWED because it will cause a deadlock. Users wanting to give permanent bus access to the 82586 should connect HLDA with HOLD.
CA	35	12	I TTL	The CA pin is a Channel Attention input used by the CPU to initiate the 82586 execution of memory resident Command Blocks. The CA signal is synchronized internally. The signal must be HIGH for at least one system clock period. It is latched internally on HIGH to LOW edge and then detected by the 82586.
$\overline{BHE}$	44	66	0 TTL	The Bus High Enable signal ($\overline{BHE}$) is used to enable data onto the most significant half of the data bus. Its timing is identical to that of A16–A23. With a 16-bit bus it is LOW and with an 8-bit bus it is HIGH. Note: after RESET, the 82586 is configured to 8-bit bus.
READY	39	5	I TTL	This active HIGH signal is the acknowledgement from the addressed memory that the transfer cycle can be completed. While LOW, it causes wait states to be inserted. This signal must be externally synchronized with the system clock. The Ready signal internal to the 82586 is a logical OR between READY and SRDY/ARDY.
ARDY/SRDY	37	7	I TTL	This active HIGH signal performs the same function as READY. If it is programmed at configure time to SRDY, it is identical to READY. If it is programmed to ARDY, the positive edge of the Ready signal is internally synchronized. Note, the negative edge must still meet setup and hold time specifications, when in ARDY mode. The ARDY signal must be active for at least one system clock HIGH period for proper strobing. The Ready signal internal to the 82586 is a logical OR between READY (in Maximum Mode only) and SRDY/ARDY. Note that following RESET, this pin assumes ARDY mode.

Table 1. 82586 Pin Description (Continued)

Symbol	48 Pin DIP Pin No.	68 Pin PLCC Pin No.	Type Level	Name and Function
$\overline{S0}$, $\overline{S1}$	40,41	4, 3	0 TTL	Maximum mode only. These status pins define the type of DMA transfer during the current memory cycle. They are encoded as follows: $\overline{S1}$ $\overline{S0}$ 0 0 Not Used 0 1 Read Memory 1 0 Write Memory 1 1 Passive Status is active from the middle of t4 to the end of t2. They return to the passive state during t3 or during tW when READY or ARDY is HIGH. These signals can be used by the 8288 Bus Controller to generate all memory control and timing signals.* Any change from the passive state, signals the 8288 to start the next t1 to t4 bus cycle. These pins are pulled HIGH and floated after a system RESET and when the bus is not acquired.
$\overline{RD}$	46	64	0 TTL	Used in minimum mode only. The read strobe indicates that the 82586 is performing a memory read cycle. $\overline{RD}$ is active LOW during t2, t3 and tW of any read cycle. This signal is pulled HIGH and floated after a RESET and when the bus is not acquired.
$\overline{WR}$	45	65	0 TTL	Used in minimum mode only. The write strobe indicates that the 82586 is performing a write memory cycle. $\overline{WR}$ is active LOW during t2, t3 and tW of any write cycle. It is pulled HIGH and floats after RESET and when the bus is not acquired.
ALE	39	5	0 TTL	Used in minimum mode only. Address Latch Enable is provided by the 82586 to latch the address into the 8282/8283 address latch. It is a HIGH pulse, during t1 ('clock low') of any bus cycle. Note that ALE is never floated.
$\overline{DEN}$	40	4	0 TTL	Used in minimum mode only. Data ENable is provided as output enable for the 8286/8287 transceivers in a stand-alone (no 8288) system. $\overline{DEN}$ is active LOW during each memory access. For a read cycle, it is active from the middle of t2 until the beginning of t4. For a write cycle, it is active from the beginning of t2 until the middle of t4. It is pulled HIGH and floats after a system RESET or when the bus is not acquired.
DT/$\overline{R}$	41	3	0 TTL	Used in minimum mode only. DT/$\overline{R}$ is used in non-8288 systems using an 8286/8287 data bus transceiver. It controls the direction of data flow through the Transceiver. Logically, DT/$\overline{R}$ is equivalent to $\overline{S1}$. It becomes valid in the t4 preceding a bus cycle and remains valid until the final t4 of the cycle. This signal is pulled HIGH and floated after a RESET or when the bus is not acquired.

NOTE:
*8288 does not support 10 MHz operation.

82586/HOST CPU INTERACTION

Communication between the 82586 and the host is carried out via shared memory. The 82586's on-chip DMA capability allows autonomous transfer of data blocks (buffers, frames) and relieves the CPU of byte transfer overhead. The 82586 is optimized to interface the iAPX 186, but due to the small number of hardware signals between the 82586 and the CPU, the 82586 can operate easily with other processors. The 82586/host interaction is explained separately in terms of the logical interface and the hardware bus interface.

The 82586 consists of two independent units: Command Unit (CU) and Receive Unit (RU). The CU executes commands from shared memory. The RU handles all activities related to frame reception. The CU and RU enable the 82586 to engage in the two types of activities simultaneously: the CU may be fetching and executing commands out of memory, and the RU may be storing received frames in memory. CPU intervention is only required after the CU executes a sequence of commands or the RU stores a sequence of frames.

The only hardware signals that connect the CPU and the 82586 are INTERRUPT and CHANNEL ATTENTION (see Figure 3). Interrupt is used by the 82586 to draw the CPU's attention to a change in the contents of the SCB. Channel Attention is used by the CPU to draw the 82586's attention.

82586 SYSTEM MEMORY STRUCTURE

The Shared Memory structure consists of four parts: Initialization Root, System Control Block (SCB), Command List, and Receive Frame Area (RFA) (see Figure 4).

The Initialization Root is at a predetermined location in the memory space, (0FFFFF6H), known to both the host CPU and the 82586. The root is accessed at initialization and points to the System Control Block.

The System Control Block (SCB) functions as a bidirectional mail drop between the host CPU, CU and RU. It is the central element through which the CPU and the 82586 exchange control and status information. The SCB consists of two parts, the first of which entails instructions from the CPU to the 82586. These include: control of the CU and RU (START, ABORT, SUSPEND, RESUME), a pointer to the list of commands for the CU, a pointer to the receive frame area, and a set of Interrupt acknowledge bits. The second entails status information keyed by the 82586 to the CPU, including: state of the CU and RU (e.g. IDLE, ACTIVE READY, SUSPENDED, NO RECEIVE RESOURCES), interrupts bits (command completed, frame received, CU not ready, RU not ready), and statistics (see Figure 4).

The Command List serves as a program for the CU. Individual commands are placed in memory units called a Command Block, or CB. CB's contain command specific parameters and command specific statuses. Specifically, these high level commands are called Action Commands (e.g. Transmit, Configure).

A specific command, Transmit, causes transmission of a frame by the 82586. The Transmit command block includes Destination Address, Length Field, and a pointer to a list of linked buffers that holds the frame to be constructed from several buffers scattered in memory. The Command Unit performs with-

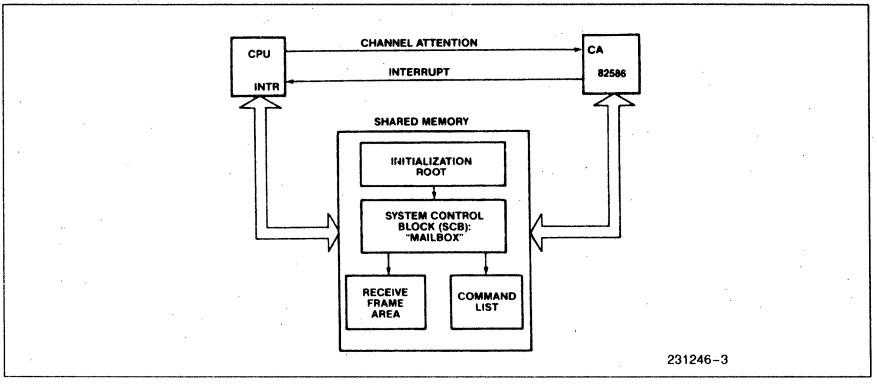


Figure 3. 82586/Host CPU Interaction

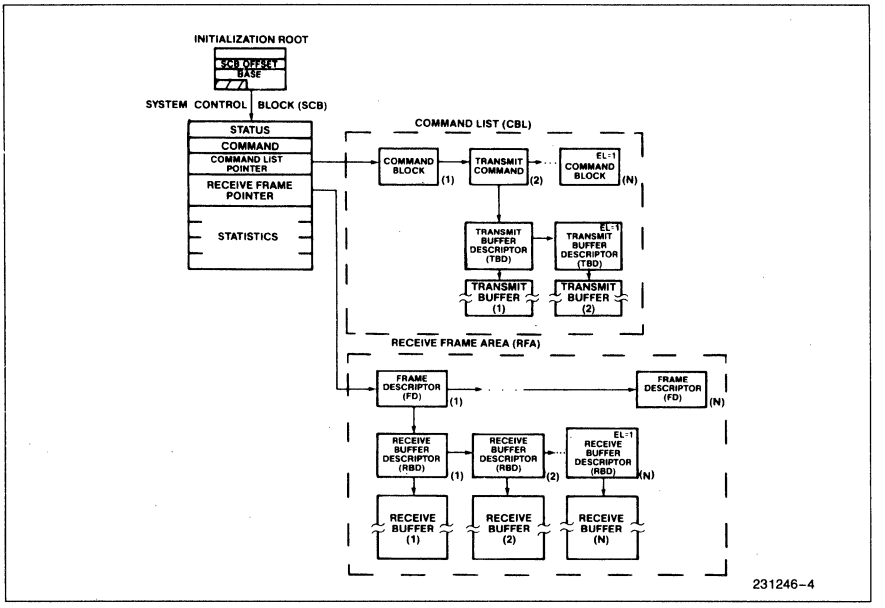


Figure 4. 82586 Shared Memory Structure

out the CPU intervention, the DMA of each buffer and the prefetching of references to new buffers in parallel. The CPU is notified only after successful transmission or retransmission.

The Receive Frame Area is a list of Free Frame Descriptors (Descriptors not yet used) and a list of buffers prepared by the user. It is conceptually distinct from the Command List. Frames arrive without being solicited by the 82586. The 82586 must be prepared to receive them even if it is engaged in other activities and to store them in the Free Frame Area. The Receive Unit fills the buffers upon frame reception and reformats the Free Buffer List into received frame structures. The frame structure is virtually identical to the format of the frame to be transmitted. The first frame descriptor is referenced by SCB. A Frame Descriptor and the associated Buffer Descriptor wasted upon receiving a Bad Frame (CRC or Alignment errored, Receive DMA overrun errored, or Collision fragmented frame) are automatically reclaimed and returned to the Free Buffer List, unless the chip is configured to Save Bad Frames.

Receive buffer chaining (i.e. storing incoming frames in a linked list of buffers) improves memory utilization significantly. Without buffer chaining, the user must allocate consecutive blocks of the maximum frame size (1518 bytes in Ethernet) for each frame. Taking into account that a typical frame size may be about 100 bytes, this practice is very inefficient. With buffer chaining, the user can allocate small buffers and the 82586 uses only as many as needed.

In the past, the drawback of buffer chaining was the CPU processing overhead and the time involved in the buffer switching (especially at 10 Mb/s). The 82586 overcomes this drawback by performing buffer management on its own for both transmission and reception (completely transparent to the user).

The 82586 has a 22-bit memory address range in minimum mode and 24-bit memory address range in maximum mode. All memory structures, the System Control Block, Command List, Receive Descriptor List, and all buffer descriptors must reside within one 64K-byte memory segment. The Data Buffers can be located anywhere in the memory space.

TRANSMITTING FRAMES

The 82586 executes high level action commands from the Command List in external memory. Action commands are fetched and executed in parallel with the host CPU's operation, thereby significantly improving system performance. The general action commands format is shown in Figure 5.

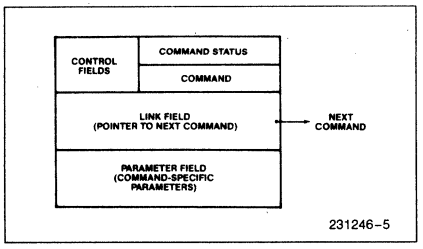


Figure 5. Action Command Format

Message transmission is accomplished by using the Transmit command. A single Transmit command contains, as part of the command-specific parameters, the destination address and length field for the transmitted frame along with a pointer to a buffer area in memory containing the data portion of the frame. (See Figure 15.) The data field is contained in a memory data structure consisting of a Buffer Descriptor (BD) and Data Buffer (or a linked list of buffer descriptors and buffers) as shown in Figure 6. The BD contains a Link Field which points to the next BD on the list and a 24-bit address pointing to the Data Buffer itself. The length of the Data Buffer is specified by the Actual Count field of the BD.

Using the BD's and Data Buffers, multiple Data Buffers can be 'chained' together. Thus, a frame with a long Data Field can be transmitted using multiple (shorter) Data buffers chained together. This chaining technique allows the system designer to develop efficient buffer management policies.

The 82586 automatically generates the preamble (alternating 1's and 0's) and start frame delimiter, fetches the destination address and length field from the Transmit command, inserts its unique address as the source address, fetches the data field from

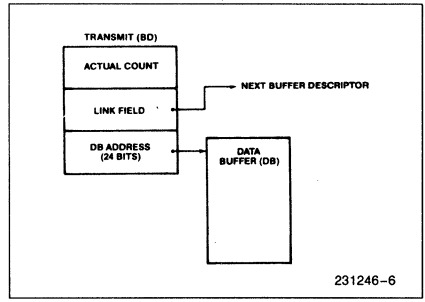


Figure 6. Data Buffer Descriptor and Data Buffer Structure

buffers pointed to by the Transmit command, and computes and appends the CRC at the end of the frame. See Figure 7.

The 82586 can be configured to generate either the Ethernet or HDLC start and end frame delimiters. In the Ethernet mode, the start frame delimiter is 10101011 and the end frame delimiter indicated by the lack of a signal after transmitting the last bit of the frame check sequence field. When in the HDLC mode, the 82586 will generate the 01111110 'flag' for the start and end frame delimiters and perform the standard 'bit stuffing/stripping'. In addition, the 82586 will optionally pad frames that are shorter than the specified minimum frame length by appending the appropriate number of flags to the end of the frame.

In the event of a collision (or collisions), the 82586 manages the entire jam, random wait and retry process, reinitializing DMA pointers without CPU intervention. Multiple frames can be sent by linking the appropriate number of Transmit commands together. This is particularly useful when transmitting a message that is larger than the maximum frame size (1518 bytes for Ethernet).

RECEIVING FRAMES

In order to minimize CPU overhead, the 82586 is designed to receive frames without CPU supervision. The host CPU first sets aside an adequate

PREAMBLE	START FRAME DELIMITER	DEST ADDR	SOURCE ADDR	LENGTH FIELD	DATA FIELD	FRAME CHECK SEQUENCE	END FRAME DELIMITER

Figure 7. Frame Format

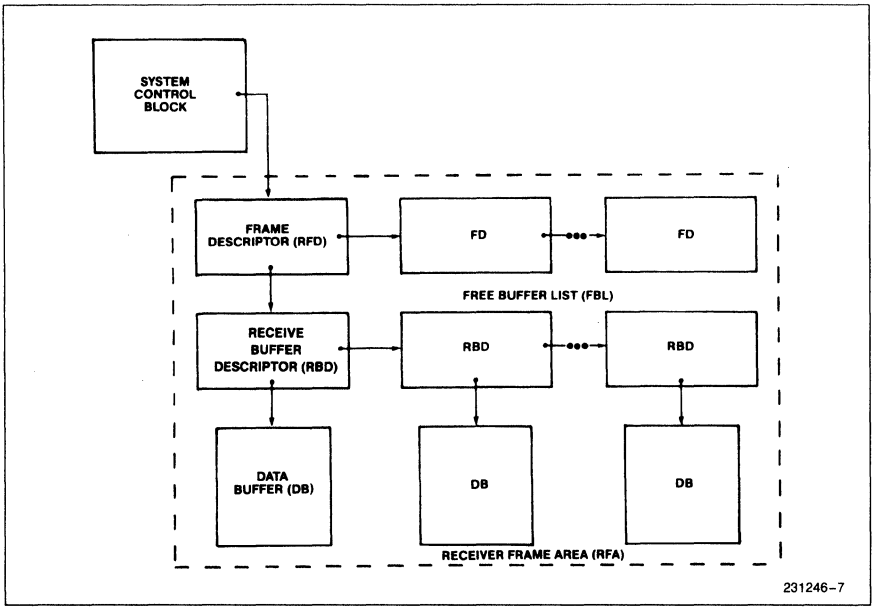


Figure 8. Receive Frame Area Diagram

amount of receive buffer space and then enables the 82586's Receive Unit. Once enabled, the RU 'watches' for any of its frames which it automatically stores in the Receive Frame Area (RFA). The RFA consists of a Receive Descriptor List (RDL) and a list of free buffers called the Free Buffer List (FBL) as shown in Figure 8. The individual Receive Frame Descriptors that make up the RDL are used by the 82586 to store the destination and source address, length field and status of each frame that is received. (Figure 9.)

The 82586, once enabled, checks each passing frame for an address match. The 82586 will recognize its own unique address, one or more multicast addresses or the broadcast address. If a match occurs, it stores the destination and source address and length field in the next available RFD. It then begins filling the next free Data Buffer on the FBL (which is pointed to by the current RFD) with the data portion of the incoming frame. As one DB is filled, the 82586 automatically fetches the next DB on the FBL until the entire frame is received. This buffer chaining technique is particularly memory efficient because it allows the system designer to set aside buffers that fit a frame size that may be much shorter than the maximum allowable frame.

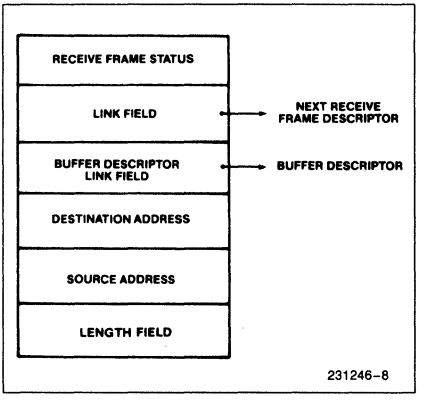


Figure 9. Receive Frame Descriptor

Once the entire frame is received without error, the 82586 performs the following housekeeping tasks:

- Updates the Actual Count field of the last Buffer Descriptor used to hold the frame just received with the number of bytes stored in its associated Data Buffer.

- Fetches the address of the next free Receive Frame Descriptor.
- Writes the address of the next free Buffer Descriptor into the next free Receive Frame Descriptor.
- Posts a 'Frame Received' interrupt status bit in the SCB.
- Interrupts the CPU.

In the event of a frame error, such as a CRC error, the 82586 automatically reinitializes its DMA pointers and reclaims any data buffers containing the bad frame. As long as Receive Frame Descriptors and data buffers are available, the 82586 will continue to receive frames without further CPU help.

82586 NETWORK MANAGEMENT AND DIAGNOSTIC FUNCTIONS

The behavior of data communication networks is typically very complex due to their distributed and asynchronous nature. It is particularly difficult to pinpoint a failure when it occurs. The 82586 was designed in anticipation of these problems and includes a set of features for improving reliability and testability.

The 82586 reports on the following events after each frame transmitted:

- Transmission successful.
- Transmission unsuccessful; lost Carrier Sense.
- Transmission unsuccessful; lost Clear-to-Send.
- Transmission unsuccessful; DMA underrun because the system bus did not keep up with the transmission.
- Transmission unsuccessful; number of collisions exceeded the maximum allowed.

The 82586 checks each incoming frame and reports on the following errors, (if configured to 'Save Bad Frame'):

- CRC error: incorrect CRC in a well aligned frame.
- Alignment error: incorrect CRC in a misaligned frame.
- Frame too short: the frame is shorter than the configured value for minimum frame length.
- Overrun: the frame was not completely placed in memory because the system bus did not keep up with incoming data.
- Out of buffers: no memory resources to store the frame, so part of the frame was discarded.

NETWORK PLANNING AND MAINTENANCE

To perform proper planning, operation, and maintenance of a communication network, the network management entity must accumulate information on network behavior. The 82586 provides a rich set of network-wide diagnostics that can serve as the basis for a network management entity.

Network Activity information is provided in the status of each frame transmitted. The activity indicators are:

- Number of collisions: number of collisions the 82586 experienced in attempting to transmit this frame.
- Deferred transmission: indicates if the 82586 had to defer to traffic on the link during the first transmission attempt.

Statistics registers are updated after each received frame that passes the address filtering, and is longer than the Minimum Frame Length configuration parameter.

- CRC errors: number of frames that experienced a CRC error and were properly aligned.
- Alignment errors: number of frames that experienced a CRC error and were misaligned.
- No-resources: number of correct frames lost due to lack of memory resources.
- Overrun errors: number of frame sequences lost due to DMA overrun.

The 82586 can be configured to Promiscuous Mode. In this mode it captures all frames transmitted on the Network without checking the Destination Address. This is useful in implementing a monitoring station to capture all frames for analysis.

The 82586 is capable of determining if there is a short or open circuit anywhere in the Network using the built in Time Domain Reflectometer (TDR) mechanism.

STATION DIAGNOSTICS

The chip can be configured to External Loopback. The transmitter to receiver interconnection can be placed anywhere between the 82586 and the link to locate faults, for example: the 82586 output pins, the Serial Interface Unit, the Transceiver cable, or in the Transceiver.

The 82586 has a mechanism recognizing the transceiver 'heart beat' signal for verifying the correct operation of the Transceiver's collision detection circuitry.

82586 SELF TESTING

The 82586 can be configured to Internal Loopback. It disconnects itself from the Serial Interface Unit, and any frame transmitted is received immediately. The 82586 connects the Transmit Data to the Receive Data signal and the Transmit Clock to the Receive Clock.

The Dump Command causes the chip to write over 100 bytes of its internal registers to memory.

The Diagnose command checks the exponential Backoff random number generator internal to the 82586.

CONTROLLING THE 82586

The CPU controls operation of the 82586's Command Unit (CU) and Receive Unit (RU) of the 82586 via the System Control Block.

THE COMMAND UNIT (CU)

The Command Unit is the logical unit that executes Action Commands from a list of commands very similar to a CPU program. A Command Block (CB) is associated with each Action Command.

The CU can be modeled as a logical machine that takes, at any given time, one of the following states:

- IDLE—CU is not executing a command and is not associated with a CB on the list. This is the initial state.
- SUSPENDED—CU is not executing a command but (different from IDLE) is associated with a CB on the list.
- ACTIVE—CU is currently executing an Action Command, and points to its CB.

The CPU may affect the CU operation in two ways: issuing a CU control Command or setting bits in the COMMAND word of the Action Command.

THE RECEIVE UNIT (RU)

The Receive Unit is the logical unit that receives frames and stores them in memory.

The RU is modeled as a logical machine that takes, at any given time, one of the following states:

- IDLE—RU has no memory resources and is discarding incoming frames. This is the initial RU state.
- NO-RESOURCES—RU has no memory resources and is discarding incoming frames. This state differs from the IDLE state in that RU accumulates statistics on the number of frames it had to discard.
- SUSPENDED—RU has free memory resources to store incoming frames but discard them anyway.

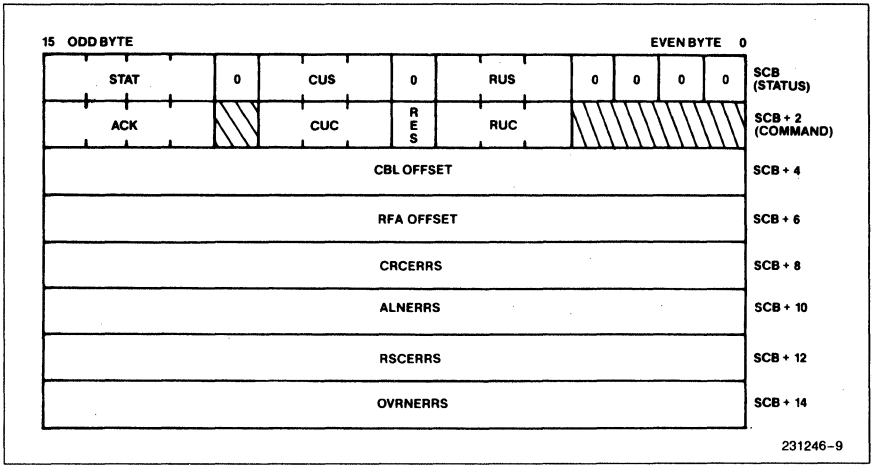


Figure 10. System Control Block (SCB) Format

- READY—RU has free memory resources and stores incoming frames.

The CPU may affect RU operation in three ways: issuing an RU Control Command, setting bits in Frame Descriptor, FD, COMMAND word of the frame currently being received, or setting EL bit of Buffer Descriptor, BD, of the buffer currently being filled.

SYSTEM CONTROL BLOCK (SCB)

The System Control Block is the communication mail-box between the 82586 and the host CPU. The SCB format is shown in Figure 10.

The host CPU issues Control Commands to the 82586 via the SCB. These commands may appear at any time during routine operation, as determined by the host CPU. After the required Control Command is setup, the CPU sends a CA signal to the 82586.

SCB is also used by the 82586 to return status information to the host CPU. After inserting the required status bits into SCB, the 82586 issues an Interrupt to the CPU.

The format is as follows:

STATUS word: Indicates the status of the 82586. This word is modified only by the 82586. Defined bits are:

CX	(Bit 15)	• A command in the CBL having its 'I' (interrupt) bit set has been executed.
FR	(Bit 14)	• A frame has been received.
CNR	(Bit 13)	• The Command Unit left the Active state.
RNR	(Bit 12)	• The Receive Unit left the Ready state.
CUS	(Bits 8–10)	• (3 bits) this field contains the status of the Command Unit. Valid values are: 0 — Idle 1 — Suspended 2 — Active 3–7 — Not Used
RUS	(Bits 4—6)	• (3 bits) this field contains the status of the Receive Unit. Valid values are: 0 — Idle 1 — Suspended 2 — No Resources 3 — Not Used 4 — Ready 5–7 — Not Used

COMMAND word: Specifies the action to be performed as a result of the CA. This word is set by the CPU and cleared by the 82586. Defined bits are:

ACK-CX	(Bit 15)	• Acknowledges the command executed event.
ACK-FR	(Bit 14)	• Acknowledges the frame received event.
ACK-CNA	(Bit 13)	• Acknowledes that the Command Unit became not ready.
ACK-RNR	(Bit 12)	• Acknowledges that the Receive Unit became not ready.
CUC	(Bits 8–10)	• (3 bits) this field contains the command to the Command Unit.
	0	• NOP (doesn't affect current state of the unit).
	1	• Start execution of the first command on the CBL. If a command is in execution, then complete it before starting the new CBL. The beginning of the CBL is in CBL OFFSET.
	2	• Resume the operation of the command unit by executing the next command. This operation assumes that the command unit has been previously suspended.
	3	• Suspend execution of commands on CBL after current command is complete.
	4	• Abort execution of commands immediately.
	5–7	• Reserved, illegal for use.
RUC	(Bits 4–6)	• (3 bits) This field contains the command to the receive unit. Valid values are:
	0	• NCP (does not alter current state of unit).
	1	• Start reception of frames. If a frame is being received, then complete reception before starting. The beginning of the RFA is contained in the RFA OFFSET.
	2	• Resume frame receiving (only when in suspended state.)
	3	• Suspend frame receiving. If a frame is being received, then complete its reception before suspending.
	4	• Abort receiver operation immediately.
	5–7	• Reserved, illegal for use.
RESET	(Bit 7)	• Reset chip (logically the same as hardware RESET).

CBL-OFFSET:

Gives the 16-bit offset address of the first command (Action Command) in the command list to be executed following CU-START. Thus, the 82586 reads this word only if the CUC field contained a CU-START Control Command.

RFA-OFFSET:

Points to the first Receive Frame Descriptor in the Receive Frame Area.

CRCERRS:

CRC Errors - contains the number of properly aligned frames received with a CRC error.

ALNERRS:

Alignment Errors - contains the number of misaligned frames received with a CRC error.

RSCERRS:

Resource Errors - records the number of correct incoming frames discarded due to lack of memory resources (buffer space or received frame descriptors).

OVRNERRS:

Overrun Errors - counts the number of received frame sequences lost because the memory bus was not available in time to transfer them.

ACTION COMMANDS

The 82586 executes a 'program' that is made up of action commands in the Command List. As shown in Figure 5, each command contains the command field, status and control fields, link to the next action command in the CL, and any command-specific parameters. This command format is called the Command Block.

The 82586 has a repertoire of 8 commands:

NOP
Setup Individual Address
Configure
Setup Multicast Address
Transmit
TDR
Diagnose
Dump

NOP

This command results in no action by the 82586, except as performed in normal command processing. It is present to aid in Command List manipulation.

NOP command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0-2)	• NOP = 0

LINK OFFSET: Address of next Command Block

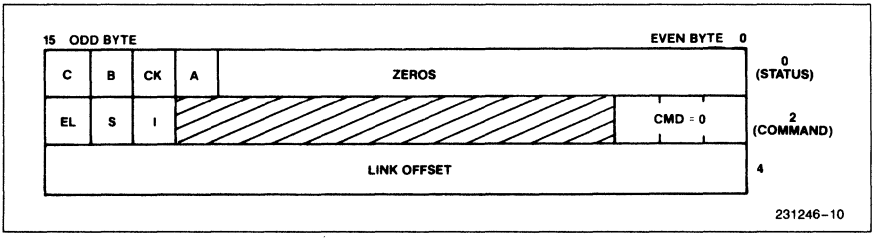


Figure 11. The NOP Command Block

IA-SETUP

This command loads the 82586 with the Individual Address. This address is used by the 82586 for recognition of Destination Address during reception and insertion of Source Address during transmission.

The IA-SETUP command includes the following fields:

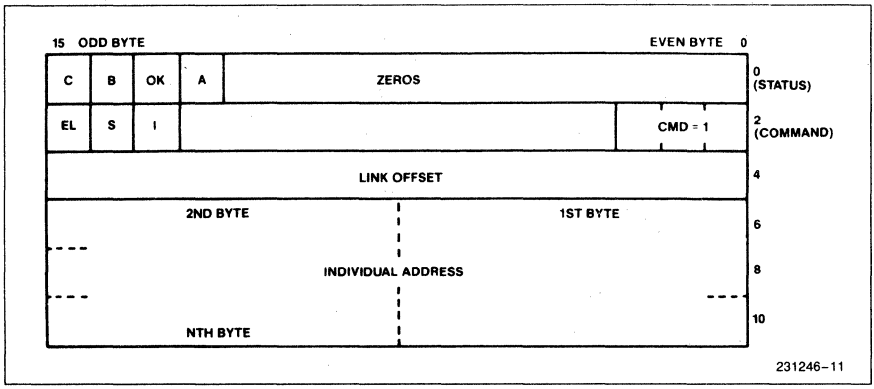


Figure 12. The IA-SETUP Command Block

STATUS word (written by 82586).

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
A	(Bit 12)	• Command Aborted

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• IA-SETUP = 1

LINK OFFSET: Address of next Command Block

INDIVIDUAL ADDRESS: Individual Address parameter

The least significant bit of the Individual Address parameter must be zero for IEEE 802.3/Ethernet. However, no enforcement of 0 is provided by the 82586. Thus, an Individual Address with least significant bit 1, is possible.

CONFIGURE

The CONFIGURE command is used to update the 82586 operating parameters.

The CONFIGURE command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
A	(Bit 12)	• Command Aborted

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• Configure = 2

LINK OFFSET: Address of next Command Block

Byte 6–7:

BYTE CNT	(Bits 0–3)	• Byte Count, Number of bytes including this one, holding the parameters to be configured. A number smaller than 4 is interpreted as 4. A number greater than 12 is interpreted as 12.

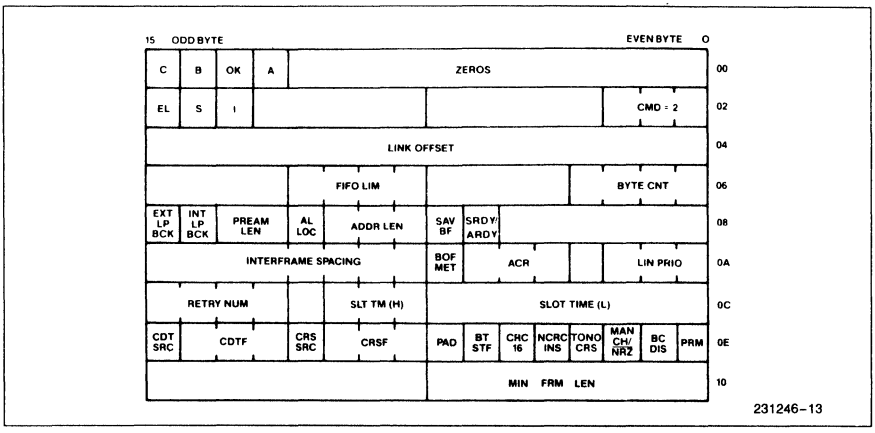


Figure 13. The CONFIGURE Command Block

FIFO-LIM	(Bits 8–11)	• Value of FIFO Threshold.

Byte 8–9:

SRDY/ARDY	(Bit 6)	
	0	• SRDY/ARDY pin operates as ARDY (internal synchronization).
	1	• SRDY/ARDY pin operates as SRDY (external synchronization).
SAV-BF	(Bit 7)	
	0	• Received bad frames are not saved in memory.
	1	• Received bad frames are saved in memory.
ADD-LEN	(Bits 8–10)	• Number of address byes. NOTE: 7 is interpreted as 0.
AL-LOC	(Bit 11)	
	0	• Address and Length Fields separated from data and associated with Transmit Command Block or Receive Frame Descriptor. For transmitted Frame, Source Address is inserted by the 82586.
	1	• Address and Length Fields are part of the Transmit/Receive data buffers, including Source Address (which is not inserted by the 82586).
PREAM-LEN	(Bits 12–13)	• Preamble Length including Beginning of Frame indicator: 00 - 2 bytes 01 - 4 bytes 10 - 8 bytes 11 - 16 bytes
INT-LPBCK	(Bit 14)	• Internal Loopback
EXT-LPBCK	(Bit 15)	• External Loopback. NOTE: Bits 14 and 15 configured to 1, cause Internal Loopback.

Byte 10–11:

LIN-PRIO	(Bits 0–2)	• Linear Priority
ACR	(Bits 4–6)	• Accelerated Contention Resolution (Exponential Priority)
BOF-MET	(Bit 7)	• Exponential Backoff Method 0 - IEEE 802.3/Ethernet 1 - Alternate Method

INTER FRAME SPACING	(Bits 8–15)	• Number indicating the Interframe Spacing in TxC period units.

Byte 12–13:

SLOT-TIME (L)	(Bits 0–7)	• Slot Time Number, Low Byte
SLT-TM (H)	(Bits 8–10)	• Slot Time Number, High Bits
RETRY-NUM	(Bits 12–15)	• Maximum Number of Transmission Retries on Collisions

Byte 14–15:

PRM	(Bit 0)	• Promiscuous Mode
BC-DIS	(Bit 1)	• Broadcast Disable
MANCH/$\overline{NRZ}$	(Bit 2)	• Manchester or NRZ • Encoding/Decoding
	0	• NRZ
	1	• Manchester
TONO-CRS	(Bit 3)	• Transmit on No Carrier Sense
	0	• Cease Transmission if $\overline{CRS}$ Goes Inactive During Frame Transmission
	1	• Continue Transmission Even if no Carrier Sense
NCRC-INS	(Bit 4)	• No CRC Insertion
CRC-16	(Bit 5)	• CRC Type:
	0	• 32 bit Autodin II CRC Polynomial
	1	• 16 bit CCITT CRC Polynomial
BT-STF	(Bit 6)	• Bitstuffing:
	0	• End of Carrier Mode (Ethernet)
	1	• HDLC like Bitstuffing Mode
PAD	(Bit 7)	• Padding
	0	• No Padding
	1	• Perform Padding by Transmitting Flags for Remainder of Slot Time
CRSF	(Bits 8–9)	• Carrier Sense Filter in Bit Times
CRS-SRC	(Bit 11)	• Carrier Sense Source
	0	• External
	1	• Internal
CDTF	(Bits 12–14	• Collision Detect Filter in Bit Times
CDT-SRC	(Bit 15)	• Collision Detect Source
	0	• External
	1	• Internal

Byte 16:

MIN-FRM-	(Bits 0–7)	• Minimum Number of Bytes in a Frame

CONFIGURATION DEFAULTS

The default values of the configuration parameters are compatible with the IEEE 802.3/Ethernet Standards. RESET configures the 82586 according to the defaults shown in Table 2.

Table 2. 82586 Default Values

Preamble Length (Bytes)	=	8
Address Length (Bytes)	=	6
Broadcast Disable	=	0
CRC-16/CRC-32	=	0
No CRC Insertion	=	0
Bitstuffing/EOC	=	0
Padding	=	0
Min-Frame-Length (Bytes)	=	64
Interframe Spacing (Bits)	=	96
Slot Time (Bits)	=	512
Number of Retries	=	15
Linear Priority	=	0
Accelerated Contention Resolution	=	0
Exponential Backoff Method	=	0
Manchester/$\overline{NRZ}$	=	0
Internal CRS	=	0
CRS Filter	=	0
Internal CDT	=	0
CDT Filter	=	0
Transmit On No CRS	=	0
FIFO THRESHOLD	=	8
SRDY/$\overline{ARDY}$	=	0
Save Bad Frame	=	0
Address/Length Location	=	0
INT Loopback	=	0
EXT Loopback	=	0
Promiscuous Mode	=	0

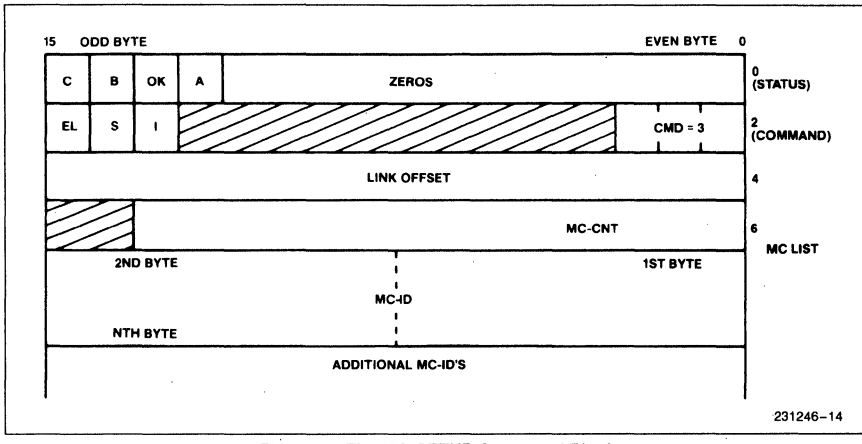


Figure 14. The MC-SETUP Command Block

MC-SETUP

This command sets up the 82586 with a set of Multicast Addresses. Subsequently, incoming frames with Destination Addresses from this set are accepted.

The MC-SETUP command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
A	(Bit 12)	• Command Aborted

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• MC-SETUP = 3

LINK OFFSET: Address of next Command Block

MC-CNT: A 14-bit field indicating the number of bytes in the MC-LIST field. MC-CNT is truncated to the nearest multiple of Address Length (in bytes). Issuing a MC-SETUP command with MC-CNT = 0 disables reception of any incoming frame with a Multicast Address.

MC-LIST: A list of Multicast Addresses to be accepted by the 82586. Note that the most significant byte of an address is followed immediately by the least significant byte of the next address. Note also that the least significant bit of each Multicast Address in the set must be a one.

The Transmit-Byte-Machine maintains a 64-bit HASH table used for checking Multicast Addresses during reception.

An incoming frame is accepted if it has a Destination Address whose least significant bit is a one, and after hashing points to a bit in the HASH table whose value is one. The hash function is selecting bits 2 to 7 of the CRC register. RESET causes the HASH table to become all zeros.

After the Transmit-Byte-Machine reads a MC-SETUP command from TX-FIFO, it clears the HASH table and reads the bytes in groups whose length is determined by the ADDRESS length. Each group is hashed using CRC logic and the bit in the HASH table to which bits 2–7 of the CRC register point is set to one. A group that is not complete has no effect on the HASH table. Transmit-Byte-Machine notifies CU after completion.

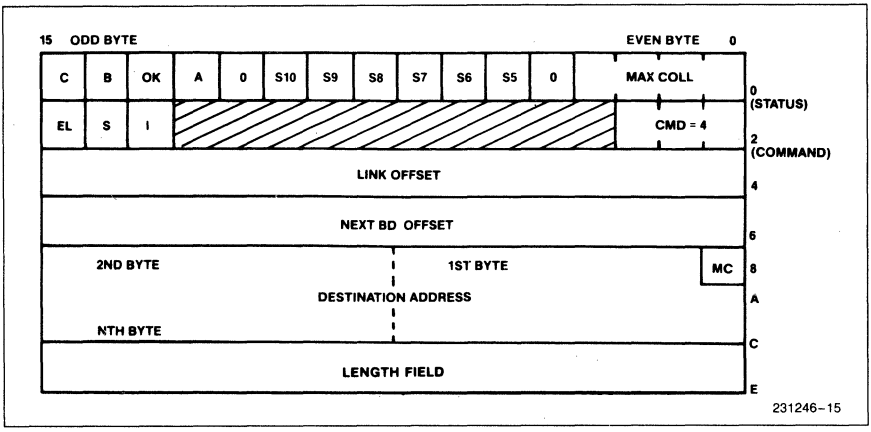


Figure 15. The Transmit Command Block

TRANSMIT

The TRANSMIT command causes transmission (and if necessary retransmission) of a frame.

TRANSMIT CB includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
A	(Bit 12)	• Command Aborted
S10	(Bit 10)	• No Carrier Sense signal during transmission (between beginning of Destination Address and end of Frame Check Sequence).
S9	(Bit 9)	• Transmission unsuccessful (stopped) due to loss of Clear-to-Send signal.
S8	(Bit 8)	• Transmission unsuccessful (stopped) due to DMA underrun, (i.e. data not supplied from the sytem for transmission).
S7	(Bit 7)	• Transmission had to Defer to traffic on the link.
S6	(Bit 6)	• Heart Beat, indicates that during Interframe Spacing period after the previous transmission, a pulse was detected on the Collision Detect pin.
S5	(Bit 5)	• Transmission attempt stopped due to number of collisions exceeding the maximum number of retries.
MAX-COLL	(Bits 3–0)	• Number of Collisions experienced by this frame. S5 = 1 and MAX-COLL = 0 indicates that there were 16 collisions.
COMMAND word:		
EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• TRANSMIT = 4

LINK OFFSET: Address of next Command Block

TBD OFFSET: Address of list of buffers holding the information field. TBD-OFFSET = 0FFFFH indicates that there is no Information field.

DESTINATION ADDRESS: Destination Address of the frame.

LENGTH FIELD: Length field of the frame.

STATUS word:

EOF		• Indicates that this is the Buffer Descriptor of the last buffer of this frame's Information Field.
ACT-COUNT	(Bits 0–13)	• Actual number of data bytes in buffer (can be even or odd).

NEXT BD OFFSET: points to next Buffer Descriptor in list. If EOF is set, this field is meaningless.

BUFFER ADDRESS: 24-bit absolute address of buffer.

TIME DOMAIN REFLECTOMETER - TDR

This command performs a Time Domain Reflectometer test on the serial link. By performing the command, the user is able to identify shorts or opens and their location. Along with transmission of 'All Ones,' the 82586 triggers an internal timer. The timer measures the time elapsed from transmission start until 'echo' is obtained. 'Echo' is indicated by Collision Detect going active or Carrier Sense signal drop.

TDR command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• TDR = 5

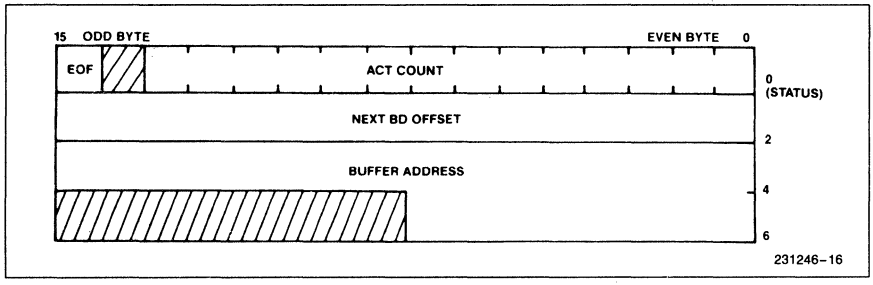


Figure 16. The Transmit Buffer Description

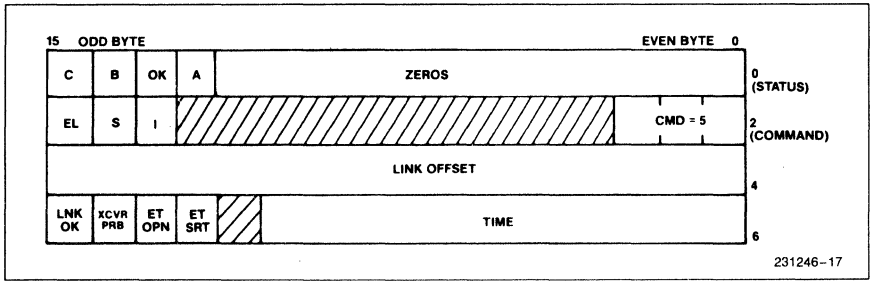


Figure 17. The TDR Command Block

LINK OFFSET: Address of next Command Block

RESULT word:

LNK-OK	(Bit 15)	• No Link Problem Identified
XCVR-PRB	(Bit 14)	• Transceiver Cable Problem identified (valid only in the case of a Transceiver that does not return Carrier Sense during transmission).
ET-OPN	(Bit 13)	• Open on the link identified (valid only in the case of a Transceiver that returns Carrier Sense during transmission).
ET-SRT	(Bit 12)	• Short on the link identified (valid only in the case of a Transceiver that returns Carrier Sense during transmission).
TIME	(Bits 0–10)	• Specifying the distance to a problem on the link (if one exists) in transmit clock cycles.

DUMP

This command causes the contents of over a hundred bytes of internal registers to be placed in memory. It is supplied as a self diagnostic tool, as well as to supply registers of interest to the user.

DUMP command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• DUMP = 6

LINK OFFSET: Address of next Command Block

BUFFER OFFSET: This word specifies the offset portion of the memory address which points to the top of the buffer allocated for the dumped registers contents. The length of the buffer is 170 bytes.

DUMP AREA FORMAT

Figure 18 shows the format of the DUMP area. The fields are as follows:

Bytes 00H to 0AH: These bytes correspond to the 82586 CONFIGURE command field.

Bytes 0CH to 11H: The Individual Address Register content. IARO is the Individual Address least significant byte.

Bytes 12H to 13H: Status word of last command block (only bits 0–13).

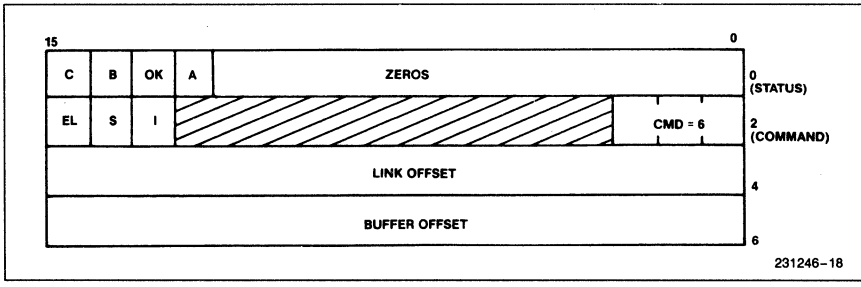


Figure 18. The DUMP Command Block

Bytes 14H to 17H: Content of the Transmit CRC generator. TXCRCRO is the least significant byte. The contents are dependent on the activity before the DUMP command:

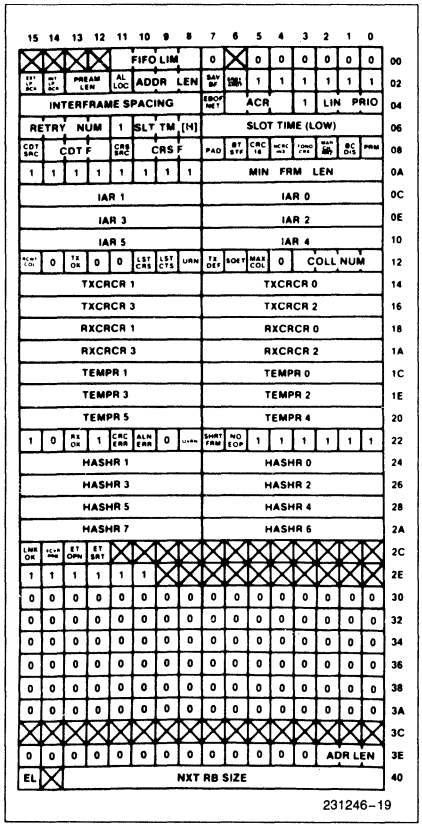
After RESET - 'All Ones.'

After successful transmission - 'All Zeros'.

After MC-SETUP command - Generated CRC value of the last MC address, on MC-LIST.

After unsuccessful transmission, depends on where it stopped.

NOTE:
For 16-bit CRC only TXCRCRO and TXCRCR1 are valid.



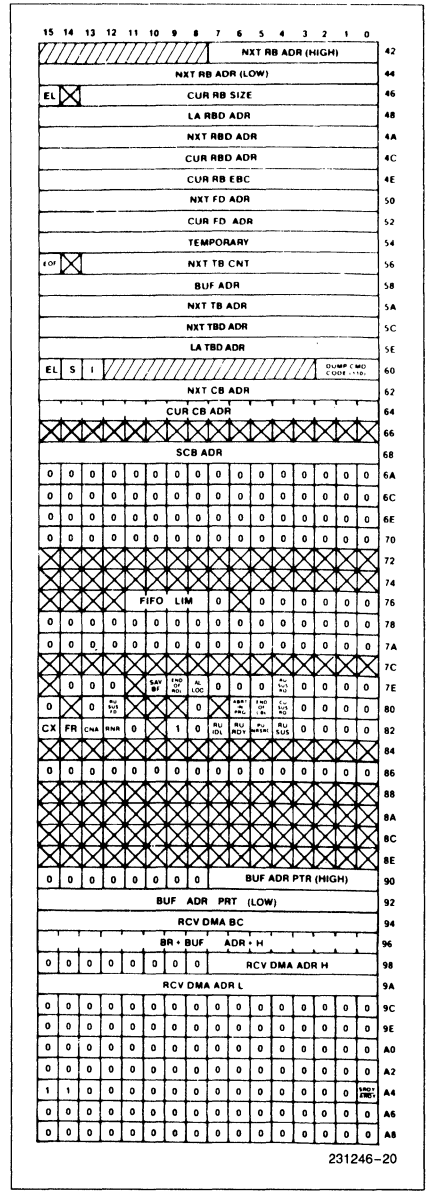


Figure 19. The DUMP Area

Bytes 18H to 1BH: Contents of Receive CRC Checker. RXCRCRO is the least significant byte. The contents are dependent on the activity performed before the DUMP command:

After RESET - 'All Ones.'

After good frame reception—

1. For CRC-CCITT - 0ID0FH
2. For CRC-Autodin-II - C704DD7BH

After Bad Frame reception - corresponds to the received information.

After reception attempt, i.e. unsuccessful check for address match, corresponds to the CRC performed on the frame address.

NOTE:
Any frame on the serial link modifies this register contents.

Bytes 1CH to 21H: Temporary Registers.

Bytes 22H to 23H: Receive Status Register. Bits 6, 7, 8, 10, 11 and 13 assume the same meaning as corresponding bits in the Receive Frame Descriptor Status field.

Bytes 24H to 2BH: HASH TABLE.

Bytes 2CH to 2DH: Status bits of the last time TDR command that was performed.

NXT-RB-SIZE: Let N be the last buffer of the last received frame, then NXT-RB-SIZE is the number of bytes of available in the N + 1 buffer. EL - The EL bit of the Receive Buffer Descriptor.

NXT-RB-ADR: Let N be the last Receive Buffer used, then NXT-RB-ADR is the BUFFER-ADDRESS field in the N + 1 Receive-Buffer Descriptor, i.e. the pointer to the N + 1 Receive Buffer.

CUR-RB-SIZE: The number of bytes in the last buffer of the last received frame. EL - The EL bit of the last buffer in the last received frame.

LA-RBD-ADR: Look Ahead Buffer Descriptor, i.e. the pointer to N + 2 Receiver Buffer Descriptor.

NXT-RBD-ADR: Next Receive Buffer Descriptor Address. Similar to LA-RBD-ADR but points to N + 1 Receive Buffer Descriptor.

CUR-RBD-ADR: Current Receive Buffer Descriptor Address. Similar to LA-RBD-ADR, but point to Nth Receive Buffer Descriptor.

CUR-RB-EBC: Current Receive Buffer Empty Byte Count Let N be the currently used Receive Buffer. Then CUR-RB-EBC indicates the Empty part of the buffer, i.e. the ACT-COUNT of buffer N is given by the difference between its SIZE and the CUR-RB-EBC.

NXT-FD-ADR: Next Frame Descriptor Address. Define N as the last Receive Frame Descriptor with bits C = 1 and B = 0, then NXT-FD-ADR is the address of N + 2 Receive Frame Descriptor (with B = C = 0) and is equal to the LINK-ADDRESS field in N + 1 Receive Frame Descriptor.

CUR-FD-ADR: Current Frame Descriptor Address. Similar to next NXT-FD-ADR but refers to N + 1 Receive Frame Descriptor (with B = 1, C = 0).

Bytes 54H to 55H: Temporary register.

NXT-TB-CNT: Next Transmit Buffer Count. Let N be the last transmitted buffer of the TRANSMIT command executed recently, the NXT-TB-CNT is the ACT-COUNT field in the Nth Transmit Buffer Descriptor. EOF - Corresponds to the EOF bit of the Nth Transmit Buffer Descriptor. EOF = 1 indicates that the last buffer accessed by the 82586 during Transmit was the last Transmit Buffer in the data buffer chain associated with the Transmit Command.

BUF-ADR: Buffer Address. The BUF-PTR field in the DUMP-STATUS Command Block.

NXT-TB-AD-L: Next Transmit Buffer Address Low. Let N be the last Transmit Buffer in the transmit buffer chain of the TRANSMIT Command performed recently, then NXT-TB-AD-L are the two least significant bytes of the Nth buffer address.

LA-TB-ADR: Look Ahead Transmit Buffer Descriptor Address. Let N be the last Transmit Buffer in the transmit buffer chain of the TRANSMIT Command performed recently, then LA-TBD-ADR is the NEXT-BD-ADDRESS field of the Nth Buffer Descriptor.

NXT-TBD-ADR: Next Transmit Buffer Descriptor Address. Similar in function to LA-TBD-ADR but related to Transmit Buffer Descriptor N-1. Actually, it is the address of Transmit Buffer Descriptor N.

Bytes 60H, 61H: This is a copy of the 2nd word in the DUMP-STATUS command presently executing.

NXT-CB-ADR: Next Command Block Address. The LINK-ADDRESS field in the DUMP Command Block presently executing. Points to the next command.

CUR-CB-ADR: Current Command Block Address. The address of the DUMP Command Block currently executing.

SCB-ADR: Offset of the System Control Block (SCB).

Bytes 7EH, 7FH:

RU-SUS-RQ (Bit 4) - Receive Unit Suspend Request.

Bytes 80H, 81H:

CU-SUS-RQ (Bit 4) - Command Unit Suspend Request.

END-OF-CBL (Bit 5) - End of Command Block List. If "1" indicates that DUMP-STATUS is the last command in the command chain.

ABRT-IN-PROG (Bit 6) - Command Unit Abort Request.

RU-SUS-FD (Bit 12) - Receive Unit Suspend Frame Descriptor Bit. Assume N is the Receive Frame Descriptor used recently, then RU-SUS-FD is equivalent to the S bit of N + 1 Receive Frame Descriptor.

Bytes 82H, 83H:

RU-SUS (Bit 4) - Receive Unit in SUSPENDED state.

RU-NRSRC (Bit 5) - Receive Unit in NO RESOURCES state.

RU-RDY (Bit 6) - Receive Unit in READY state.

RU-IDL (Bit 7) - Receive Unit in IDLE state.

RNR (Bit 12) - RNR Interrupt in Service bit.

CNA (Bit 13) - CNA Interrupt in Service bit.

FR (Bit 14) - FR Interrupt in Service bit.

CX (Bit 15) - CX Interrupt in Service bit.

Bytes 90H to 93H:

BUF-ADR-PTR - Buffer pointer is the absolute address of the bytes following the DUMP Command block.

Bytes 94H to 95H:

RCV-DMA-BC - Receive DMA Byte Count. This field contains number of bytes to be transferred during the next Receive DMA operation. The value depends on AL-LOCation configuration bit.

1. If AL-LOCation = 0 then RCV-DMA-BC = (2 times ADDR-LEN plus 2) if the next Receive Frame Descriptor has already been fetched.
2. If AL-LOCation = 1 then it contains the size of the next Receive Buffer.

BR + BUF − PTR + 96H - Sum of Base Address plus BUF − PTR field and 96H.

RCV-DMA-ADR - Receive DMA absolute Address. This is the next RCV-DMA start address. The value depends on AL-LOCation configuration bit.

1. If AL-LOCation = 0, then RCV-DMA-ADR is the Destination Address field located in the next Receive Frame Descriptor.
2. If AL-LOCation = 1, then RCV-DMA-ADR is the next Receive Data Buffer Address.

The following nomenclature has been used in the DUMP table:

0	• The 82586 writes zero in this location.
1	• The 82586 writes one in this location.
X	• The 82586 writes zero or one in this location.
///	• The 82586 copies this location from the corresponding position in the memory structure.

DIAGNOSE

The DIAGNOSE Command triggers an internal self test procedure of backoff related registers and counters.

The DIAGNOSE command includes the following:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
FAIL	(Bit 11)	• Indicates that the Self Test Procedured Failed

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• DIAGNOSE = 7

LINK OFFSET: Address of next Command Block.

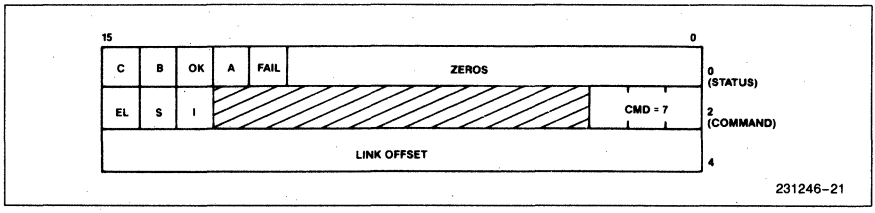


Figure 20. The DIAGNOSE Command Block

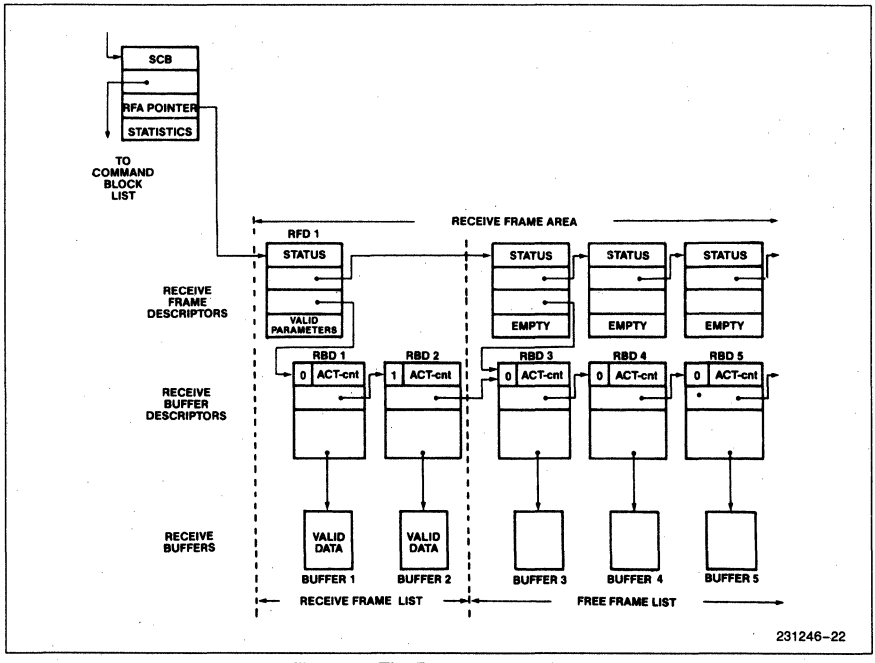


Figure 21. The Receive Frame Area

RECEIVE FRAME AREA (RFA)

The Receive Frame Area, RFA, is prepared by the host CPU, data is placed into the RFA by the 82586 as frames are received. RFA consists of a list of Receive Frame Descriptors (FD), each of which is associated with a frame. RFA-OFFSET field of SCB points to the first FD of the chain; the last FD is identified by the End-of-Listing flag (EL). See Figure 21.

FRAME DESCRIPTOR (FD) FORMAT

The FD includes the following fields:

STATUS word (set by the 82586):

C	(Bit 15)	• Completed Storing Frame.
B	(Bit 14)	• FD was Consumed by RU.

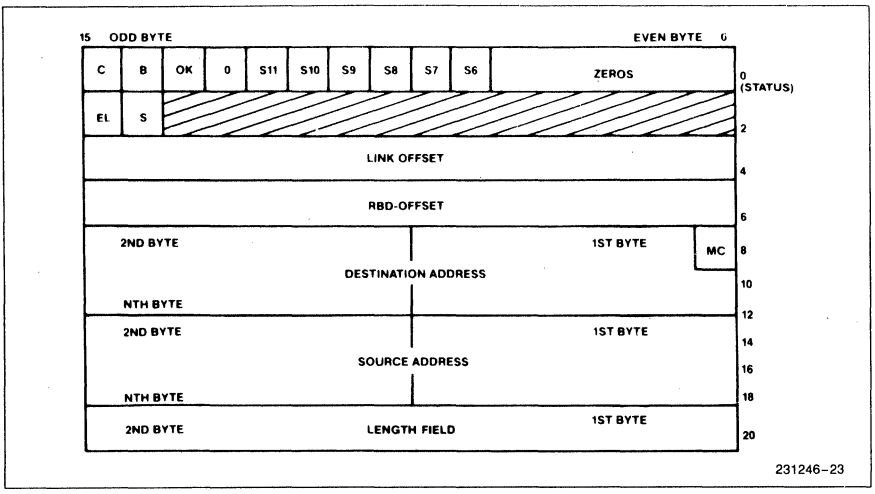


Figure 22. The Frame Descriptor (FD) Format

OK	(Bit 13)	• Frame received successfully. If this bit is set, then all others will be reset; if it is reset, then the other bits will indicate the nature of the error.
S11	(Bit 11)	• Received Frame Experienced CRC Error.
S10	(Bit 10)	• Received Frame Experienced an Alignment Error.
S9	(Bit 9)	• RU ran out of resources during reception of this frame.
S8	(Bit 8)	• RCV-DMA Overrun.
S7	(Bit 7)	• Received frame had fewer bits than configured Minimum Frame Length.
S6	(Bit 6)	• No EOF flag detected (only when configured to Bitstuffing).

COMMAND word:

EL	(Bit 15)	• Last FD in the List.
S	(Bit 14)	• RU should be suspended after receiving this frame.

LINK OFFSET: Address of next FD in list.

RBD-OFFSET: (initially prepared by the CPU and later may be updated by 82586): Address of the first RBD that represents the Information Field. RBD-OFFSET = 0FFFFH means there is no Information Field.

DESTINATION ADDRESS (written by 82586): Contains Destination Address of received frame. The length in bytes, it is determined by the Address Length configuration parameter.

SOURCE ADDRESS (written by 82586): Contains Source Address of received frame. Its length is the same as DESTINATION ADDRESS.

LENGTH FIELD (written by 82586): Contains the 2 byte Length or Type Field of received frame.

RECEIVE BUFFER DESCRIPTOR FORMAT

The Receive Buffer Descriptor (RBD) holds information about a buffer; size and location, and the means for forming a chain of RBDs, (forward pointer and end-of-frame indication).

The Buffer Descriptor contains the following fields.

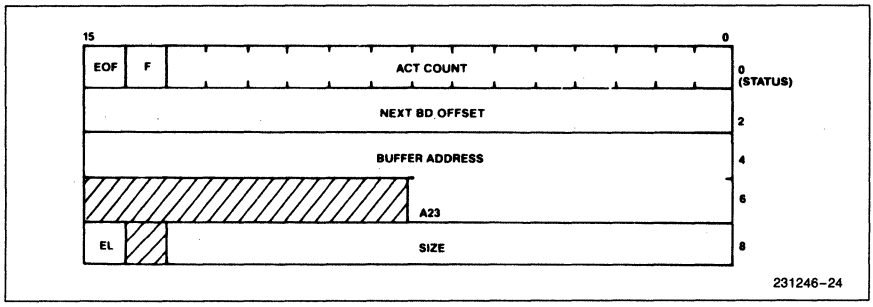


Figure 23. The Receive Buffer Descriptor (RBD) Format

STATUS word (written by the 82586).

EOF	(Bit 15)	• Last buffer in received frame.
F	(Bit 14)	• ACT COUNT field is valid.
ACT COUNT	(Bits 0–13)	• Number of bytes in the buffer that are actually occupied.

NEXT RBD OFFSET: Address of next BD in list of BD's.

BUFFER ADDRESS: 24-bit absolute address of buffer.

EL/SIZE:

EL	(BIT 15)	• Last BD in list.
SIZE	(Bits 0–13)	• Number of bytes the buffer is capable of holding.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature.....−65°C to 150°C
Voltage on Any Pin with
Respect to Ground.....−1.0V to +7V
Power Dissipation.....3.0 Watts

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, T_C = 0°C to 105°C, V_{CC} = 5V ±10%, CLK has MOS levels (See V_{MIL}, V_{MIH}, V_{MOL}, V_{MOH}). $\overline{TxC}$ and $\overline{RxC}$ have 82C501 compatible levels (V_{MIL}, V_{TIH}, V_{RIH}). All other signals have TTL levels (see V_{IL}, V_{IH}, V_{OL}, $_{OH}$).

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (TTL)	−0.5	+0.8	V	
V_{IH}	Input High Voltage (TTL)	2.0	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage (TTL)		0.45	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage (TTL)	2.4		V	I_{OH} − 400 μA
V_{MIL}	Input Low Voltage (MOS)	−0.5	0.6	V	
V_{MIH}	Input High Voltage (MOS)	3.9	V_{CC} + 0.5	V	
V_{TIH}	Input High Voltage ($\overline{TxC}$)	3.3	V_{CC} + 0.5	V	
V_{RIH}	Input High Voltage ($\overline{RxC}$)	3.0	V_{CC} + 0.5	V	
V_{MOL}	Output Low Voltage (MOS)		0.45	V	I_{OL} 2.5 mA
V_{MOH}	Output High Voltage (MOS)	V_{CC} − 0.5		V	I_{OH} − 400 μA
I_{LI}	Input Leakage Current		±10	μA	$0 \leq V_{IN} \leq V_{CC}$
I_{LO}	Output Leakage Current		±10	μA	$0.45 \leq V_{OUT} \leq V_{CC}$
C_{IN}	Capacitance of Input Buffer		10	pF	FC = 1 MHz
C_{OUT}	Capacitance of Output Buffer		20	pF	FC = 1 MHz
I_{CC}	Power Supply Current		550 450	mA	T_A = 0°C T_A = 70°C

SYSTEM INTERFACE A.C. TIMING CHARACTERISTICS

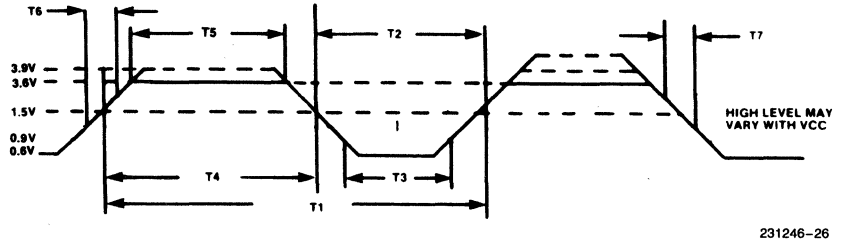
T_A = 0°C to 70°C, T_C = 0°C to 105°C, V_{CC} = 5V ±10%. Figures 24 and 25 define how the measurements should be done.

INPUT AND OUTPUT WAVEFORMS FOR A.C. TESTS

231246-25

AC Testing Inputs are Driven at 2.4V for a Logic 1 and 0.45 for a Logic 0. Timing measurements are made at 1.5V for both a Logic 1 and 0

Figure 24. TTL Input/Output Voltage Levels for Timing Measurements



231246-26

MOS I/O measurements are taken at 0.1 and 0.9 of the voltage swing

Figure 25. System Clock CMOS Input Voltage Levels for Timing Measurements

INPUT TIMING REQUIREMENTS*

Symbol	Parameter	82586-6 (6 MHz)		82586 (8 MHz)		82586-10 (10 MHz)		Comments
		Min	Max	Min	Max	Min	Max	
T1	CLK Cycle Period	166	2000	125	2000	100	200	
T2	CLK Low Time at 1.5V	73	1000	55	1000	44	1000	
T3	CLK Low Time at 0.9V			42.5	1000	42.5	1000	
T4	CLK High Time at 1.5V	73		55		44		
T5	CLK High Time at 3.6V			42.5		42.5		
T6	CLK Rise Time		15		15		12	Note 1
T7	CLK Fall Time		15		15		12	Note 2
T8	Data in Setup Time	20		20		15		
T9	Data in Hold Time	10		10		10		
T10	Async RDY Active Setup Time	20		20		15		Note 3
T11	Async RDY Inactive Setup Time	35		35		25		Note 3
T12	Async RDY Hold Time	15		15		15		Note 3
T13	Synchronous Ready/Active Setup	35		35		20		
T14	Synchronous Ready Hold Time	0		0		0		
T15	HLDA Setup Time	20		20		20		Note 3
T16	HLDA Hold Time	10		10		5		Note 3
T17	Reset Setup Time	20		20		20		Note 3
T18	Reset Hold Time	10		10		10		Note 3
T19	CA Pulse Width	1 T1		1 T1		1 T1		
T20	CA Setup Time	20		20		20		Note 3
T21	CA Hold Time	10		10		10		Note 3

OUTPUT TIMINGS**

Symbol	Parameter	Min	Max	Min	Max	Min	Max	Comments
T22	DT/R Valid Delay	0	60	0	60	0	44	
T23	$\overline{WR}$, $\overline{DEN}$ Active Delay	0	70	0	70	0	56	
T24	$\overline{WR}$, $\overline{DEN}$ Inactive Delay	10	65	10	65	10	45	
T25	Int. Active Delay	0	85	0	85	0	70	Note 4
T26	Int. Inactive Delay	0	85	0	85	0	70	Note 4
T27	Hold Active Delay	0	85	0	85	0	70	Note 4
T28	Hold Inactive Delay	0	85	0	85	0	70	Note 4
T29	Address Valid Delay	0	55	0	55	0	50	
T30	Address Float Delay	0	50	0	50	12	50	
T31	Data Valid Delay	0	55	0	55	0	50	Note 7
T32	Data Hold Time	0		0		0		
T33	Status Active Delay	10	60	10	60	10	45	

OUTPUT TIMINGS** (Continued)

Symbol	Parameter	82582-6 (6 MHz)		82586 (8 MHz)		82586-10 (10 MHz)		Comments
		Min	Max	Min	Max	Min	Max	
T34	Status Inactive Delay	10	70	10	70	10	50	Note 8
T35	ALE Active Delay	0	45	0	45	0	35	Note 5
T36	ALE Inactive Delay	0	45	0	45	0	37	Note 5
T37	ALE Width	T2-10		T2-10		T2-10		Note 5
T38	Address Valid to ALE Low	T2-40		T2-30		T2-25		
T39	Address Hold to ALE Inactive	T4-10		T4-10		T4-10		
T40	$\overline{RD}$ Active Delay	10	95	10	95	10	95	
T41	$\overline{RD}$ Inactive Delay	10	70	10	70	10	70	
T42	$\overline{RD}$ Width	2T1-50		2T1-50		2T1-46		
T43	Address Float to $\overline{RD}$ Active	10		10		0		
T44	$\overline{RD}$ Inactive to Address Active	T1-40		T1-40		T1-34		
T45	$\overline{WR}$ Width	2T1-40		2T1-40		2T1-34		
T46	Data Hold After $\overline{WR}$	T2-25		T2-25		T2-25		
T47	Control Inactive After Reset	0	60	0	60	0	60	Note 6

*All units are in ns.
**CL on all outputs is 20-200 pF unless otherwise specified.

NOTES:
1. 1.0V to 3.5V
2. 3.5V to 1.0V
3. To guarantee recognition at next clock
4. CL = 50 pF
5. CL = 100 pF
6. Affects:
   MIN MODE: $\overline{RD}$, $\overline{WR}$, DT/$\overline{R}$, $\overline{DEN}$
   MAX MODE: $\overline{S0}$, $\overline{S1}$
7. High address lines (A16-A24, BHE) become valid one clock before T1 only on first memory cycle after the 82586 acquired the bus.
8. $\overline{ST}$, S0 go inactive just prior to T4.

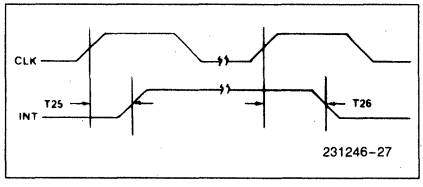


Figure 26. INT Output Timing

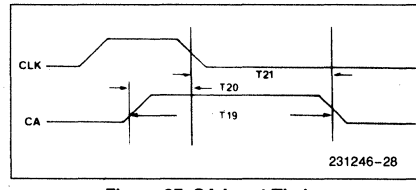


Figure 27. CA Input Timing

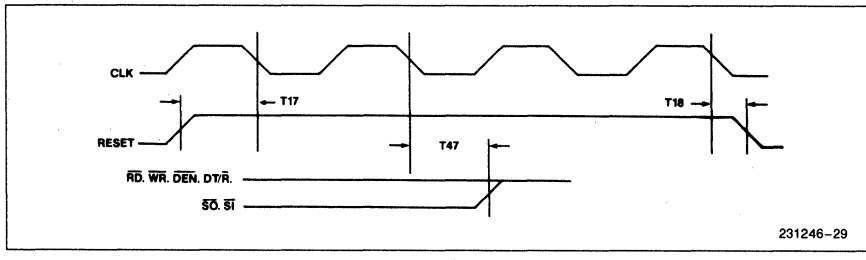


Figure 28. RESET Timing

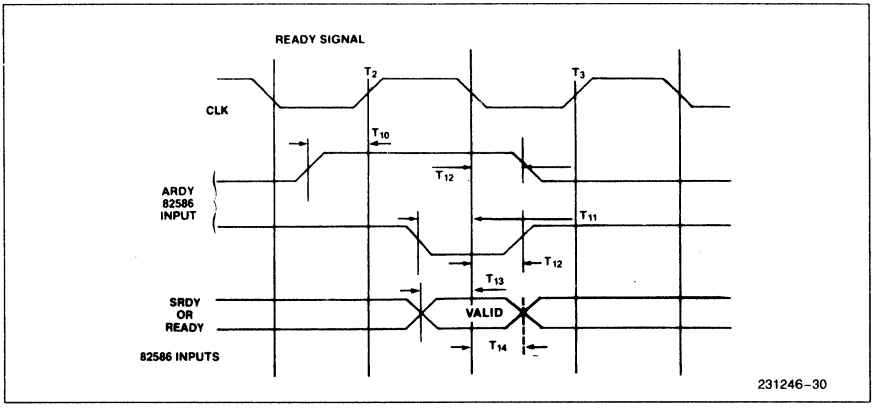


Figure 29. ARDY and SRDY Timings Relative to CLK

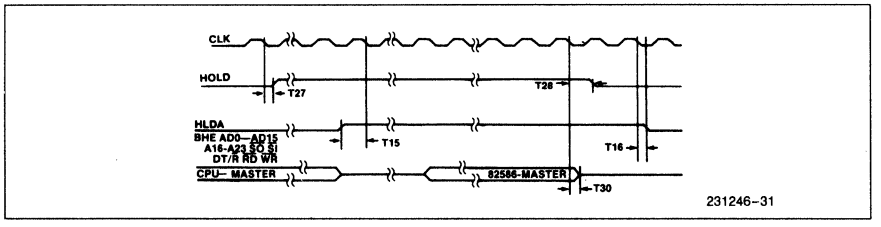


Figure 30. HOLD/HLDA Timing Relative to CLK

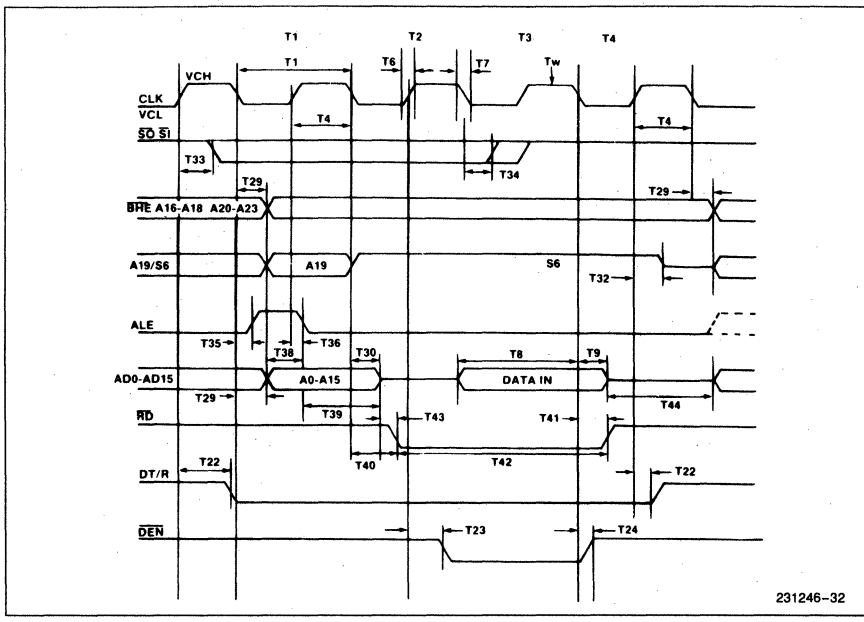


Figure 31. Read Cycle Timing

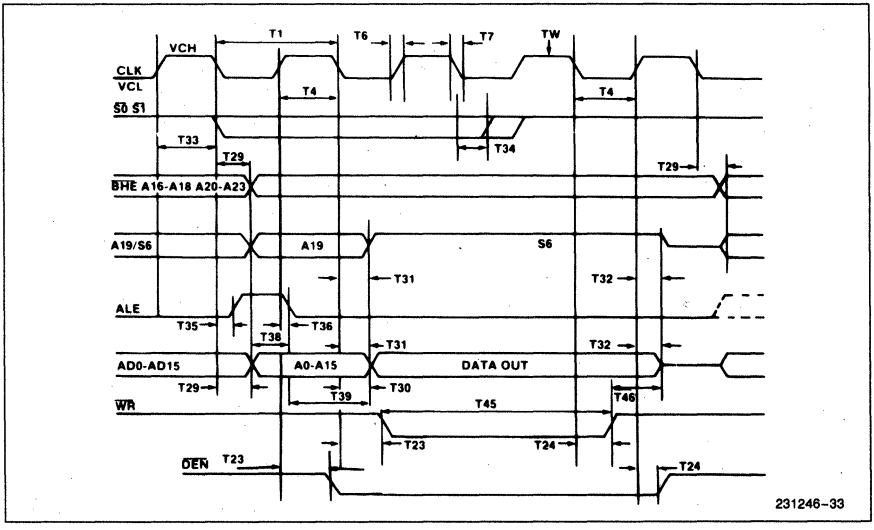


Figure 32. Write Cycle Timing

SERIAL INTERFACE A.C. TIMING CHARACTERISTICS

CLOCK SPECIFICATION

Applies for $\overline{TxC}$, $\overline{RxC}$ for NRZ:

f min = 100 kHz ± 100 ppm

f max = 10 MHz ± 100 ppm

for Manchester:

f min = 500 kHz ± 100 ppm

f max = 10 MHz ± 100 ppm

for Manchester, symmetry is needed:

$$T51, T52 = \frac{1}{2f} \pm 5\%$$

A.C. CHARACTERISTICS

TRANSMIT AND RECEIVE TIMING PARAMETER SPECIFICATION*

Symbol	Parameter	Min	Max	Comments
TRANSMIT CLOCK PARAMETERS				
T48	$\overline{TxC}$ Cycle	100	1000	Notes 14, 2
T48	$\overline{TxC}$ Cycle	100		Notes 14, 3
T49	$\overline{TxC}$ Rise Time		5	Note 14
T50	$\overline{TxC}$ Fall Time		5	Note 14
T51	$\overline{TxC}$ High Time @ 3.0V	40	1000	Note 14
T52	$\overline{TxC}$ Low Time @0.9V	40		Notes 14, 4
TRANSMIT DATA PARAMETERS				
T53	TxD Rise Time		10	Notes 5, 13
T54	TxD Fall Time		10	Notes 5, 13
T55	TxD Transition-Transition	Min (T51, T52) − 7		Notes 2, 5
T56	$\overline{TxC}$ Low to TxD Valid		40	Notes 3, 5
T57	$\overline{TxC}$ Low to TxD Transition		30	Notes 2, 5
T58	$\overline{TxC}$ High to TxD Transition		30	Notes 2, 5
T59	$\overline{TxC}$ Low to TxD High at the Transmission End		40	Note 5
REQUEST TO SEND/CLEAR TO SEND PARAMETERS				
T60	$\overline{TxC}$ Low to $\overline{RTS}$ Low. Time to Activate $\overline{RTS}$		40	Note 6
T61	$\overline{CTS}$ Valid to $\overline{TxC}$ Low. $\overline{CTS}$ Setup Time	45		
T62	$\overline{TxC}$ Low to $\overline{CTS}$ Invalid. $\overline{CTS}$ Hold Time	20		Note 7
T63	$\overline{TxC}$ Low to $\overline{RTS}$ High, time to Deactivate $\overline{RTS}$		40	Note 6
RECEIVE CLOCK PARAMETERS				
T64	$\overline{RxC}$ Clock Cycle	100		Notes 15, 3
T65	$\overline{RxC}$ Rise Time		5	Note 15
T66	$\overline{RxC}$ Fall Time		5	Note 15
T67	$\overline{RxC}$ High Time @ 2.7V	36	1000	Note 15
T68	$\overline{RxC}$ Low Time @0.9V	40		Note 15

*All units are in ns.

A.C. CHARACTERISTICS (Continued)

TRANSMIT AND RECEIVE TIMING PARAMETER SPECIFICATION* (Continued)

Symbol	Parameter	Min	Max	Comments
RECEIVE DATA PARAMETERS				
T69	RxD Setup Time	30		Note 1
T70	RxD Hold Time	30		Note 1
T71	RxD Rise Time		10	Note 1
T72	RxD Fall Time		10	Note 1
CARRIER SENSE/COLLISION DETECT PARAMETERS				
T73	$\overline{CDT}$ Valid to $\overline{TxC}$ High Ext. Collision Detect Setup Time	30		Note 12
T74	$\overline{TxC}$ High to $\overline{CDT}$ Inactive. $\overline{CDT}$ Hold Time	20		Note 12
T75	$\overline{CDT}$ Low to Jamming Start			Note 8
T76	$\overline{CRS}$ Valid to $\overline{TxC}$ High Ext. Carrier Sense Setup Time	30		Note 12
T77	$\overline{TxC}$ High to $\overline{CRS}$ Inactive. $\overline{CRS}$ Hold Time	20		Note 12
T78	$\overline{CRS}$ Low to Jamming Start			Note 9
T79	Jamming Period			Note 10
T80	$\overline{CRS}$ Inactive Setup Time to $\overline{RxC}$ High End of Receive Frame	60		
T81	$\overline{CRS}$ Active Hold Time from $\overline{RxC}$ High	3		
INTERFRAME SPACING PARAMETER				
T82	Inter Frame Delay			Note 11

*All units are in ns.

NOTES:
1. TTL levels
2. Manchester only
3. NRZ only
4. Manchester requires 50% duty cycle
5. 1 TTL load + 50 pF
6. 1 TTL load + 100 pF
7. Abnormal end of transmission. $\overline{CTS}$ expires before $\overline{RTS}$
8. Programmable value:
   T75 = NCDF × T48 + (12.5 to 23.5) × T48 if collision occurs after preamble
   NCDF—The collision detection filter configuration value
9. Programmable value:
   T78 = NCSF × T48 + (12.5 to 23.5) × T48
   NCSF—The carrier sense filter configuration value
   TBD is a function of internal/external carrier sense bit
10. T79 = 32 ⁄ T48
11. Programmable value:
    T82 = NIFS ⁄ T48
    NIFS—the IFS configuration value
*12. To guarantee recognition on the next clock
13. Applies to TTL levels
14. 82C501 compatible levels, see Figure 34
15. 82C501 compatible levels, see Figure 35

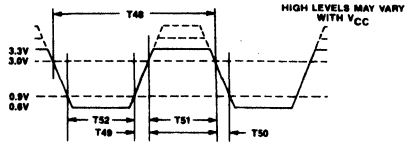
A.C. TIMING CHARACTERISTICS

Input and Output Waveforms for AC Tests

231246-34

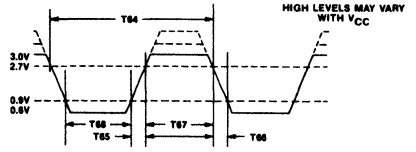
AC testing inputs are driven at 2.4V for a Logic 1 and 0.45 for a Logic 0. Timing measurements are made at 1.5V for both a Logic 1 and 0.

Figure 33. TTL Input/Output Voltage Levels for Timing Measurements



231246-35

Figure 34. $\overline{TxC}$ Input Voltage Levels for Timing Measurements



231246-36

Figure 35. $\overline{RxC}$ Input Voltage Levels for Timing Measurements

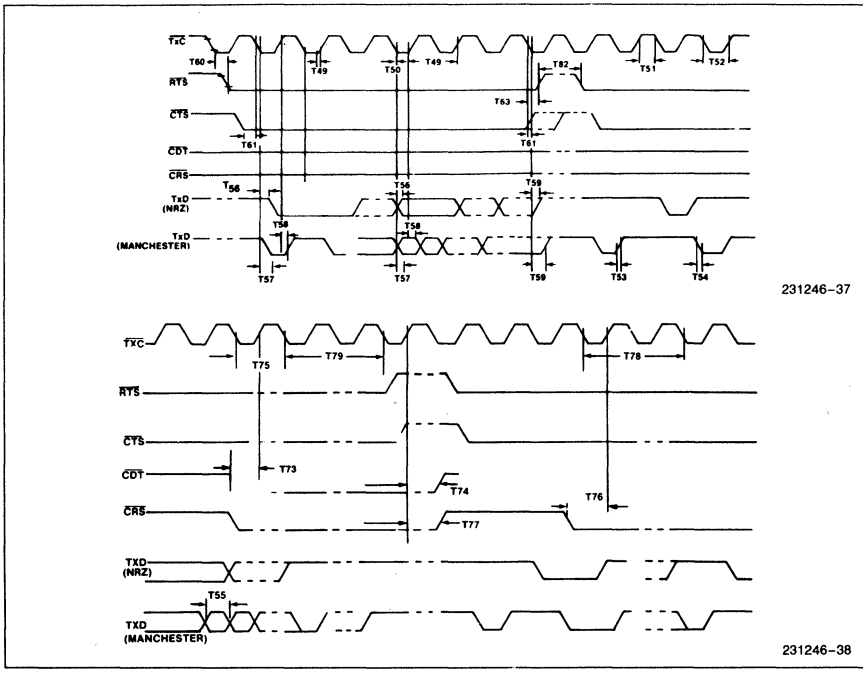


Figure 36. Transmit and Control and Data Timing

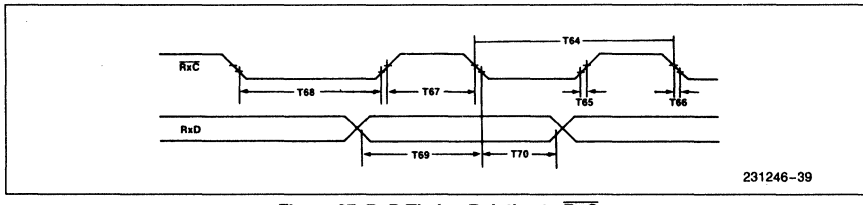


Figure 37. RxD Timing Relative to $\overline{RxC}$

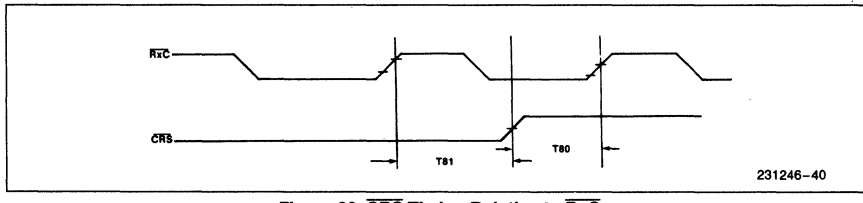


Figure 38. $\overline{CRS}$ Timing Relative to $\overline{RxC}$

Appendix

C

82596CA High-Performance 32-Bit LAN Coprocessor*

*Reprinted by permission of Intel Corporation. Copyright/Intel Corporation 1990.

- **Performs Complete CSMA/CD Medium Access Control (MAC) Functions—Independently of CPU**
  - **— IEEE 802.3 (EOC) Frame Delimiting**
  - **— HDLC Frame Delimiting**
- **Supports Industry Standard LANs**
  - **— IEEE TYPE 10BASE5 (Ethernet*), IEEE TYPE 10BASE2 (Cheapernet), IEEE TYPE 1BASE5 (StarLAN), and the Proposed Standards TYPE 10BASE-T and 10BASE-F**
  - **— Proprietary CSMA/CD Networks Up to 20 Mb/s**
- **On-Chip Memory Management**
  - **— Automatic Buffer Chaining**
  - **— Buffer Reclamation after Receipt of Bad Frames; Optional Save Bad Frames**
  - **— 32-Bit Segmented or Linear (Flat) Memory Addressing Formats**
- **Network Management and Diagnostics**
  - **— Monitor Mode**
  - **— 32-Bit Statistical Counters**
- **82586 Software Compatible**
- **Optimized CPU Interface**
  - **— Optimized Bus Interface to Intel's i486™ and 80960CA Processors**
  - **— Supports Big Endian and Little Endian Byte Ordering**
- **32-Bit Bus Master Interface**
  - **— 106 MB/s Bus Bandwidth**
  - **— Burst Bus Transfers**
  - **— Bus Throttle Timers**
  - **— Transfers Data at 100% of Serial Bandwidth**
  - **— 128-Byte Receive FIFO, 64-Byte Transmit FIFO**
- **Self-Test Diagnostics**
- **Configurable Initialization Root for Data Structures**
- **High-Speed, 5V, CHMOS** IV Technology**
- **132-Pin Plastic Quad Flat Pack (PQFP) and PGA Package**

(See Packaging Spec Order No. 231369)

i486 is a trademark of Intel Corporation.
*Ethernet is a registered trademark of Xerox Corporation.
**CHMOS is a patented process of Intel Corporation.

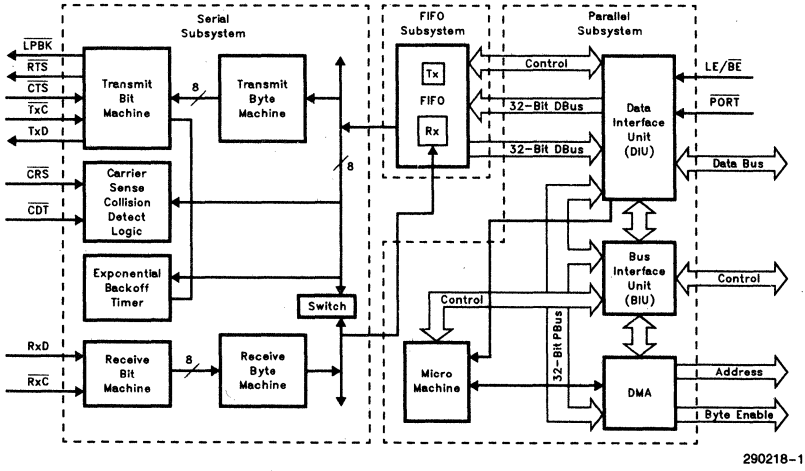


Figure 1. 82596CA Block Diagram

82596CA High-Performance 32-Bit Local Area Network Coprocessor

CONTENTS

PAGE

INTRODUCTION 1-40

PIN DESCRIPTIONS 1-44

82596 AND HOST CPU INTERACTION 1-48

82596 BUS INTERFACE 1-48

82596 MEMORY ADDRESSING 1-48

82596 SYSTEM MEMORY STRUCTURE 1-50

TRANSMIT AND RECEIVE MEMORY STRUCTURES 1-51

TRANSMITTING FRAMES 1-54

RECEIVING FRAMES 1-55

82596 NETWORK MANAGEMENT AND DIAGNOSTICS 1-55

NETWORK PLANNING AND MAINTENANCE 1-57

STATION DIAGNOSTICS AND SELF-TEST 1-58

82586 SOFTWARE COMPATIBILITY ... 1-58

INITIALIZING THE 82596 1-58

SYSTEM CONFIGURATION POINTER (SCP) 1-58

Writing the Sysbus 1-59

INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP) 1-60

INITIALIZATION PROCESS 1-60

CONTROLLING THE 82596CA 1-61

82596 CPU ACCESS INTERFACE (PORT) 1-61

MEMORY ADDRESSING FORMATS 1-62

LITTLE ENDIAN AND BIG ENDIAN BYTE ORDERING 1-62

COMMAND UNIT (CU) 1-62

CONTENTS

PAGE

RECEIVE UNIT (RU) 1-63

SYSTEM CONTROL BLOCK (SCB) 1-63

SCB OFFSET ADDRESSES 1-66

CBL Offset (Address) 1-66

RFA Offset (Address) 1-66

SCB STATISTICAL COUNTERS 1-67

Statistical Counter Operation 1-67

ACTION COMMANDS AND OPERATING MODES 1-68

NOP 1-69

Individual Address Setup 1-69

Configure 1-70

Multicast-Setup 1-76

Transmit 1-77

Jamming Rules 1-79

TDR 1-80

Dump 1-82

Diagnose 1-85

RECEIVE FRAME DESCRIPTOR 1-86

Simplified Memory Structure 1-86

Flexible Memory Structure 1-87

Receive Buffer Descriptor (RBD) 1-88

ELECTRICAL AND TIMING CHARACTERISTICS 1-93

DC Characteristics 1-93

AC Characteristics 1-94

82596CA Input/Output System Timings 1-94

Transmit/Receive Clock Parameters 1-96

82596CA BUS Operation 1-99

System Interface AC Timing Characteristics 1-100

Input Waveforms 1-101

Serial AC Timing Characteristics 1-103

OUTLINE DIAGRAM 1-105

REVISION HISTORY 1-109

INTRODUCTION

The 82596CA is an intelligent, high-performance 32-bit Local Area Network coprocessor. The 82596CA implements the CSMA/CD access method and can be configured to support all existing IEEE 802.3 standards—TYPEs 10BASE5, 10BASE2, 1BASE5, and 10BROAD36. It can also be used to implement the proposed standards TYPE 10BASE-T and 10BASE-F. The 82596CA performs high-level commands, command chaining, and interprocessor communications via shared memory, thus relieving the host CPU of many tasks associated with network control. All time-critical functions are performed independently of the CPU, this increases network performance and efficiency. The 82596CA bus interfaces is optimized for Intel's i486™, 80960CA, and 80960KB processors.

The 82596CA implements all IEEE 802.3 Medium Access Control and channel interface functions, these include framing, preamble generation and stripping, source address generation, destination address checking, short-frame detection, and automatic length-field handling. Data rates up to 20 Mb/s are supported.

The 82596CA provides a powerful host system interface. It manages memory structures automatically, with command chaining and bidirectional data chaining. An on-chip DMA controller manages four channels, this allows autonomous transfer of data blocks (buffers and frames) and relieves the CPU of byte transfer overhead. Buffers containing errored or collided frames can be automatically recovered without CPU intervention. The 82596CA provides an upgrade path for existing 82586 software drivers by providing an 82586-software-compatible mode that supports the current 82586 memory structure. The 82586CA also has a Flexible memory structure and a Simplified memory structure. The 82596CA can address up to 4 gigabytes of memory. The 82596CA supports Little Endian and Big Endian byte ordering.

The 82596CA bus interface can achieve a burst transfer rate of 106 MB/s at 33 MHz. The bus interface employs bus throttle timers to regulate 82596CA bus use. Two large, independent FIFOs—128 bytes for Receive and 64 bytes for Transmit—tolerate long bus latencies and provide programmable thresholds that allow the user to optimize bus overhead for any worst-case bus latency. The high-performance bus is capable of back-to-back transmission and reception during the IEEE 802.3 9.6-μs Interframe Spacing (IFS) period.

The 82596CA provides a wide range of diagnostics and network management functions, these include internal and external loopback, exception condition tallies, channel activity indicators, optional capture of all frames regardless of destination address (promiscuous mode), optional capture of errored or collided frames, and time domain reflectometry for locating fault points on the network cable. The statistical counters, in 32-bit segmented and linear modes, are 32-bits each and include CRC errors, alignment errors, overrun errors, resource errors, short frames, and received collisions. The 82596CA also features a monitor mode for network analysis. In this mode the 82596CA can capture status bytes, and update statistical counters, of frames monitored on the link without transferring the contents of the frames to memory. This can be done concurrently while transmitting and receiving frames destined for that station.

The 82596CA can be used in both baseband and broadband networks. It can be configured for maximum network efficiency (minimum contention overhead) with networks of any length. Its highly flexible CSMA/CD unit supports address field lengths of zero through six bytes—configurable to either IEEE 802.3/Ethernet or HDLC frame delimitation. It also supports 16- or 32-bit cyclic redundancy checks. The CRC can be transferred directly to memory for receive operations, or dynamically inserted for transmit operations. The CSMA/CD unit can also be configured for full duplex operation for high throughput in point-to-point connections.

The 82596CA is fabricated with Intel's reliable, 5-V, CHMOS IV technology. It is available in a 132-pin PQFP or PGA package.

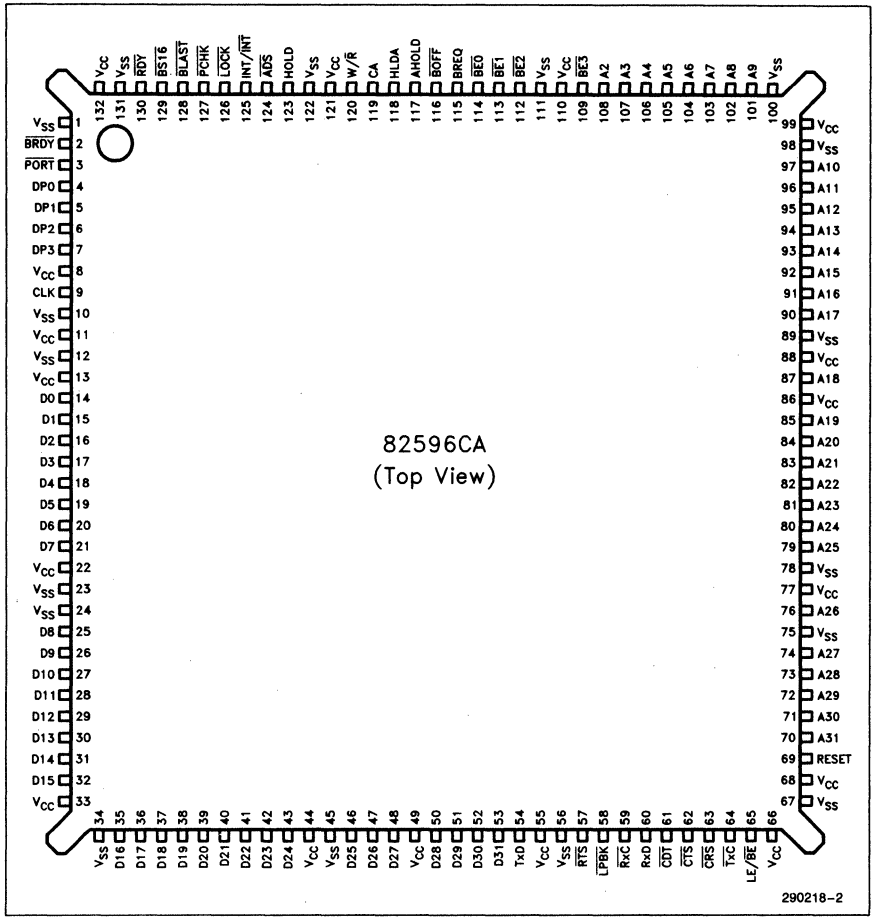


Figure 2. 82596CA PQFP Pin Configuration

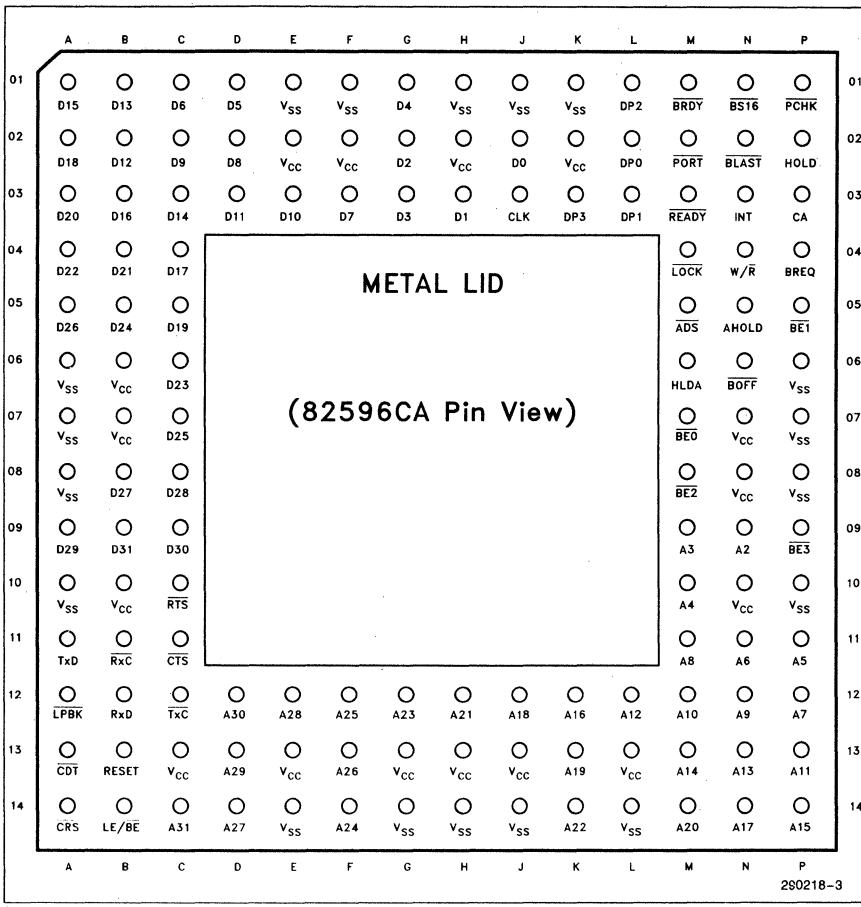


Figure 3. 82596CA PGA Pinout

82596CA PGA Cross Reference by Pin Name

Address		Data		Control		Serial Interface		V_{CC}	V_{SS}
Signal	**Pin No.**	**Signal**	**Pin No.**	**Signal**	**Pin No.**	**Signal**	**Pin No.**	**Pin No.**	**Pin No.**
A2	N9	D0	J2	$\overline{ADS}$	M5	$\overline{CDT}$	A13	B6	A6
A3	M9	D1	H3	AHOLD	N5	$\overline{CRS}$	A14	B7	A7
A4	M10	D2	G2	$\overline{BE0}$	M7	$\overline{CTS}$	C11	B10	A8
A5	P11	D3	G3	$\overline{BE1}$	P5	$\overline{LPBK}$	A12	C13	A10
A6	N11	D4	G1	$\overline{BE2}$	M8	$\overline{RTS}$	C10	E2	E1
A7	P12	D5	D1	$\overline{BE3}$	P9	$\overline{RxC}$	B11	E13	E14
A8	M11	D6	C1	$\overline{BLAST}$	N2	$\overline{RxD}$	B12	F2	F1
A9	N12	D7	F3	$\overline{BOFF}$	N6	$\overline{TxC}$	C12	G13	G14
A10	M12	D8	D2	$\overline{BRDY}$	M1	TxD	A11	H2	H1
A11	P13	D9	C2	BREQ	P4			H13	H14
A12	L12	D10	E3	$\overline{BS16}$	N1			J13	J1
A13	N13	D11	D3	CA	P3			K2	J14
A14	M13	D12	B2	CLK	J3			L13	K1
A15	P14	D13	B1	DP0	L2			N7	L14
A16	K12	D14	C3	DP1	L3			N8	P6
A17	N14	D15	A1	DP2	L1			N10	P7
A18	J12	D16	B3	DP3	K3				P8
A19	K13	D17	C4	HLDA	M6				P10
A20	M14	D18	A2	HOLD	P2				
A21	H12	D19	C5	INT/$\overline{INT}$	N3				
A22	K14	D20	A3	LE/$\overline{BE}$	B14				
A23	G12	D21	B4	$\overline{LOCK}$	M4				
A24	F14	D22	A4	$\overline{PCHK}$	P1				
A25	F12	D23	C6	$\overline{PORT}$	M2				
A26	F13	D24	B5	$\overline{READY}$	M3				
A27	D14	D25	C7	RESET	B13				
A28	E12	D26	A5	W/$\overline{R}$	N4				
A29	D13	D27	B8						
A30	D12	D28	C8						
A31	C14	D29	A9						
		D30	C9						
		D31	B9						

PIN DESCRIPTIONS

<table>
<tr><th>Symbol</th><th>PQFP Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>CLK</td><td>9</td><td>I</td><td>CLOCK. The system clock input provides the fundamental timing for the 82596. It is a 1X CLK input used to generate the 82596 clock and requires TTL levels. All external timing parameters are specified in reference to the rising edge of CLK.</td></tr>
<tr><td>D0–D31</td><td>14–53</td><td>I/O</td><td>DATA BUS. The 32 Data Bus lines are bidirectional, tri-state lines that provide the general purpose data path between the 82596 and memory. With the 82596 the bus can be either 16 or 32 bits wide; this is determined by the $\overline{BS16}$ signal. The 82596 always drives all 32 data lines during Write operations, even with a 16-bit bus. D31– D0 are floated after a Reset or when the bus is not acquired.
These lines are inputs during a CPU Port access; in this mode the CPU writes the next address to the 82596 through the data lines. During $\overline{PORT}$ commands (Relocatable SCP, Self-Test, Reset and Dump) the address must be aligned to a 16-byte boundary. This frees the D_3–D_0 lines so they can be used to distinguish the commands. The following is a summary of the decoding data.
<table>
<tr><th>D0</th><th>D1</th><th>D2</th><th>D3</th><th>D31–D4</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0000</td><td>Reset</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>ADDR</td><td>Relocatable SCP</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>ADDR</td><td>Self-Test</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>ADDR</td><td>Dump Command</td></tr>
</table></td></tr>
<tr><td>DP0–DP3</td><td>4–7</td><td>I/O</td><td>DATA PARITY. These are tri-stated data parity pins. There is one parity line for each byte of the data bus. The 82596 drives them with even-parity information during write operations having the same timing as data writes. Likewise, even-parity information, with the same timing as read information, must be driven back to the 82596 over these pins to ensure that the correct parity check status is indicated by the 82596.</td></tr>
<tr><td>$\overline{PCHK}$</td><td>127</td><td>O</td><td>PARITY CHECK. This pin is driven high one clock after $\overline{RDY}$ to inform Read operations of the parity status of data sampled at the end of the previous clock cycle. When driven low it indicates that incorrect parity data has been sampled. It only checks the parity status of enabled bytes, which are indicated by the Byte Enable and Bus Size signals. $\overline{PCHK}$ is only valid for one clock time after data read is returned to the 82596; i.e., it is inactive (high) at all other times.</td></tr>
<tr><td>A31-A2</td><td>7008</td><td>O</td><td>ADDRESS LINES. These 30 tri-stated Address lines output the address bits required for memory operation. These lines are floated after a Reset or when the bus is not acquired.</td></tr>
<tr><td>$\overline{BE3}$–$\overline{BE0}$</td><td>109–114</td><td>O</td><td>BYTE ENABLE. These tri-stated signals are used to indicate which bytes are involved with the current memory access. The number of Byte Enable signals asserted indicates the physical size of the data being transferred (1, 2, 3, or 4 bytes).
• $\overline{BE0}$ indicates D7–D0
• $\overline{BE1}$ indicates D15–D8
• $\overline{BE2}$ indicates D23–D16
• $\overline{BE3}$ indicates D31–D24
These lines are floated after a Reset or when the bus is not acquired.</td></tr>
<tr><td>W/$\overline{R}$</td><td>120</td><td>O</td><td>WRITE/READ. This dual function pin is used to distinguish Write and Read cycles. This line is floated after a Reset or when the bus is not acquired.</td></tr>
</table>

PIN DESCRIPTIONS (Continued)

Symbol	PQFP Pin No.	Type	Name and Function
$\overline{ADS}$	124	O	**ADDRESS STATUS.** The 82596 uses this tri-state pin to indicate to indicate that a valid bus cycle has begun and that A31–A2, $\overline{BE3}$–$\overline{BE0}$, and W/$\overline{R}$ are being driven. It is asserted during t1 bus states. This line is floated after a Reset or when the bus is not acquired.
$\overline{RDY}$	130	I	**READY.** Active low. This signal is the acknowledgment from addressed memory that the transfer cycle can be completed. When high, it causes wait states to be inserted. It is ignored at the end of the first clock of the bus cycle's data cycle. This active-low signal does not have an internal pull-up resistor. This signal must meet the setup and hold times to operate correctly.
$\overline{BRDY}$	2	I	**BURST READY.** Active low. Burst Ready, like $\overline{RDY}$, indicates that the external system has presented valid data on the data pins in response to a Read, or that the external system has accepted the 82596 data in response to a Write request. Also, like $\overline{RDY}$, this signal is ignored at the end of the first clock in a bus cycle. If the 82596 can still receive data from the previous cycle, ADS will not be asserted in the next clock cycle; however, Address and Byte Enable will change to reflect the next data item expected by the 82596. $\overline{BRDY}$ will be sampled during each succeeding clock and if active, the data on the pins will be strobed to the 82596 or to external memory (read/write). $\overline{BRDY}$ operates exactly like READY during the last data cycle of a burst sequence and during nonburstable cycles.
$\overline{BLAST}$	128	O	**BURST LAST.** A signal (active low) on this tri-state pin indicates that the burst cycle is finished and when $\overline{BRDY}$ is next returned it will be treated as a normal ready; i.e., another set of addresses will be driven with $\overline{ADS}$ or the bus will go idle. $\overline{BLAST}$ is not asserted if the bus is not acquired.
AHOLD	117	I	**ADDRESS HOLD.** This hold signal is active high, it allows another bus master to access the 82596 address bus. In a system where an 82596 and an i486 processor share the local bus, AHOLD allows the cache controller to make a cache invalidation cycle while the 82596 holds the address lines. In response to a signal on this pin, the 82596 immediately (i.e. during the next clock) stops driving the entire address bus (A31–A2); the rest of the bus can remain active. For example, data can be returned for a previously specified bus cycle during Address Hold. The 82596 will not begin another bus cycle while AHOLD is active.
$\overline{BOFF}$	116	I	**BACKOFF.** This signal is active low, it informs the 82596 that another bus master requires access to the bus before the 82596 bus cycle completes. The 82596 immediately (i.e. during the next clock) floats its bus. Any data returned to the 82596 while $\overline{BOFF}$ is asserted is ignored. $\overline{BOFF}$ has higher priority than $\overline{RDY}$ or $\overline{BRDY}$; if two such signals are returned in the same clock period, $\overline{BOFF}$ is given preference. The 82596 remains in Hold until $\overline{BOFF}$ goes high, then the 82596 resumes its bus cycle by driving out the address and status, and asserting $\overline{ADS}$.
$\overline{LOCK}$	126	O	**LOCK.** This tri-state pin is used to distinguish locked and unlocked bus cycles. $\overline{LOCK}$ generates a semaphore handshake to the CPU. $\overline{LOCK}$ can be active for several memory cycles, it goes active during the first locked memory cycle (t1) and goes inactive at the last locked cycle (t2). This line is floated after a Reset or when the bus is not acquired. $\overline{LOCK}$ can be disabled via the sysbus byte in software.

PIN DESCRIPTIONS (Continued)

Symbol	PQFP Pin No.	Type	Name and Function
$\overline{BS16}$	129	I	**BUS SIZE.** This signal allows the 82596CA to work with either 16- or 32-bit bytes. Inserting $\overline{BS16}$ low causes the 82596 to perform two 16-bit memory accesses when transferring 32-bit data. In little endian mode the D15–D0 lines are driven when $\overline{BS16}$ is inserted, in Big Endian mode the D31–D16 lines are driven.
HOLD	123	O	**HOLD.** The HOLD signal is active high, the 82596 uses it to request local bus mastership. In normal operation HOLD goes inactive before HLDA. The 82596 can be forced off the bus by deasserting HLDA or if the bus throttle timers expire.
HLDA	118	I	**HOLD ACKNOWLEDGE.** The HLDA signal is active high, it indicates that bus mastership has been given to the 82596. HLDA is internally synchronized; after HOLD is detected low, the CPU drives HLDA low. **NOTE** *Do not connect HLDA to V_{CC}—it will cause a deadlock.* A user wanting to give the 82596 permanent access to the bus should connect HLDA to HOLD. If HLDA goes inactive before HOLD, the 82596 will release the bus (by deasserting HOLD) within a maximum of within a specified number of bus cycles as specified in the 82596 User's Manual.
BREQ	115	I	**BUS REQUEST.** This signal, when configured to an externally activated mode, is used to trigger the bus throttle timers.
$\overline{PORT}$	3	I	**PORT.** When this signal is received, the 82596 latches the data on the data bus into an internal 32-bit register. When the CPU is asserting this signal it can write into the 82596 (via the data bus). This pin must be activated twice during all CPU Port access commands.
RESET	69	I	**RESET.** This active high, internally synchronized signal causes the 82596 to terminate current activity. The signal must be high for at least five system clock cycles. After five system clock cycles and four $\overline{TxC}$ clock cycles the 82596 will execute a Reset when it receives a high RESET signal. When RESET returns to low the 82596 waits for the first CA signal and then begins the initialization sequence.
LE/$\overline{BE}$	65	I	**LITTLE ENDIAN/BIG ENDIAN.** This dual-function pin is used to select byte ordering. When LE/$\overline{BE}$ is high, little endian byte ordering is used; when low, big endian byte ordering is used for data in frames (bytes) and for control (SCB, RFD, CBL, etc).
CA	119	I	**CHANNEL ATTENTION.** The CPU uses this pin to force the 82596 to begin executing memory resident Command blocks. The CA signal is internally synchronized. The signal must be high for at least one system clock. It is latched internally on the high to low edge and then detected by the 82596. The first CA after a Reset forces the 82596 into the initialization sequence beginning at location 00FFFFF6h or an SCP address written to the 82596 using CPU Port access. All subsequent CA signals cause the 82596 to begin executing new command sequences from the SCB.
INT/$\overline{INT}$	125	O	**INTERRUPT.** A high signal on this pin notifies the CPU that the 82596 is requesting an interrupt. This signal is an edge triggered interrupt signal, and can be configured to be active high or low.

PIN DESCRIPTIONS (Continued)

Symbol	PQFP Pin No.	Type	Name and Function
V_{CC}	18 Pins		**POWER.** +5 V ±10%.
V_{SS}	18 Pins		**GROUND.** 0 V.
TxD	54	O	**TRANSMIT DATA.** This pin transmits data to the serial link. It is high when not transmitting.
$\overline{TxC}$	64	I	**TRANSMIT CLOCK.** This signal provides the fundamental timing for the serial subsystem. The clock is also used to transmit data synchronously on the TxD pin. For NRZ encoding, data is transferred to the TxD pin on the high to low clock transition. For Manchester encoding, the transmitted bit center is aligned with the low to high transition. Transmit clock must always be running for proper device operation.
$\overline{LPBK}$	58	O	**LOOPBACK.** This TTL-level control signal enables the loopback mode. In this mode serial data on the TxD input is routed through the 82C501 internal circuits and back to the RxD output without driving the transceiver cable. To enable this signal, both internal and external loopback need to be set with the Configure command.
RxD	60	I	**RECEIVE DATA.** This pin receives NRZ serial data only. It must be high when not receiving.
$\overline{RxC}$	59	I	**RECEIVE CLOCK.** This signal provides timing information to the internal shifting logic. For NRZ data the state of the RxD pin is sampled on the high to low transition of the clock.
$\overline{RTS}$	57	O	**REQUEST TO SEND.** When this signal is low the 82596 informs the external interface that it has data to transmit. It is forced high after a Reset or when transmission is stopped.
$\overline{CTS}$	62	I	**CLEAR TO SEND.** An active-low signal that enables the 82596 to send data. It is normally used as an interface handshake to $\overline{RTS}$. Asserting $\overline{CTS}$ high stops transmission. $\overline{CTS}$ is internally synchronized. If $\overline{CTS}$ goes inactive, meeting the setup time to the $\overline{TxC}$ negative edge, the transmission will stop and $\overline{RTS}$ will go inactive within, at most, two $\overline{TxC}$ cycles.
$\overline{CRS}$	63	I	**CARRIER SENSE.** This signal is active low, it is used to notify the 82596 that traffic is on the serial link. It is only used if the 82596 is configured for external Carrier Sense. In this configuration external circuitry is required for detecting traffic on the serial link. $\overline{CRS}$ is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CRSF = 0).
$\overline{CDT}$	61	I	**COLLISION DETECT.** This active-low signal informs the 82596 that a collision has occurred. It is only used if the 82596 is configured for external Collision Detect. External circuitry is required for collision detection. $\overline{CDT}$ is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CDTF = 0).

82596 AND HOST CPU INTERACTION

The 82596CA and the host CPU communicate through shared memory. Because of its on-chip DMA capability, the 82596 can make data block transfers (buffers and frames) independently of the CPU; this greatly reduces the CPU byte transfer overhead.

The 82596 is a multitasking coprocessor that comprises two independent logical units—the Command Unit (CU) and the Receive Unit (RU). The CU executes commands from shared memory. The RU handles all activities related to frame reception. The independence of the CU and RU enables the 82596 to engage in both activities simultaneously—the CU can fetch and execute commands from memory while the RU is storing received frames in memory. The CPU is only involved with this process after the CU has executed a sequence of commands or the RU has finished storing a sequence of frames.

The CPU and the 82596 use the hardware signals Interrupt (INT) and Channel Attention (CA) to initiate communication with the System Control Block (SCB), see Figure 4. The 82596 uses INT to alert the CPU of a change in the contents of the SCB, the CPU uses CA to alert the 82596.

The 82596 has a CPU Port Access state that allows the CPU to execute certain functions without accessing memory. The 82596 $\overline{\text{PORT}}$ pin and data bus pins are used to enable this feature. The CPU can directly activate four operations when the 82596 is in this state.

- Write an alternative System Configuration Pointer (SCP). This can be used when the 82596 cannot use the default SCP address space.
- Write a different Dump Command Pointer and execute Dump. This can be used for troubleshooting No Response problems.
- The CPU can reset the 82596 via software without disturbing the rest of the system.
- A self-test can be used for board testing; the 82596 will execute a self-test and write the results to memory.

82596 BUS INTERFACE

The 82596CA has bus interface timings and pin definitions that are compatible with Intel's 32-bit i486 microprocessor. This eliminates the need for additional bus interface logic. Operating at 33 MHz, the 82596's bus bandwidth can be as high as 106 MB/s. Since Ethernet only requires 1.25 MB/s, this leaves a considerable amount of bandwidth for the CPU. The 82596 also has a bus throttle to regulate its use of the bus. Two timers can be programmed through the SCB: one controls the maximum time the 82596 can remain on the bus, the other controls the time the 82596 must stay off the bus (see Figure 5). The bus throttle can be programmed to trigger internally with HLDA or externally with BREQ. These timers can restrict the 82596 HOLD activation time and improve bus utilization.

82596 MEMORY ADDRESSING

The 82596 has a 32-bit memory address range, which allows addressing up to four gigabytes of memory. The 82596 has three memory addressing modes (see Table 1).

- **82586 Mode.** The 82596 has a 24-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-KB memory segment. Transmit and Receive buffers can reside in a 24-bit address space.
- **32-Bit Segmented Mode.** The 82596 has a 32-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-KB memory segment. Transmit and Receive buffers can reside in a 32-bit address space.
- **Linear Mode.** The 82596 has a 32-bit memory address range. Any memory structure can reside anywhere within the 32-bit memory address range.

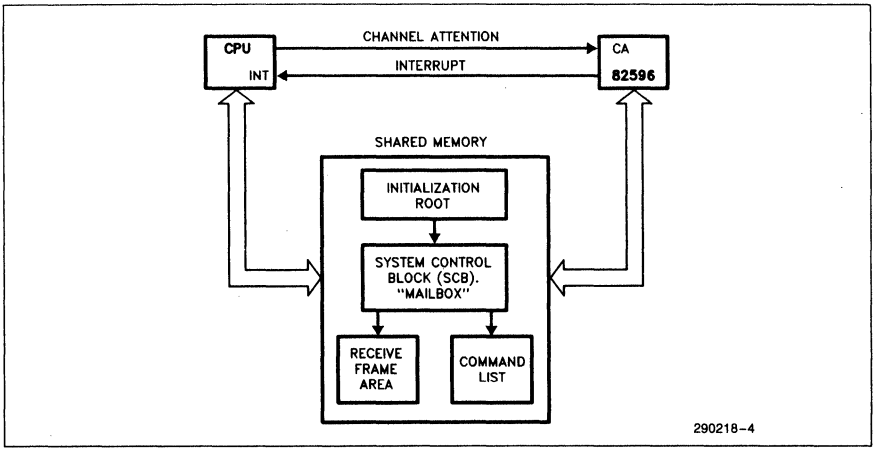


Figure 4. 82596 and Host CPU Intervention

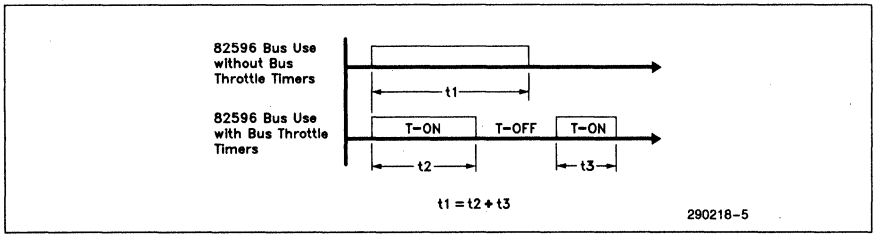


Figure 5. Bus Throttle Timers

Table 1. 82596 Memory Addressing Formats

Pointer or Offset	Operation Mode		
	82586	32-Bit Segmented	Linear
ISCP Address	24-Bit Linear	32-Bit Linear	32-Bit Linear
SCB Address	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Command Block Pointers	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Rx Frame Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Tx Frame Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Rx Buffer Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Tx Buffer Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Rx Buffers	24-Bit Linear	32-Bit Linear	32-Bit Linear
Tx Buffers	24-Bit Linear	32-Bit Linear	32-Bit Linear

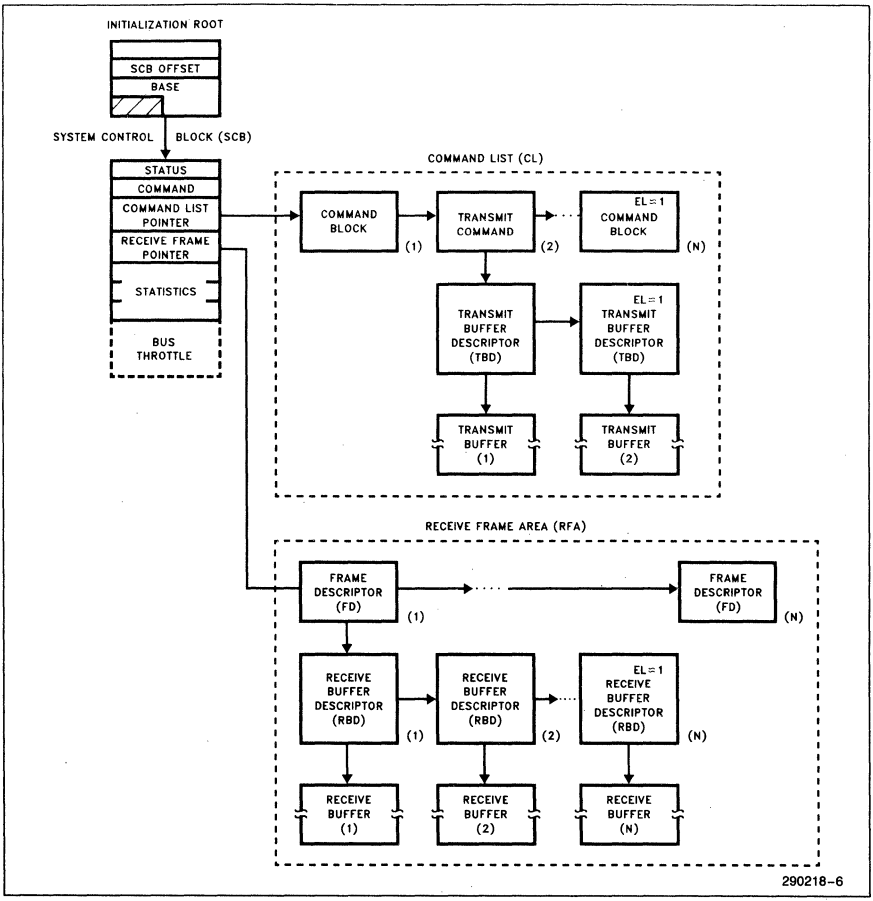


Figure 6. 82596 Shared Memory Structure

82596 SYSTEM MEMORY STRUCTURE

The Shared Memory structure consists of four parts: the Initialization Root, the System Control Block, the Command List, and the Receive Frame Area (see Figure 6).

The Initialization Root is in an established location known to the host CPU and the 82596 (00FFFFF6h). However, the CPU can establish the Initialization Root in another location by using the CPU Port access. This root is accessed during initialization, and points to the System Control Block.

The System Control Block serves as a bidirectional mail drop for the host CPU and the 82596 CU and RU. It is the central point through which the CPU and the 82596 exchange control and status information. The SCB has two areas. The first contains instructions from the CPU to the 82596. These include: control of the CU and RU (Start, Abort, Suspend, and Resume), a pointer to the list of CU commands, a pointer to the Receive Frame Area, a set of Interrupt Acknowledge bits, and the T-ON and T-OFF timers for the bus throttle. The second area contains status information the 82596 is sending to the CPU. Such as, the CU and RU states (Idle, Active

Ready, Suspended, No Receive Resources, etc.), interrupt bits (Command Completed, Frame Received, CU Not Ready, and RU Not Ready), and statistical counters.

The Command List functions as a program for the CU; individual commands are placed in memory units called Command Blocks (CBs). These CBs contain the parameters and status of specific high-level commands called Action Commands; e.g., Transmit or Configure.

Transmit causes the 82596 to transmit a frame. The Transmit CB contains the destination address, the length field, and a pointer to a list of linked buffers holding the frame that is to be constructed from several buffers scattered throughout memory. The Command Unit operates without CPU intervention; the DMA for each buffer, and the prefetching of references to new buffers, is performed in parallel. The CPU is notified only after a transmission is complete.

The Receive Frame Area is a list of Free Frame Descriptors (descriptors not yet used) and a list of user-prepared buffers. Frames arrive at the 82596 unsolicited; the 82596 must always be ready to receive and store them in the Free Frame Area. The Receive Unit fills the buffers when it receives frames, and reformats the Free Buffer List into received-frame structures. The frame structure is, for all practical purposes, identical to the format of the frame to be transmitted. The first Frame descriptor is referenced by the SCB. Unless the 82596 is configured to Save Bad Frames, the frame descriptor, and the associated buffer descriptor, which is wasted when a bad frame is received, are automatically reclaimed and returned to the Free Buffer List.

Receive buffer chaining (storing incoming frames in a linked buffer list) significantly improves memory utilization. Without buffer chaining, the user must allocate consecutive blocks of memory, each capable of containing a maximum frame (for Ethernet, 1518 bytes). Since an average frame is about 200 bytes, this is very inefficient. With buffer chaining, the user can allocate small buffers and the 82596 will only use those that are needed.

Figure 7 A–D illustrates how the 82596 uses the Receive Frame Area. Figure 7A shows an unused Receive Frame Area composed of Free Frame Descriptors and Free Receive Buffers prepared by the user. The SCB points to the first Frame Descriptor of the Frame Descriptor List. Figure 7B shows the same Receive Frame Area after receiving one frame. This first frame occupies two Receive Buffers and one Frame Descriptor—a valid received frame will only occupy one Frame Descriptor. After receiving this frame the 82596 sets the next Free Frame Descriptor RBD pointer to the next Free RBD. Figure 7C shows the RFA after receiving a second frame. In this example the second frame occupies only one Receive Buffer and one RFD. The 82596 again sets the RBD pointer. This process is repeated again in Figure 7D, showing the reception of another frame using one Receive Buffer; in this example there is an extra Frame Descriptor.

TRANSMIT AND RECEIVE MEMORY STRUCTURES

There are three memory structures for reception and transmission. The 82586 memory structure, the Flexible memory structure, and the Simplified memory structure. The 82586 mode is selected by configuring the 82596 during initialization. In this mode all the 82596 memory structures are compatible with the 82586 memory structures.

When the 82596 is not configured to the 82586 mode, the other two memory structures, Simplified and Flexible, are available for transmitting and receiving. These structures can be selected on a frame-by-frame basis by setting the S/F bit in the Transmit Command and the Receive Frame Descriptor (see Figures 29, 30, 41, and 42). The Simplified memory structure offers a simple structure for ease of programming (see Figure 8). All information about a frame is contained in one structure; for example, during reception the RFD and data field are contained in one structure.

The Flexible memory structure (see Figure 9) has a control field that allows the programmer to specify the amount of receive data the RFD will contain for receive operations and the amount of transmit data the Transmit Command Block will contain for transmit operations. For example, when the control field in the RFD is set to 20 bytes during a reception, the first 20 bytes of the data field are stored in the RFD (6 bytes of destination address, 6 bytes of source address, 2 bytes of length field, and 6 bytes of data) and the remainder of the data field is stored in the Receive Data Buffers. This is useful for capturing frame headers when header information is contained in the data field. The header information can then be automatically stored in the RFD partitioned from the Receive Data Buffer.

The control field can also be used for the Transmit Command when the Flexible memory structure is used. The quantity of data field bytes to be transmitted from the Transmit Command Block is specified by the variable control field.

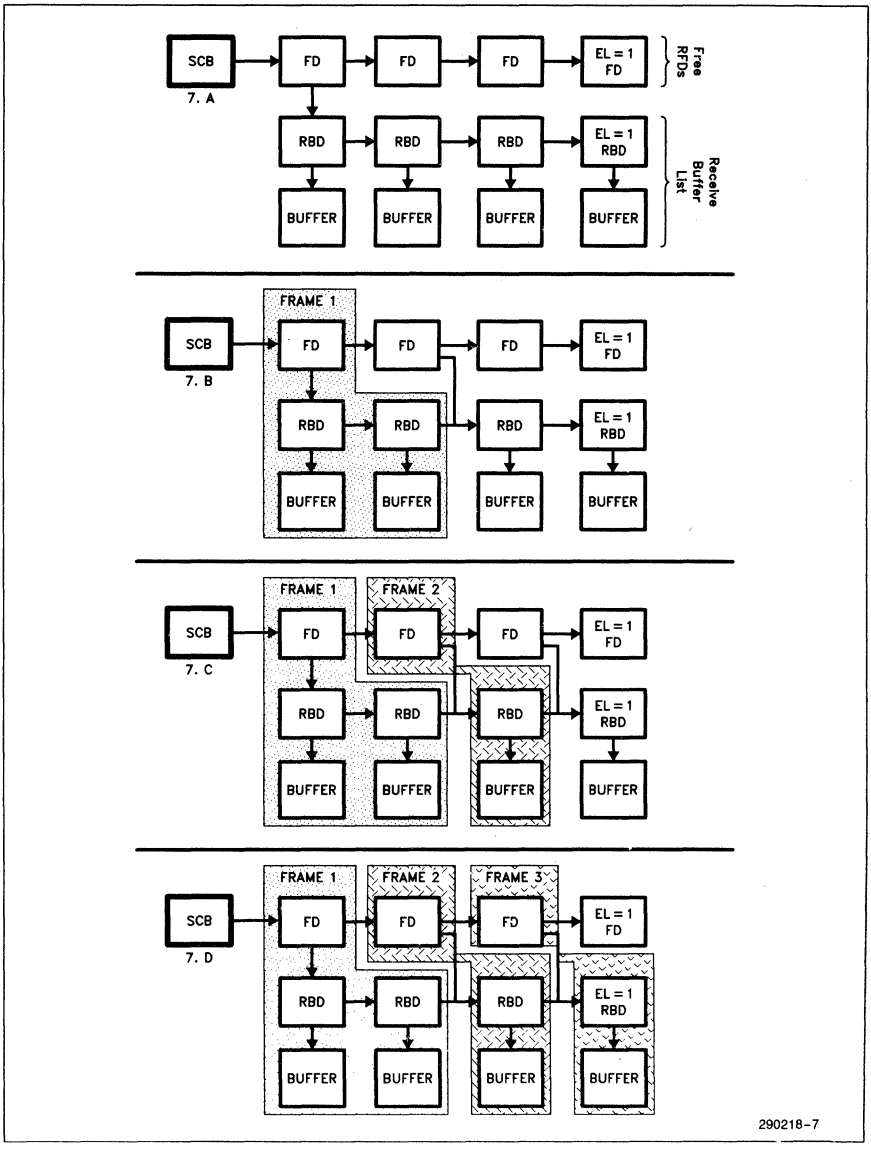


Figure 7. Frame Reception in the RFA

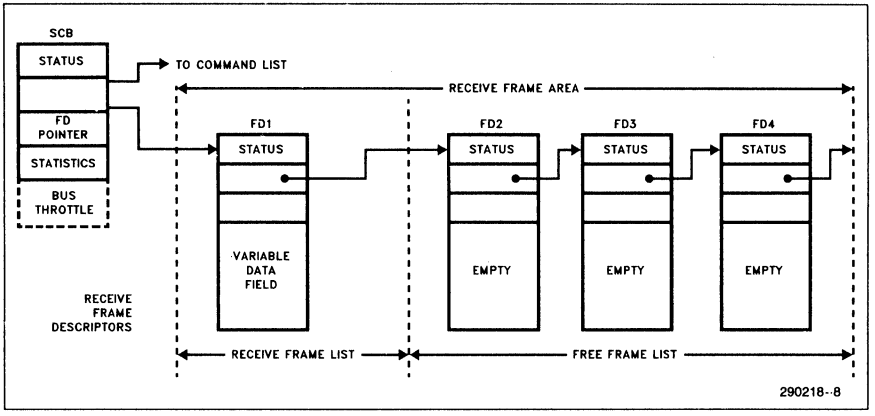


Figure 8. Simplified Memory Structure

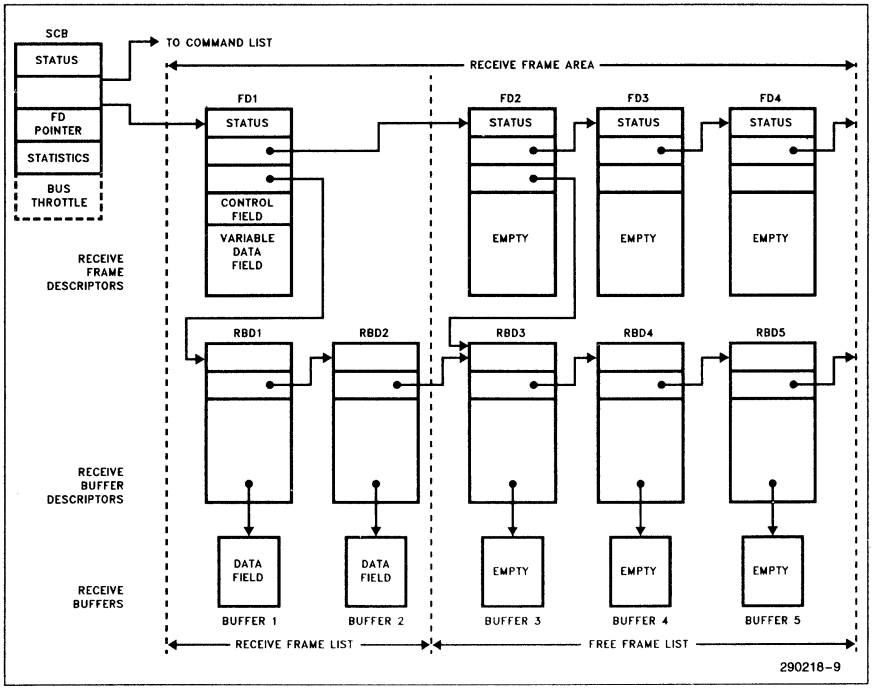


Figure 9. Flexible Memory Structure

TRANSMITTING FRAMES

The 82596 executes high-level Action Commands from the Command List in system memory. Action Commands are fetched and executed in parallel with the host CPU operation, thereby significantly improving system performance. The format of the Action Commands is shown in Figure 10. Figure 28 shows the 82586 mode, and Figures 29 and 30 show the command formats of the Linear and 32-bit Segmented modes.

A single Transmit command contains, as part of the command-specific parameters, the destination address and length field of the transmitted frame and a pointer to buffer area in memory containing the data portion of the frame. The data field is contained in a memory data structure consisting of a buffer descriptor (BD) and a data buffer—or a linked list of buffer descriptors and buffers—as shown in Figure 11.

Multiple data buffers can be chained together using the BDs. Thus, a frame with a long data field can be transmitted using several (shorter) data buffers chained together. This chaining technique allows the system designer to develop efficient buffer management.

The 82596 automatically generates the preamble (alternating 1s and 0s) and start frame delimiter, fetches the destination address and length field from the Transmit command, inserts its unique address as the source address, fetches the data field specified by the Transmit command, and computes and appends the CRC to the end of the frame (see Figure 12). In the Linear and 32-bit Segmented mode the CRC can be optionally inserted on a frame-by-frame basis by setting the NC bit in the Transmit Command Block (see Figures 29 and 30).

The 82596 can be configured to generate two types of start and end frame delimiters—End of Carrier (EOC) or HDLC. In EOC mode the start frame delimiter is 10101011 and the end frame delimiter is indicated by the lack of a signal after the last bit of the frame check sequence field has been transmitted. In EOC mode the 82596 can be configured to extend short frames by adding pad bytes (7Eh) during transmission, according to the length field. In HDLC mode the 82596 will generate the 01111110 flag for the start and end frame delimiters, and do standard bit stuffing and stripping. Furthermore, the 82596 can be configured to pad frames shorter than the specified minimum frame length by appending the appropriate number of flags to the end of the frame.

When a collision occurs, the 82596 manages the jam, random wait, and retry processes, reinitializing DMA pointers without CPU intervention. Multiple frames can be sent by linking the appropriate number of Transmit commands together. This is particularly useful when transmitting a message larger than the maximum frame size (1518 bytes for Ethernet).

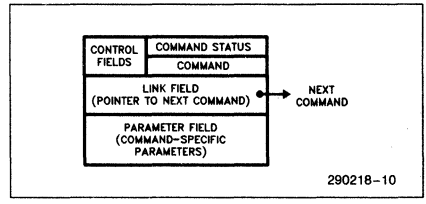


Figure 10. Action Command Format

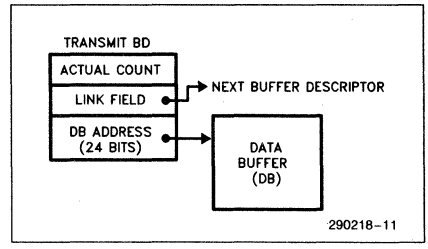


Figure 11. Data Buffer Descriptor and Data Buffer Structure

PREAMBLE	START FRAME DELIMITER	DESTINATION ADDRESS	SOURCE ADDRESS	LENGTH FIELD	DATA FIELD	FRAME CHECK SEQUENCE	END FRAME DELIMITER

Figure 12. Frame Format

RECEIVING FRAMES

To reduce CPU overhead, the 82596 is designed to receive frames without CPU supervision. The host CPU first sets aside an adequate receive buffer space and then enables the 82596 Receive Unit. Once enabled, the RU watches for arriving frames and automatically stores them in the Receive Frame Area (RFA). The RFA contains Receive Frame Descriptors, Receive Buffer Descriptors, and Data Buffers (see Figure 13). The individual Receive Frame Descriptors make up a Receive Descriptor List (RDL) used by the 82596 to store the destination and source addresses, the length field, and the status of each frame received (see Figure 14).

Once enabled, the 82596 checks each passing frame for an address match. The 82596 will recognize its own unique address, one or more multicast addresses, or the broadcast address. If a match is found the 82596 stores the destination and source addresses and the length field in the next available RFD. It then begins filling the next available Data Buffer on the FBL, which is pointed to by the current RFD, with the data portion of the incoming frame. As one Data Buffer is filled, the 82596 automatically fetches the next DB on the FBL until the entire frame is received. This buffer chaining technique is particularly memory efficient because it allows the system designer to set aside buffers to fit frames much shorter than the maximum allowable frame length. If AL-LOC = 1, or if the flexible memory structure is used, the addresses and length field can be placed in the Receive Buffer.

Once the entire frame is received without error, the 82596 does the following housekeeping tasks.

- The actual count field of the last Buffer Descriptor used to hold the frame just received is updated with the number of bytes stored in the associated Data Buffer.
- The next available Receive Frame Descriptor is fetched.
- The address of the next available Buffer Descriptor is written to the next available Receive Frame Descriptor.
- A frame received interrupt status bit is posted in the SCB.
- An interrupt is sent to the CPU.

If a frame error occurs, for example a CRC error, the 82596 automatically reinitializes its DMA pointers and reclaims any data buffers containing the bad frame. The 82596 will continue to receive frames without CPU help as long as Receive Frame Descriptors and Data Buffers are available.

82596 NETWORK MANAGEMENT AND DIAGNOSTICS

The behavior of data communication networks is normally very complex because of their distributed and asynchronous nature. It is particularly difficult to pinpoint a failure when it occurs. The 82596 has extensive diagnostic and network management functions that help improve reliability and testability. The 82596 reports on the following events after each frame is transmitted.

- Transmission successful.
- Transmission unsuccessful. Lost Carrier Sense.
- Transmission unsuccessful. Lost Clear to Send.
- Transmission unsuccessful. A DMA underrun occurred because the system bus did not keep up with the transmission.
- Transmission unsuccessful. The number of collisions exceeded the maximum allowed.
- Number of Collisions. The number of collisions experienced during the frame.
- Heartbeat Indicator. This indicates the presence of a heartbeat during the last Interframe Spacing (IFS) after transmission.

When configured to Save Bad Frames the 82596 checks each incoming frame and reports the following errors.

- CRC error. Incorrect CRC in a properly aligned frame.
- Alignment error. Incorrect CRC in a misaligned frame.
- Frame too short. The frame is shorter than the value configured for minimum frame length.
- Overrun. Part of the frame was not placed in memory because the system bus did not keep up with incoming data.
- Out of buffer. Part of the frame was discarded because of insufficient memory storage space.
- Receive collision. A collision was detected during reception.
- Length error. A frame not matching the frame length parameter was detected.

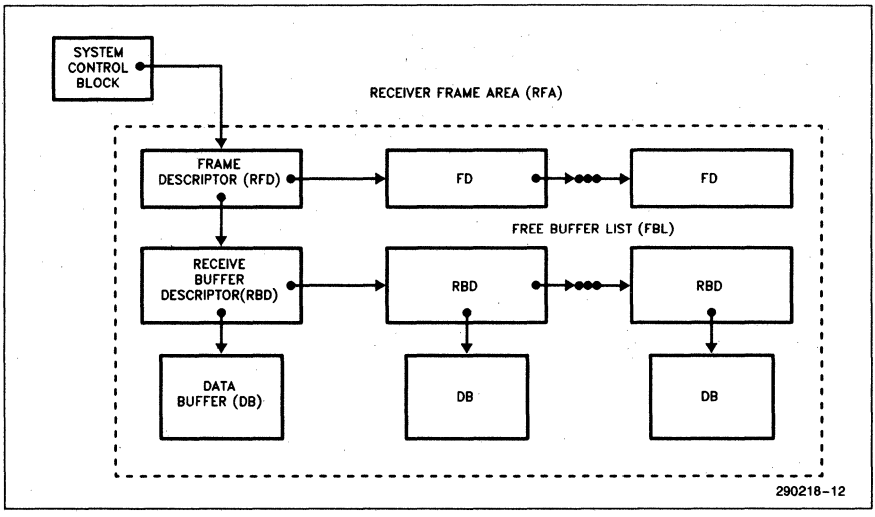


Figure 13. Receive Frame Area Diagram

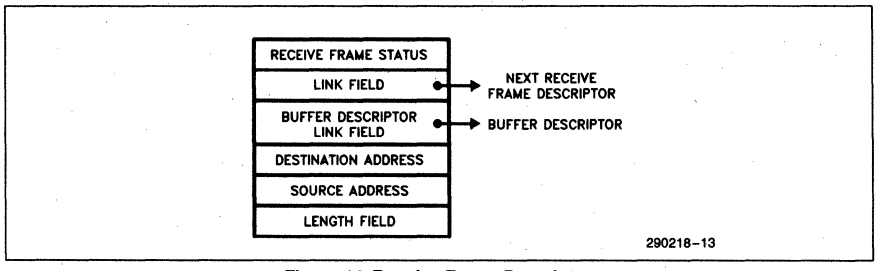


Figure 14. Receive Frame Descriptor

NETWORK PLANNING AND MAINTENANCE

To properly plan, operate, and maintain a communication network, the network management entity must accumulate information on network behavior. The 82596 provides a rich set of network-wide diagnostics that can serve as the basis for a network management entity.

Information on network activity is provided in the status of each frame transmitted. The 82596 reports the following activity indicators after each frame.

- Number of collisions. The number of collisions the 82596 experienced while attempting to transmit the frame.
- Deferred transmission. During the first transmission attempt the 82596 had to defer to traffic on the link.

The 82596 updates its 32-bit statistical counters after each received frame that both passes address filtering and is longer than the Minimum Frame Length configuration parameter. The 82596 reports the following statistics.

- CRC errors. The number of well-aligned frames that experienced a CRC error.
- Alignment errors. The number of misaligned frames that experienced a CRC error.
- No resources. The number of frames that were discarded because of insufficient resources for reception.
- Overrun errors. The number of frames that were not completely stored in memory because the system bus did not keep up with incoming data.
- Receive Collision counter. The number of collisions detected during receive.
- Short Frame counter. The number of frames that were discarded because they were shorter than the configured minimum frame length.

The 82596 can be configured to Promiscuous mode. In this mode it captures all frames transmitted on the network without checking the Destination Address. This is useful when implementing a monitoring station to capture all frames for analysis.

A useful method of capturing frame headers is to use the Simplified memory mode, configure the 82596 to Save Bad Frames, and configure the 82596 to Promiscuous mode with space in the RFD allocated for specific number of receive data bytes. The 82596 will receive all frames and put them in the RFD. Frames that exceed the available space in the RFD will be truncated, the status will be updated, and the 82596 will retrieve the next RFD. This allows the user to capture the initial data bytes of each frame (for instance, the header) and discard the remainder of the frame.

The 82596 also has a monitor mode for network analysis. During normal operation the receive function enables the 82596 to receive frames that pass address filtering. These frames must have the Start of Frame Delimiter (SFD) field and must be longer than the absolute minimum frame length of 5 bytes (6 bytes in case of Multicast address filtering). Contents and status of the received frames are transferred to memory. The monitor function enables the 82596 to simply evaluate the incoming frames. The 82596 can monitor the frames that pass or do not pass the address filtering. It can also monitor frames which do not have the SFD fields. The 82596 can be configured to only keep statistical information about monitor frames. Three options are available in the Monitor mode. These options are selected by the two monitor mode configuration bits available in the configuration command.

When the first option is selected, the 82596 receives good frames that pass address filtering and transfers them to memory while monitoring frames that do not pass address filtering or are shorter than the minimum frame size (these frames are not transferred to memory). When this option is used the 82596 updates six counters: CRC errors, alignment errors, no resource errors, overrun errors, short frames and total good frames received.

When the second option is selected, the receive function is completely disabled. The 82596 monitors only those frames that pass address filterings and meet the minimum frame length requirement. When this option is used the 82596 updates six counters: CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected and total good frames.

When the third option is selected, the receive function is completely disabled. The 82596 monitors all frames, including frames that do not have a Start Frame Delimiter. When this option is used the 82596 updates six counters: CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected and total good frames.

STATION DIAGNOSTICS AND SELF-TEST

The 82596 provides a large set of diagnostic and network management functions. These include internal and external loopback and time domain reflectometry for locating fault points in the network cable. The 82596 ensures software reliability by dumping the contents of the 82596 internal registers into system memory. The 82596 has a self-test mode that enables it to run an internal self-test and place the results in system memory.

82586 SOFTWARE COMPATIBILITY

The 82596 has a software-compatible state in which all its memory structures are compatible with the 82586 memory structure. This includes all the Action Commands, the Receive Frame Area (including the RFD, Buffer Descriptors, and Data Buffers), the System Control Block, and the initialization procedures. There are two minor differences between the 82596 in the 82586-Compatible memory structure and the 82586.

- When the internal and external loopback bits in the Configure command are set to 11 the 82596 is in external loopback and the $\overline{LPBK}$ pin is activated; in the 82586 this situation would produce internal loopback.
- During a Dump command both the 82596 and 82586 dump the same number of bytes; however, the data format is different.

INITIALIZING THE 82596

A Reset command is issued to the 82596 to prepare it for normal operation. The 82596 is initialized through two data structures that are addressed by two pointers, the System Configuration Pointer (SCP) and the Intermediate System Configuration Pointer (ISCP). The initialization procedure begins when a Channel Attention signal is asserted after RESET. The 82596 uses the address of the double word that contains the SCP as a default—00FFFFF4h. Before the CA signal is asserted this default address can be changed to any other available address by asserting the $\overline{PORT}$ pin and providing the desired address over the $D_{31}-D_4$ pins of the address bus. Pins D_3-D_0 must be 0010; i.e., any alternative address must be aligned to 16-byte boundaries. All addresses sent to the 82596 must be word aligned, which means that all pointers and memory structures must start on an even address (A_0 = zero).

SYSTEM CONFIGURATION POINTER (SCP)

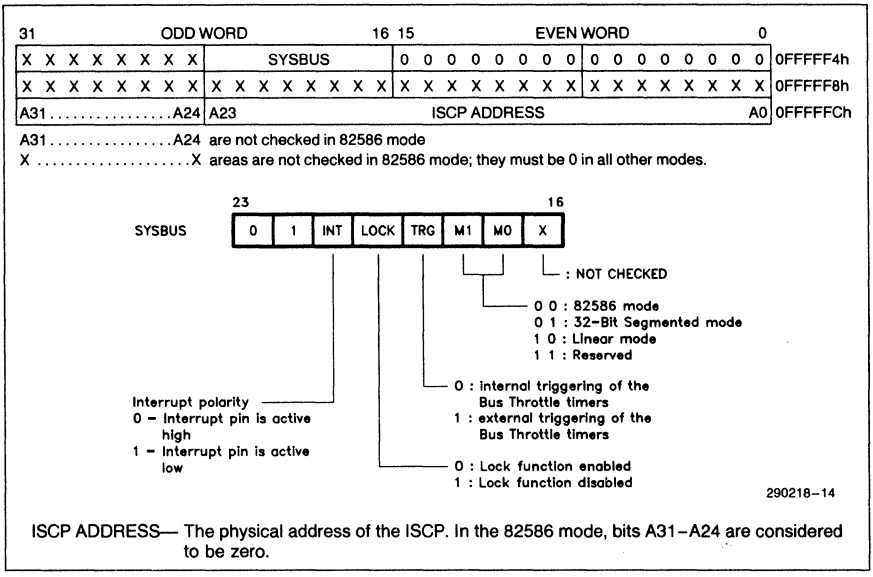
The SCP contains the sysbus byte and the location of the next structure of the initialization process, the ISCP. The following parameters are selected in the SYSBUS.

- The 82596 operation mode.
- The Bus Throttle timer triggering method.
- Lock enabled.
- Interrupt polarity.

Byte ordering is determined by the LE/$\overline{BE}$ pin. LE/$\overline{BE}$ = 1 selects Little Endian byte ordering and LE/$\overline{BE}$ = 0 selects Big Endian byte ordering.

NOTE:
In the following, X indicates a bit not checked 82586 mode. This bit must be set to 0 in all other modes.

The following diagram illustrates the format of the SCP.



ISCP ADDRESS— The physical address of the ISCP. In the 82586 mode, bits A31–A24 are considered to be zero.

Figure 15. The System Configuration Pointer

Writing the Sysbus

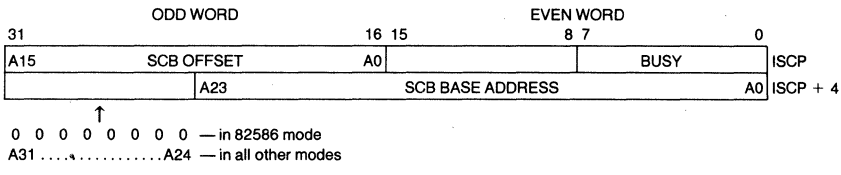
When writing the sysbus byte it is important to pay attention to the byte order.

- When a Little Endian processor is used, the sysbus byte is located at byte address 00FFFFF6h (or address $n+2$ if an alternative SCP address n was programmed).
- When a processor using Big Endian byte ordering is used, the sysbus, alternative SCP, and ISCP addresses will be different.
  - The sysbus byte is located at 00FFFFF5h.
  - If an alternative SCP address is programmed, the sysbus byte should be at byte address $n+1$.

INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP)

The ISCP indicates the location of the System Control Block. Often the SCP is in ROM and the ISCP is in RAM. The CPU loads the SCB address (or an equivalent data structure) into the ISCP and asserts CA. This Channel Attention signal causes the 82596 to begin its initialization procedure and to get the SCB address from the ISCP and SCP. In 82586 and 32-bit Segmented modes the SCP base address is also the base address of all Command Blocks, Frame Descriptors, and Buffer Descriptors (but not buffers). All these data structures must reside in one 64-KB segment; however, in Linear mode no such limitation is imposed.

The following diagram illustrates the ISCP format.

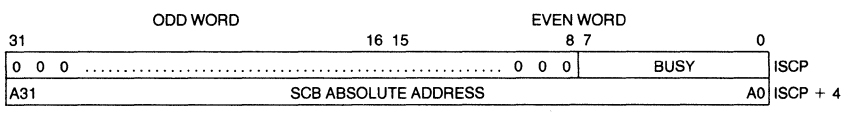


BUSY — Indicates that the 82596 is being initialized. The CPU sets the ISCP to 01h before it gives the first CA to the 82596. The ISCP is cleared by the 82596 after the SCB base and offset are read. Note that the most significant byte of the first word of the ISCP is not modified when BUSY is cleared.

SCB OFFSET— This 16-bit quantity specifies the offset portion of the address of the SCB.

SCB BASE — Specifies the base portion of the address of the SCB. The base of SCB is also the base of all 82596 Command Blocks, Frame Descriptors and Buffer Descriptors. In the 82586 mode, bits A31–A24 are considered to be zero.

Figure 16. The Intermediate System Configuration Pointer—82586 and 32-Bit Segmented Modes



BUSY — Indicates that the 82596 is being initialized. The ISCP is set to 01h by the CPU before its first CA to the 82596. It is cleared by the 82596 after the SCB address is read.

SCB ADDRESS— This 32-bit quantity specifies the physical address of the SCB.

Figure 17. The Intermediate System Configuration Pointer—Linear Mode.

INITIALIZATION PROCESS

The CPU sets up the SCP, ISCP, and the SCB structures, and, if desired, an alternative SCP address. It also sets BUSY to 01h. The 82596 is initialized when a Channel Attention signal follows a Reset signal, causing the 82596 to access the System Configuration Pointer. The sysbus byte, the operational mode, the bus throttle timer triggering method, the interrupt polarity, and the state of LOCK are read. After reset the Bus Throttle timers are essentially disabled—the T-ON value is infinite, the T-OFF value is zero. After the SCP is read, the 82596 reads the ISCP and saves the SCB address. In 82586 and 32-bit Segmented modes this address is represented as a base address plus the offset (this base address is also the base address of all the control blocks). In Linear mode the base address is also an absolute address. The 82596 clears BUSY, sets CX and CNR to equal 1 in the SCB, clears the SCB command word, sends an interrupt to the CPU, and awaits another Channel Attention signal. RESET configures the 82596 to its default state before CA is asserted.

CONTROLLING THE 82596CA

The host CPU controls the 82596 with the commands, data structures, and methods described in this section. The CPU and the 82596 communicate through shared memory structures. The 82596 contains two independent units: the Command Unit and the Receive Unit. The Command Unit executes commands from the CPU, and the Receive Unit handles frame reception. These two units are controlled and monitored by the CPU through a shared memory structure called the System Control Block (SCB). The CPU and the 82596 use the CA and INT signals to communicate with the SCB.

82596 CPU ACCESS INTERFACE ($\overline{PORT}$)

The 82596 has a CPU access interface that allows the host CPU to do four things.

- Write an alternative System Configuration Pointer address.
- Write an alternative Dump area pointer and perform Dump.
- Execute a software reset.
- Execute a self-test.

The following events initiate the CPU access state.

- Presence of an address on the $D_{31}-D_4$ data bus pins.
- The D_3-D_0 pins are used to select one of the four functions.
- The $\overline{PORT}$ input pin is asserted, as in a regular write cycle.

NOTE.
The SCP Dump and Self-Test addresses must be 16-byte aligned.

The 82596 requires two 16-bit write cycles for a port command. The first write holds the internal machines and reads the first 16 bits; the second activates the $\overline{PORT}$ command and reads the second 16 bits.

The $\overline{PORT}$ Reset is useful when only the 82596 needs to be reset. The CPU must wait for 10-system and 5-serial clocks before issuing another CA to the 82596; this new CA begins a new initialization process.

The Dump function is useful for troubleshooting No Response problems. If the chip is in a No Response state, the $\overline{PORT}$ Dump operation can be executed and a $\overline{PORT}$ Reset can be used to reinitialize the 82596 without disturbing the rest of the system.

The Self-Test function can be used for board testing; the 82596 will execute a self-test and write the results to memory.

Table 2. $\overline{PORT}$ Function Selection

	D31 ..... D4						D0
Function	**Addresses and Results**			**D3**	**D2**	**D1**	**D0**
Reset	A31	Don't Care	A4	0	0	0	0
Self-Test	A31	Self-Test Results Address	A4	0	0	0	1
SCP	A31	Alternative SCP Address	A4	0	0	1	0
Dump	A31	Dump Area Pointer	A4	0	0	1	1

MEMORY ADDRESSING FORMATS

The 82596 accesses memory by 32-bit addresses. There are two types of 32-bit addresses: linear and segmented. The type of address used depends on the 82596 operating mode and the type of memory structure it is addressing. The 82596 has three operating modes.

- 82586 Mode
  - A Linear address is a single 24-bit entity. Address pins A_{31}–A_{24} are always zero.
  - A Segmented address uses a 24-bit base and a 16-bit offset.
- 32-bit Segmented Mode
  - A Linear address is a single 32-bit entity.
  - A Segmented address uses a 32-bit base and a 16-bit offset.

NOTE

In the previous two memory addressing modes, each command header (CB, TBD, RFD, RBD, and SCB) must wholly reside within one segment. If the 82596 encounters a memory structure that does not follow this restriction, the 82596 will fetch the next contiguous location in memory (beyond the segment).

- Linear Mode
  - A Linear address is a single 32-bit entity.
  - There are no Segmented addresses.

Linear addresses are primarily used to address transmit and receive data buffers. In the 82586 and 32-bit Segmented modes, segmented addresses (base plus offset) are used for all Command Blocks, Buffer Descriptors, Frame Descriptors, and System Control Blocks. When using Segmented addresses, only the offset portion of the entity being addressed is specified in the block. The base for all offsets is the same—that of the SCB. See Table 1.

LITTLE ENDIAN AND BIG ENDIAN BYTE ORDERING

The 82596 supports both Little Endian and Big Endian byte ordering for its memory structures.

The 82596 supports Big Endian byte ordering for word and byte entities. Dword entities are not supported with Big Endian byte ordering. This results in slightly different 82596 memory structures for Big Endian operation. These structures are defined in the *82596 User's Manual.*

NOTE

All 82596 memory entities must be word or dword aligned.

An example of a dword entity is a frame descriptor command/status dword, whereas the raw data of the frame are byte entities. Both 32- and 16-bit buses are supported. When a 16-bit bus is used with Big Endian memory organization, data lines D_{15}–D_0 are used. The 82596 has an internal crossover that handles these swap operations.

COMMAND UNIT (CU)

The Command Unit is the logical unit that executes Action Commands from a list of commands very similar to a CPU program. A Command Block is associated with each Action Command. The CU is modeled as a logical machine that takes, at any given time, one of the following states.

- **Idle.** The CU is not executing a command and is not associated with a CB on the list. This is the initial state.
- **Suspended.** The CU is not executing a command; however, it is associated with a CB on the list.
- **Active.** The CU is executing an Action Command and pointing to its CB.

The CPU can affect CU operation in two ways: by issuing a CU Control Command or by setting bits in the Command word of the Action Command.

RECEIVE UNIT (RU)

The Receive Unit is the logical unit that receives frames and stores them in memory. The RU is modeled as a logical machine that takes, at any given time, one of the following states.

- **Idle.** The RU has no memory resources and is discarding incoming frames. This is the initial state.
- **No Resources.** The RU has no memory resources and is discarding incoming frames. This state differs from Idle in that the RU accumulates statistics on the number of discarded frames.
- **Suspended.** The RU has memory available for storing frames, but is discarding them. The suspend state can only be reached if the CPU forces this through the SCB or sets the suspend bit in the RFD.
- **Ready.** The RU has memory available and is storing incoming frames.

The CPU can affect RU operation in three ways: by issuing an RU Control Command, by setting bits in the Frame Descriptor Command word of the frame being received, or by setting the EL bit of the current buffer's Buffer Descriptor.

SYSTEM CONTROL BLOCK (SCB)

The SCB is a memory block that plays a major role in communications between the CPU and the 82596. Such communications include the following.

- Commands issued by the CPU
- Status reported by the 82596

Control commands are sent to the 82596 by writing them into the SCB and then asserting CA. The 82596 examines the command, performs the required action, and then clears the SCB command word. Control commands perform the following types of tasks.

- Operation of the Command Unit (CU). The SCB controls the CU by specifying the address of the Command Block List (CBL) and by starting, suspending, resuming, or aborting execution of CBL commands.
- Operation of the Bus Throttle. The SCB controls the Bus Throttle timers by providing them with new values and sending the Load and Start timer commands. The timers can be operated in both the 32-bit Segmented and Linear modes.
- Reception of frames by the Receive Unit (RU). The SCB controls the RU by specifying the address of the Receive Frame Area and by starting, suspending, resuming, or aborting frame reception.
- Acknowledgment of events that cause interrupts.
- Resetting the chip.

The 82596 sends status reports to the CPU via the System Control Block. The SCB contains four types of status reports.

- The cause of the current interrupts. These interrupts are caused by one or more of the following 82596 events.
  - The Command Unit completes an Action Command that has its I bit set.
  - The Receive Unit receives a frame.
  - The Command Unit becomes inactive.
  - The Receive Unit becomes not ready.
- The status of the Command Unit.
- The status of the Receive Unit.
- Status reports from the 82596 regarding reception of corrupted frames.

Events can be cleared only by CPU acknowledgment. If some events are not acknowledged by the ACK field the Interrupt signal (INT) will be reissued after Channel Attention (CA) is processed. Furthermore, if a new event occurs while an interrupt is set, the interrupt is temporarily cleared to trigger edge-triggered interrupt controllers.

The CPU uses the Channel Attention line to cause the 82596 to examine the SCB. This signal is trailing-edge triggered—the 82596 latches CA on the trailing edge. The latch is cleared by the 82596 before the SCB control command is read.

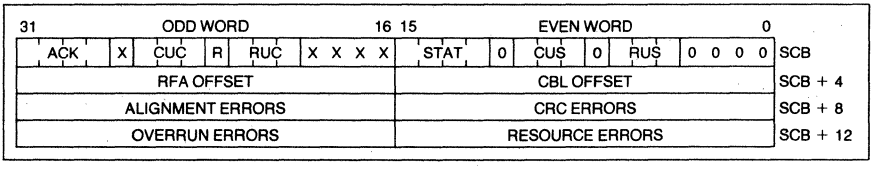


Figure 18. SCB—82586 Mode

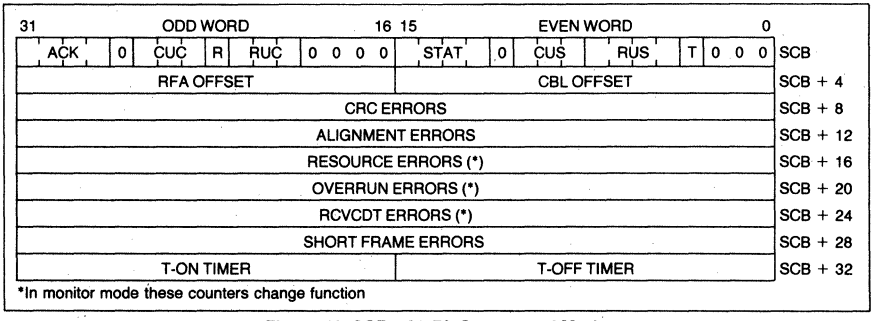


Figure 19. SCB—32-Bit Segmented Mode

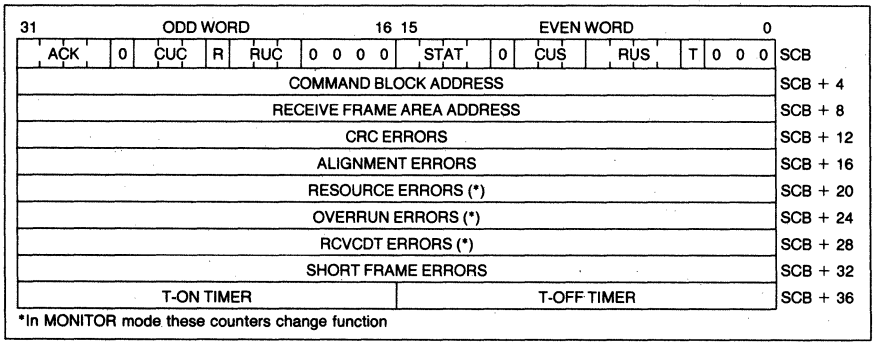


Figure 20. SCB—Linear Mode

Command Word

31								16	
ACK	0	CUC	R	RUC	0	0	0	0	SCB + 2

These bits specify the action to be performed as a result of a CA. This word is set by the CPU and cleared by the 82596. Defined bits are:

Bit 31 ACK-CX — Acknowledges that the CU completed an Action Command.

Bit 30 ACK-FR — Acknowledges that the RU received a frame.

Bit 29 ACK-CNA — Acknowledges that the Command Unit became not active.

Bit 28 ACK-RNR — Acknowledges that the Receive Unit became not ready.

Bits 24–26 CUC — (3 bits) This field contains the command to the Command Unit. Valid values are:

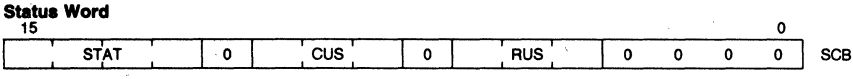
- 0 — NOP (does not affect current state of the unit).
- 1 — Start execution of the first command on the CBL. If a command is executing, complete it before starting the new CBL. The beginning of the CBL is in CBL OFFSET (address).
- 2 — Resume the operation of the Command Unit by executing the next command. This operation assumes that the Command Unit has been previously suspended.
- 3 — Suspend execution of commands on CBL after current command is complete.
- 4 — Abort current command immediately.
- 5 — Loads the Bus Throttle timers so they will be initialized with their new values after the active timer (T-ON or T-OFF) reaches Terminal Count. If no timer is active new values will be loaded immediately. This command is not valid in 82586 mode.
- 6 — Loads and immediately restarts the Bus Throttle timers with their new values. This command is not valid in 82586 mode.
- 7 — Reserved.

Bit 23 RESET — Reset chip (logically the same as hardware RESET).

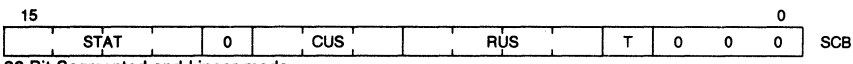
Bits 20–22 RUC — (3 bits) This field contains the command to the Receive Unit. Valid values are:

- 0 — NOP (does not alter current state of unit).
- 1 — Start reception of frames. The beginning of the RFA is contained in the RFA OFFSET (address). If a frame is being received complete reception before starting.
- 2 — Resume frame reception (only when in suspended state).
- 3 — Suspend frame reception. If a frame is being received complete its reception before suspending.
- 4 — Abort receiver operation immediately.
- 5–7 — Reserved.

Status Word



82586 mode



32-Bit Segmented and Linear mode.

Indicates the status of the 82596. This word is modified only by the 82596. Defined bits are:

Bit 15 CX — The CU finished executing a command with its I (interrupt) bit set.

Bit 14 FR — The RU finished receiving a frame.

Bit 13 CNA — The Command Unit left the Active state.

Bit 12 RNR — The Receive Unit left the Ready state.

Bits 8–10 CUS — (3 bits) This field contains the status of the command unit. Valid values are:

0 — Idle

1 — Suspended

2 — Active

3–7 — Not used

Bits 4–7 RUS — This field contains the status of the receive unit. Valid values are:

0h (0000) — Idle

1h (0001) — Suspended

2h (0010) — No Resources. This bit indicates both no resources due to lack of RFDs in the RDL and no resources due to lack of RBDs in the FBL.

4h (0100) — Ready

8h (1000) — No more RBDs (not in the 82586 mode)

Ah (1010) — No resources due to no more RBDs (not in the 82586 mode).

No other combinations are allowed

Bit 3 T — Bus Throttle timers loaded (not in 82586 mode).

SCB OFFSET ADDRESSES

CBL Offset (Address)

In 82586 and 32-bit Segmented modes this 16-bit quantity indicates the offset portion of the address for the first Command Block on the CBL. In Linear mode it is a 32-bit linear address for the first Command Block on the CBL. It is accessed only if CUC equals Start.

RFA Offset (Address)

In 82586 and 32-bit Segmented modes this 16-bit quantity indicates the offset portion of the address for the Receive Frame Area. In Linear mode it is a 32-bit linear address for the Receive Frame Area. It is accessed only if RUC equals Start.

SCB STATISTICAL COUNTERS

Statistical Counter Operation

- The CPU is responsible for clearing all error counters before initializing the 82596. The 82596 updates these counters by reading them, adding 1, and then writing them back to the SCB.
- The counters are wraparound counters. After reaching FFFFFFFFh the counters wrap around to zero.
- The 82596 updates the required counters for each frame. It is possible for more than one counter to be updated; multiple errors will result in all affected counters being updated.
- The 82596 executes the read-counter/increment/write-counter operation without relinquishing the bus (locked operation). This is to ensure that no logical contention exists between the 82596 and the CPU due to both attempting to write to the counters simultaneously. In the dual-port memory configuration the CPU should not execute any write operation to a counter if $\overline{LOCK}$ is asserted.
- The counters are 32-bits wide and their behavior is fully compatible with the IEEE 802.3 standard. The 82596 supports all relevant statistics (mandatory, optional, and desired) through the status of the transmit and receive header and directly through SCB statistics.

CRCERRS

This 32-bit quantity contains the number of aligned frames discarded because of a CRC error. This counter is updated, if needed, regardless of the RU state.

ALNERRS

This 32-bit quantity contains the number of frames that both are misaligned (i.e., where $\overline{CRS}$ deasserts on a nonoctet boundary) and contain a CRC error. The counter is updated, if needed, regardless of the RU state.

SHRTFRM

This 32-bit quantity contains the number of received frames shorter than the minimum frame length.

The last three counters change function in monitor mode.

RSCERRS

This 32-bit quantity contains the number of good frames discarded because there were no resources to contain them. Frames intended for a host whose RU is in the No Receive Resources state, fall into this category. This counter is updated only if the RU is in the No Resources state. When in Monitor mode this counter counts the total number of frames—good and bad.

OVRNERRS

This 32-bit quantity contains the number of frames known to be lost because the local system bus was not available. If the traffic problem lasts longer than the duration of one frame, the frames that follow the first are lost without an indicator, and they are not counted. This counter is updated, if needed, regardless of the RU state.

RCVCDT

This 32-bit quantity contains the number of collisions detected during frame reception. In Monitor mode this counter counts the total number of good frames.

ACTION COMMANDS AND OPERATING MODES

This section lists all the Action Commands of the Command Unit Command Block List (CBL). Each command contains the Command field, the Status and Control fields, the link to the next Action Command, and any command-specific parameters. There are three basic types of action commands: 82596 Configuration and Setup, Transmission, and Diagnostics. The following is a list of the actual commands.

- NOP
- Individual Address Setup
- Configure
- MC Setup
- Transmit
- TDR
- Dump
- Diagnose

The 82596 has three addressing modes. In the 82586 mode all the Action Commands look exactly like those of the 82586.

- **82586 Mode.** The 82596 software and memory structure is compatible with the 82586.
- **32-Bit Segmented Mode.** The 82596 can access the entire system memory and use the two new memory structures—Simplified and Flexible—while still using the segmented approach. This does not require any significant changes to existing software.
- **Linear Mode.** The 82596 operates in a flat, linear, 4 gigabyte memory space without segmentation. It can also use the two new memory structures.

In the 32-bit Segmented mode there are some differences between the 82596 and 82586 action commands, mainly in programming and activating new 82596 features. Those bits marked "don't care" in the compatible mode are not checked; however, we strongly recommend that those bits all be zeroes; this will allow future enchancements and extensions.

In the Linear mode all of the address offsets become 32-bit address pointers. All new 82596 features are accessible in this mode, and all bits previously marked "don't care" must be zeroes.

The Action Commands, and all other 82596 memory structures, must begin on even byte boundaries, i.e., they must be word aligned.

NOP

This command results in no action by the 82596 except for those performed in the normal command processing. It is used to manipulate the CBL manipulation. The format of the NOP command is shown in Figure 21.

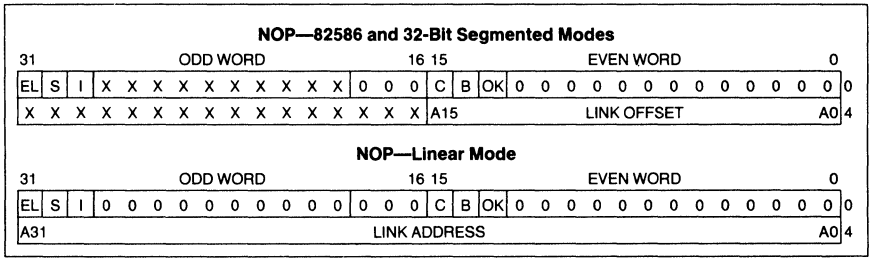


Figure 21

where:

LINK POINTER — In the 82586 or 32-bit Segmented modes this is a 16-bit offset to the next Command Block. In the Linear mode this is the 32-bit address of the next Command Block.

EL — If set, this bit indicates that this command block is the last on the CBL.

S — If set to one, suspend the CU upon completion of this CB.

I — If set to one, the 82596 will generate an interrupt after execution of the command is complete. If I is not set to one, the CX bit will not be set.

CMD (bits 16–18) — The NOP command. Value: 0h.

Bits 19–28 — Reserved (zero in the 32-bit Segmented and Linear modes).

C — This bit indicates the execution status of the command. The CPU initially resets it to zero when the Command Block is placed on the CBL. Following a command Completion, the 82596 will set it to one.

B — This bit indicates that the 82596 is currently executing the NOP command. It is initially reset to zero by the CPU. The 82596 sets it to one when execution begins and to zero when execution is completed. This bit is also set when the 82596 prefetches the command.

NOTE:

The C and B bits are modified in one operation.

OK — Indicates that the command was executed without error. If set to one no error occurred (command executed OK). If zero an error occured.

Individual Address Setup

This command is used to load the 82596 with the Individual Address. This address is used by the 82596 for inserting the Source Address during transmission and recognizing the Destination Address during reception. After RESET, and prior to Individual Address Setup Command execution, the 82596 assumes the Broadcast Address is the Individual Address in all aspects, i.e.:

- This will be the Individual Address Match reference.
- This will be the Source Address of a transmitted frame (for AL-LOC = 0 mode only).

The format of the Individual Address Setup command is shown in Figure 22.

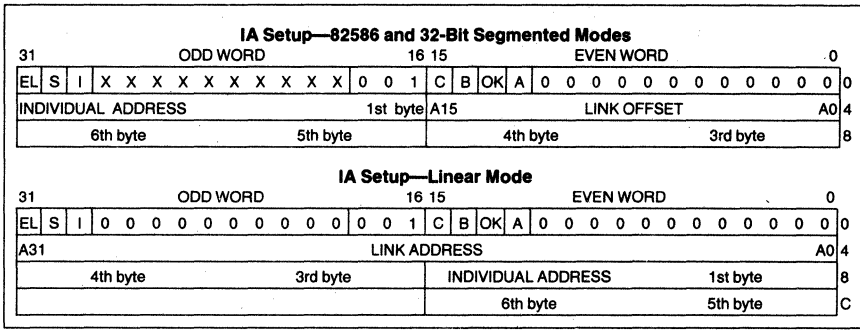


Figure 22

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)

A — Indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.

Bits 19-28 — Reserved (zero in the 32-bit Segmented and Linear modes).

CMD (bits 16-18) — The Address Setup command. Value: 1h.

INDIVIDUAL ADDRESS — The individual address of the node, 0 to 6 bytes long.

The least significant bit of the Individual Address must be zero for Ethernet (see the Command Structure). However, no enforcement of 0 is provided by the 82596. Thus, an Individual Address with 1 as its least significant bit is a valid Individual Address in all aspects.

The default address length is 6 bytes long, as in 802.3. If a different length is used the IA Setup command should be executed after the Configure command.

Configure

The Configure command loads the 82596 with its operating parameters. It allows changing some of the parameters by specifying a byte count less than the maximum number of configuration bytes (12 in the 82586 mode, 16 in the 32-Bit Segmented and Linear modes). The 82596 configuration depends on its mode of operation.

- In the 82586 mode the maximum number of configuration bytes is 12. Any number larger than 12 will be reduced to 12 and any number less than 4 will be increased to 4.
- The additional features of the serial side are disabled in the 82586 mode.
- In both the 32-Bit Segmented and Linear modes there are four additional configuration bytes, which hold parameters for additional 82596 features. If these parameters are not accessed, the 82596 will follow their default values.
- For more detailed information refer to the 32-Bit LAN Components User's Manual.

The format of the Configure command is shown in Figure 23, 24 and 25.

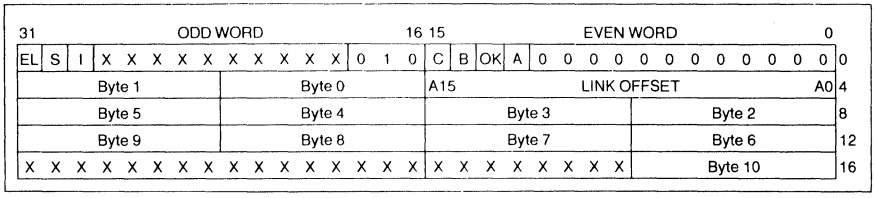


Figure 23. CONFIGURE—82586 Mode

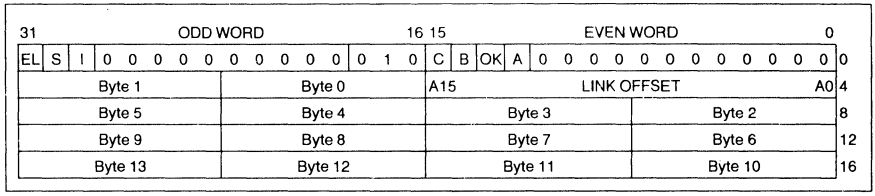


Figure 24. CONFIGURE—32-Bit Segmented Mode

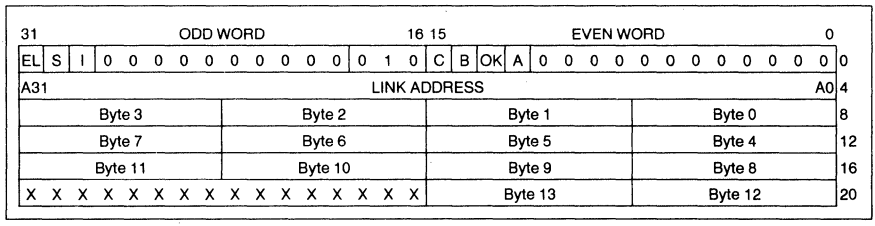


Figure 25. CONFIGURE—Linear Mode

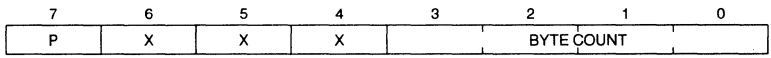
LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)

A — Indicates that the command was abnormally terminated due to a CU Abort control command. If 1, then the command was aborted and if necessary it should be repeated. If this bit is 0, the command was not aborted.

Bits 19-28 — Reserved (zero in the 32-Bit Segmented and Linear Modes)

CMD (bits 16-18) — The CONFIGURE command. Value: 2h.

The interpretation of the fields follows:

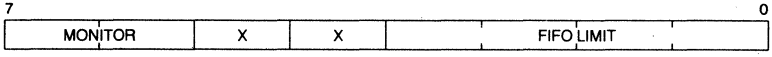


BYTE 0

BYTE CNT (Bits 0-3) — Byte Count. Number of bytes, including this one, that hold parameters to be configured.

PREFETCHED (Bit 7) — Enable the 82596 to write the prefetched bit in all prefetch RBDs.

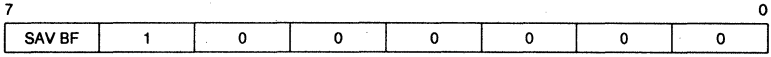
NOTE:
The P bit is valid only in the new memory structure modes. In 82586 mode this bit is disabled (i.e., no prefetched mark).



BYTE 1

FIFO Limit (Bits 0–3)	FIFO limit.
MONITOR# (Bits 6–7)	Receive monitor options. If the Byte Count of the configure command is less than 12 bytes then these Monitor bits are ignored.

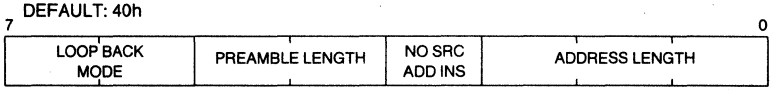
DEFAULT: C8h



BYTE 2

SAV BF (Bit 7)	0—Received bad frames are not saved in the memory. 1—Received bad frames are saved in the memory.

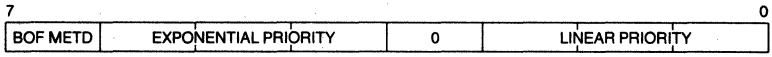
DEFAULT: 40h



BYTE 3

ADR LEN (Bits 0–2)	Address length (any kind).
NO SCR ADD INS (Bit 3)	No Source Address Insertion. In the 82586 this bit is called AL LOC.
PREAM LEN (Bits 4–5)	Preamble length.
LP BCK MODE (Bits 6–7)	Loopback mode.

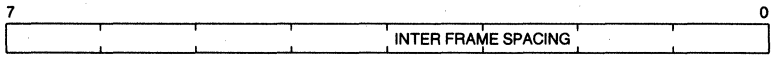
DEFAULT: 26h



BYTE 4

LIN PRIO (Bits 0–2)	Linear Priority.
EXP PRIO (Bits 4–6)	Exponential Priority.
BOF METD (Bit 7)	Exponential Backoff method.

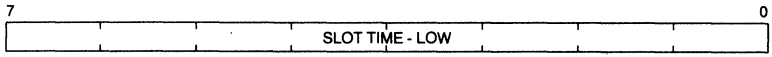
DEFAULT: 00h



BYTE 5

INTERFRAME SPACING	Interframe spacing.

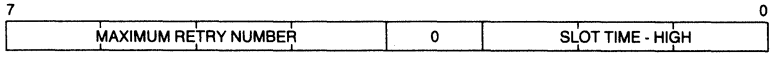
DEFAULT: 60h



BYTE 6

SLOT TIME (L) — Slot time, low byte.

DEFAULT: 00h

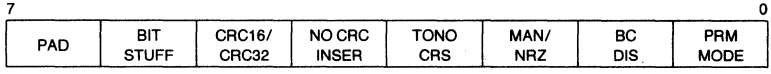


BYTE 7

SLOT TIME (H) (Bits 0–2) — Slot time, high part.

RETRY NUM (Bits 4–7) — Number of transmission retries on collision.

DEFAULT: F2h



BYTE 8

PRM (Bit 0) — Promiscuous mode.

BC DIS (Bit 1) — Broadcast disable.

MANCH/NRZ (Bit 2) — Manchester or NRZ encoding. See specific timing requirements for TXC in Manchester mode.

TONO CRS (Bit 3) — Transmit on no CRS.

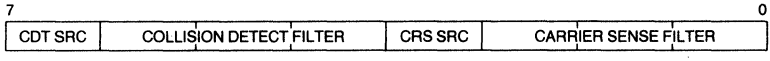
NOCRC INS (Bit 4) — No CRC insertion.

CRC-16/CRC-32 (Bit 5) — CRC type.

BIT STF (Bit 6) — Bit stuffing.

PAD (Bit 7) — Padding.

DEFAULT: 00h



BYTE 9

CRSF (Bits 0–2) — Carrier Sense filter (length).

CRS SRC (Bit 3) — Carrier Sense source.

CDTF (Bits 4–6) — Collision Detect filter (length).

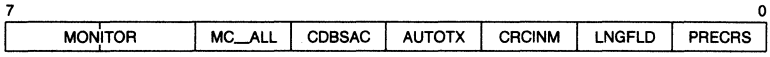
CDT SRC (Bit 7) — Collision Detect source.

DEFAULT: 00h

BYTE 10

MIN FRAME LEN — Minimum frame length.

DEFAULT: 40h



BYTE 11

PRECRS (Bit 0) — Preamble until Carrier Sense

LNGFLD (Bit 1) — Length field. Enables padding at the End-of-Carrier framing (802.3).

CRCINM (Bit 2) — Rx CRC appended to the frame in memory.

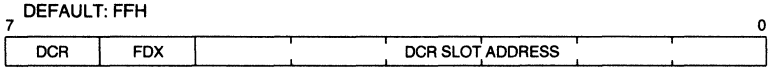
AUTOTX (Bit 3) — Auto retransmit when a collision occurs during the preamble.

CDBSAC (Bit 4) — Collision Detect by source address recognition.

MC_ALL (Bit 5) — Enable to receive all MC frames.

MONITOR (Bits 6–7) — Receive monitor options.

DEFAULT: FFH



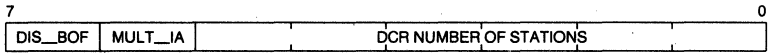
BYTE 12

DCR SLOT ADDRESS (Bits 0–5) — Station index in DCR mode.

FDX (Bit 6) — Enables Full Duplex operation.

DCR (Bit 7) — Enables Deterministic collision resolution.

DEFAULT: 00h



BYTE 13

DCR NUMBER OF STATIONS (Bits 0–5) — Number of stations in DCR mode.

MULT_IA (Bit 6) — Multiple individual address.

DIS_BOF (Bit 7) — Disable the backoff algorithm.

DEFAULT: 3Fh

A reset (hardware or software) configures the 82596 according to the following defaults.

Table 4. Configuration Defaults

	Parameter	Default Value	Units/Meaning
	ADDRESS LENGTH	**6	Bytes
	A/L FIELD LOCATION	0	Located in FD
*	AUTO RETRANSMIT	1	Auto Retransmit Enable
	BITSTUFFING/EOC	0	EOC
	BROADCAST DISABLE	0	Broadcast Reception Enabled
*	CDBSAC	1	Disabled
	CDT FILTER	0	Bit Times
	CDT SRC	0	External Collision Detection
*	CRC IN MEMORY	1	CRC Not Transferred to Memory
	CRC-16/CRC-32	**0	CRC-32
	CRS FILTER	0	0 Bit Times
	CRS SRC	0	External CRS
*	DCR	0	Disable DCR Protocol
*	DCR Slot Number	0	DCR Disabled
*	DCR Number of Stations	63	Stations
*	DISBOF	0	Backoff Enabled
	EXT LOOPBACK	0	Disabled
	EXPONENTIAL PRIORITY	**0	802.3 Algorithm
	EXPONENTIAL BACKOFF METHOD	**0	802.3 Algorithm
*	FULL DUPLEX (FDX)	0	CSMA/CD Protocol (No FDX)
	FIFO THRESHOLD	8	TX: 32 Bytes, RX: 64 Bytes
	INT LOOPBACK	0	Disabled
	INTERFRAME SPACING	**96	Bit Times
	LINEAR PRIORITY	**0	802.3 Algorithm
*	LENGTH FIELD	1	Padding Disabled
	MIN FRAME LENGTH	**64	Bytes
*	MC ALL	1	Disabled
*	MONITOR	11	Disabled
	MANCHESTER/NRZ	0	NRZ
*	MULTI IA	0	Disabled
	NUMBER OF RETRIES	**15	Maximum Number of Retries
	NO CRC INSERTION	0	CRC Appended to Frame
	PREFETCH BIT IN RBD	0	Disabled (Valid Only in New Modes)
	PREAMBLE LENGTH	**7	Bytes
*	Preamble Until CRS	1	Disabled
	PROMISCUOUS MODE	0	Address Filter On
	PADDING	0	No Padding
	SLOT TIME	**512	Bit Times
	SAVE BAD FRAME	0	Discards Bad Frames
	TRANSMIT ON NO CRS	0	Disabled

NOTES
1. This configuration setup is compatible with the IEEE 802.3 specification.
2. The Asterisk "*" signifies a new configuration parameter not available in the 82586.
3. The default value of the Auto retransmit configuration parameter is enabled(1).
4. Double Asterisk "**" signifies IEEE 802.3 requirements.

Multicast-Setup

This command is used to load the 82596 with the Multicast-IDs that should be accepted. As noted previously, the filtering done on the Multicast-IDs is not perfect and some unwanted frames may be accepted. This command resets the current filter and reloads it with the specified Multicast-IDs. The format of the Multicast-addresses setup command is:

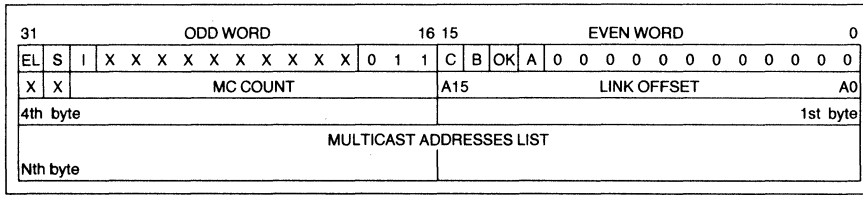


Figure 26. MC Setup—82586 and 32-Bit Segmented Modes

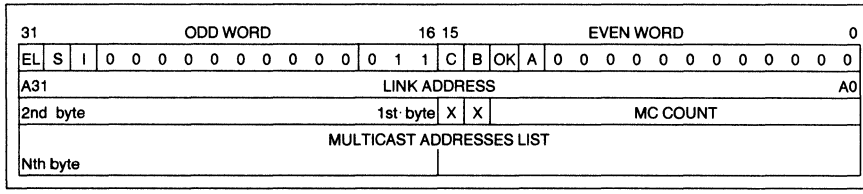


Figure 27. MC Setup—Linear Mode

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)

A — Indicates that the command was abnormally terminated due to a CU Abort control command. If one, then the command was aborted and if necessary it should be repeated. If this bit is zero, the command was not aborted.

Bits 19–28 — Reserved (0 in both the 32-Bit Segmented and Linear Modes).

CMD (bits 16–18) — The MC SETUP command value: 3h.

MC-CNT — This 14-bit field indicates the number of bytes in the MC LIST field. The MC CNT must be a multiple of the ADDR LEN; otherwise, the 82596 reduces the MC CNT to the nearest ADDR LEN multiple. MC CNT = 0 implies resetting the Hash table which is equivalent to disabling the Multicast filtering mechanism.

MC LIST — A list of Multicast Addresses to be accepted by the 82596. The least significant bit of each MC address must be 1.

NOTE:

The list is sequential; i.e., the most significant byte of an address is immediately followed by the least significant byte of the next address.

— When the 82596 is configured to recognize multiple Individual Address (Multi-IA), the MC-Setup command is also used to set up the Hash table for the individual address.

The least significant bit in the first byte of each IA address must be 0.

Transmit

This command is used to transmit a frame of user data onto the serial link. The format of a Transmit command is as follows.

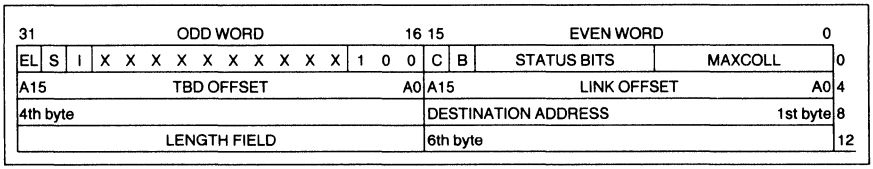


Figure 28. TRANSMIT—82586 Mode

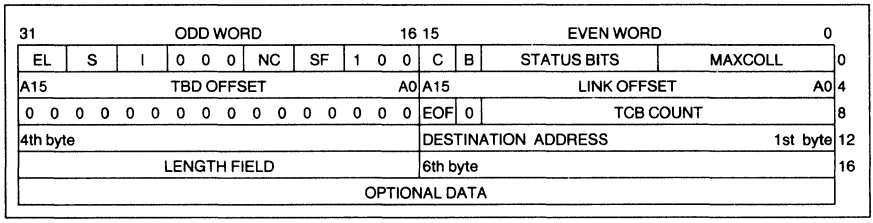


Figure 29. TRANSMIT—32-Bit Segmented Mode

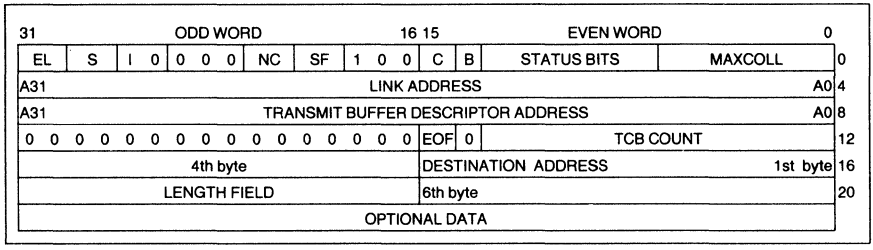
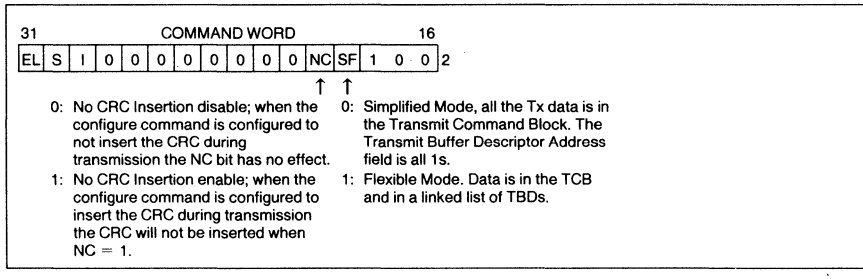


Figure 30. TRANSMIT—Linear Mode



where:

EL, B, C, I, S	— As per standard Command Block (see the NOP command for details).
OK (Bit 13)	— Error free completion.
A (Bit 12)	— Indicates that the command was abnormally terminated due to CU Abort control command. If 1, then the command was aborted, and if necessary it should be repeated. If this bit is 0, the command was not aborted.
Bits 19–28	— Reserved (0 in the 32-bit Segmented and Linear modes).
CMD (Bits 16–18)	— The transmit command: 4h.
Status Bit 11	— Late collision. A late collision (a collision after the slot time is elapsed) is detected.
Status Bit 10	— No Carrier Sense signal during transmission. Carrier Sense signal is monitored from the end of Preamble transmission until the end of the Frame Check Sequence for TONOCRS = 1 (Transmit On No Carrier Sense mode) it indicates that transmission has been executed despite a lack of CRS. For TONOCRS = 0 (Ethernet mode), this bit also indicates unsuccessful transmission (transmission stopped when lack of Carrier Sense has been detected).
Status Bit 9	— Transmission unsuccessful (stopped) due to Loss of $\overline{\text{CTS}}$.
Status Bit 8	— Transmission unsuccessful (stopped) due to DMA Underrun; i.e., the system did not supply data for transmission.
Status Bit 7	— Transmission Deferred, i.e., transmission was not immediate due to previous link activity.
Status Bit 6	— Heartbeat Indicator, Indicates that after a previously performed transmission, and before the most recently performed transmission, (Interframe Spacing) the CDT signal was monitored as active. This indicates that the Ethernet Transceiver Collision Detect logic is performing properly. The Heartbeat is monitored during the Interframe Spacing period.
Status Bit 5	— Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.
MAX-COL (Bits 3–0)	— The number of Collisions experienced during this frame. Max Col = 0 plus S5 = 1 indicates 16 collisions.
LINK OFFSET	— As per standard Command Block (see the NOP Command for details)
TBD POINTER	— In the 82586 and 32-bit Segmented modes this is the offset of the first Tx Buffer Descriptor containing the data to be transmitted. In the Linear mode this is the 32-bit address of the first Tx Buffer Descriptor on the list. If the TBD POINTER is all 1s it indicates that no TBD is used.
DEST ADDRESS	— Contains the Destination Address of the frame. The least significant bit (MC) indicates the address type. MC = 0: Individual Address. MC = 1: Multicast or Broadcast Address. If the Destination Address bits are all 1s this is a Broadcast Address.
LENGTH FIELD	— The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is transmitted; i.e., most significant byte first, least significant byte second.
TCB COUNT	— This 14-bit counter indicates the number of bytes that will be transmitted from the Transmit Command Block, starting from the third byte after the TCB COUNT field (address $n+12$ in the 32-bit Segmented mode, $N+16$ in the Linear mode). The TCB COUNT field can be any number of bytes (including an odd byte), this allows the user to transmit a frame with a header having an odd number of bytes. The TCB COUNT field is not used in the 82586 mode.
EOF Bit	— Indicates that the whole frame is kept in the Transmit Command Block. In the Simplified memory model it must be always asserted.

The interpretation of what is transmitted depends on the No Source Address insertion configuration bit and the memory model being used.

NOTES

1. The Destination Address and the Length Field are sequential. The Length Field immediately follows the most significant byte of the Destination Address.
2. In case the 82596 is configured with No Source Address insertion bit equal to 0, the 82596 inserts its configured Source Address in the transmitted frame.
   - In the 82586 mode, or when the Simplified memory model is used, the Destination and Length fields of the transmitted frame are taken from the Transmit Command Block.
   - If the FLEXIBLE memory model is used, the Destination and Length fields of the transmitted frame can be found either in the TCB or TBD, depending on the TCB COUNT.
3. If the 82596 is configured with the Address/Length Field Location equal to 1, the 82596 does not insert its configured Source Address in the transmitted frame. The first (2 × Address Length) + 2 bytes of the transmitted frame are interpreted as Destination Address, Source Address, and Length fields respectively. The location of the first transmitted byte depends on the operational mode of the 82596:
   - In the 82586 mode, it is always the first byte of the first Tx Buffer.
   - In both the 32-bit Segmented and Linear modes it depends on the SF bit and TCB COUNT:
     - — In the Simplified memory mode the first transmitted byte is always the third byte after the TCB COUNT field.
     - — In the Flexible mode, if the TCB COUNT is greater than 0 then it is the third byte after the TCB COUNT field. If TCB COUNT equals 0 then it is first byte of the first Tx Buffer.
   - Transmit frames shorter than six bytes are invalid. The transmission will be aborted (only in 82586 mode) because of a DMA Underrun.
4. Frames which are aborted during transmission are jammed. Such an interruption of transmission can be caused by any reason indicated by any of the status bits 8, 9, 10 and 12.

Jamming Rules

1. Jamming will not start before completion of preamble transmission.
2. Collisions detected during transmission of the last 11 bits will not result in jamming.

The format of a Transmit Buffer Descriptor is:

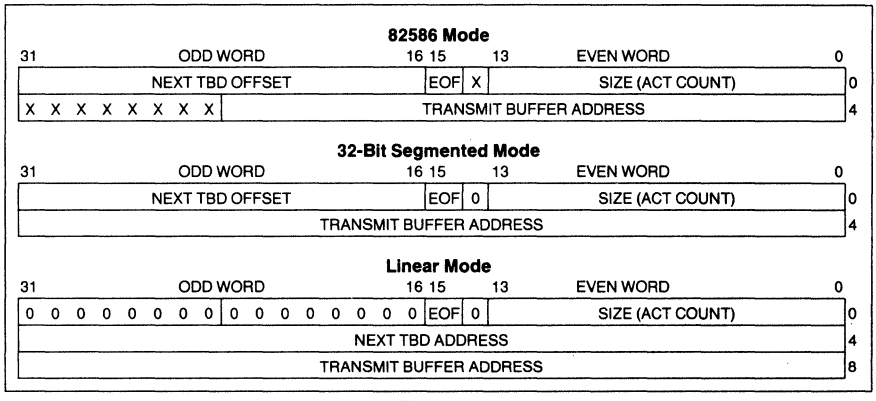


Figure 31

where:

EOF — This bit indicates that this TBD is the last one associated with the frame being transmitted. It is set by the CPU before transmit.

SIZE (ACT COUNT) — This 14-bit quantity specifies the number of bytes that hold information for the current buffer. It is set by the CPU before transmission.

NEXT TBD ADDRESS — In the 82586 and 32-bit Segmented modes, it is the offset of the next TBD on the list. In the Linear mode this is the 32-bit address of the next TBD on the list. It is meaningless if EOF = 1.

BUFFER ADDRESS — The starting address of the memory area that contains the data to be sent. In the 82586 mode, this is a 24-bit address (A31–A24 are considered to be zero). In the 32-bit Segmented and Linear modes this is a 32-bit address.

TDR

This operation activates Time Domain Reflectomet, which is a mechanism to detect open or short circuits on the link and their distance from the diagnosing station. The TDR command has no parameters. The TDR transmit sequence was changed, compared to the 82586, to form a regular transmission. The TDR bit stream is as follows.

— Preamble

— Source address

— Another Source address (the TDR frame is transmitted back to the sending station, so DEST ADR = SRC ADR).

— Data field containing 7Eh patterns.

— Jam Pattern, which is the inverse CRC of the transmitted frame.

Maximum length of the TDR frame is 2048 bits. If the 82596 senses collision while transmitting the TDR frame it transmits the jam pattern and stops the transmission. The 82596 then triggers an internal timer (STC); the timer is reset at the beginning of transmission and reset if CRS is returned. The timer measures the time elapsed from the start of transmission until an echo is returned. The echo is indicated by Collision Detect going active or a drop in the Carrier Sense signal. The following table lists the possible cases that the 82596 is able to analyze.

Conditions of TDR as Interpreted by the 82596

Transceiver Type / Condition	Ethernet	Non Ethernet
Carrier Sense was inactive for 2048-bit-time periods	Short or Open on the Transceiver Cable	NA
Carrier Sense signal dropped	Short on the Ethernet cable	NA
Collision Detect went active	Open on the Ethernet cable	Open on the Serial Link
The Carrier Sense Signal did not drop or the Collision Detect did not go active within 2048-bit time period	No Problem	No Problem

An Ethernet transceiver is defined as one that returns transmitted data on the receive pair and activates the Carrier Sense Signal while transmitting. A Non-Ethernet Transceiver is defined as one that does not do so.

The format of the Time Domain Reflectometer command is:

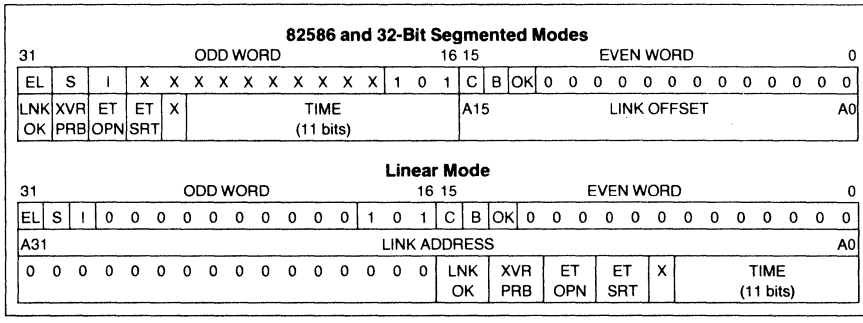


Figure 32. TDR

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details).

A — Indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.

Bits 19–28 — Reserved (0 in the 32-bit Segmented and Linear Modes).

CMD (Bits 16–18) — The TDR command. Value: 5h.

TIME — An 11-bit field that specifies the number of TxC cycles that elapsed before an echo was observed. No echo is indicated by a reception consisting of "1s" only. Because the network contains various elements such as transceiver links, transceivers, Ethernet, repeaters etc., the TIME is not exactly proportional to the problems distance.

LNK OK (Bit 15) — No link problem identified. TIME = 7FFh.

XCVR PRB (Bit 14) — Indicates a Transceiver problem. Carrier Sense was inactive for 2048-bit time period. LNK OK = 0. TIME = 7FFh.

ET OPN (Bit 13) — The transmission line is not properly terminated. Collision Detect went active and LNK OK = 0.

ET SRT (Bit 12) — There is a short circuit on the transmission line. Carrier Sense Signal dropped and LNK OK = 0.

DUMP

This command causes the contents of various 82596 registers to be placed in a memory area specified by the user. It is supplied as a 82596 self-diagnostic tool, and to provide registers of interest to the user. The format of the DUMP command is:

82586 and 32-Bit Segmented Modes

31	ODD WORD	16 15	EVEN WORD	0
EL S I X X X X X X X X X X	1 1 0	C B OK	0 0 0 0 0 0 0 0 0 0 0 0	
A15	BUFFER OFFSET	A0	A15 LINK OFFSET	A0

Linear Mode

31	ODD WORD	16 15	EVEN WORD	0
EL S I X X X X X X X X X X	1 1 0	C B OK	0 0 0 0 0 0 0 0 0 0 0 0	
A31	LINK ADDRESS			A0
A31	BUFFER ADDRESS			A0

Figure 33. Dump

where:

LINK ADDRESS, EL, B, C, I, S	— As per standard Command Block (see the NOP command for details).
OK	— Indicates error free completion.
Bits 19-28	— Reserved (0 in the 32-bit Segmented and Linear Modes).
CMD (Bits 16-18)	— The Dump command. Value: 6h.
BUFFER POINTER	— In the 82586 and 32-bit Segmented modes this is the 16-bit-offset portion of the dump area address. In the Linear mode this is the 32-bit linear address of the dump area.

Dump Area Information Format

- The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586. The compatible data will be marked with an asterisk.
- In 82586 mode the dump area is 170 bytes.
- The DUMP area format of the 32-bit Segmented and Linear modes is described in Figure 35.
- The size of the dump area of the 32-bit Segmented and Linear modes is 304 bytes.
- When the Dump is executed by the Port command an extra word will be appended to the Dump Area. The extra word is a copy of the Dump Area status word (containing the C, B, and OK Bits). The C and OK Bits are set when the 82596 has completed the Port Dump command.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
DMA CONTROL REGISTER	00
CONFIGURE BYTES* 3, 2	02
CONFIGURE BYTES* 5, 4	04
CONFIGURE BYTES* 7, 6	06
CONFIGURE BYTES* 9, 8	08
CONFIGURE BYTES* 10	0A
I.A. BYTES 1, 0*	0C
I.A. BYTES 3, 2*	0E
I.A. BYTES 5, 4*	10
LAST T.X. STATUS*	12
T.X. CRC BYTES 1, 0*	14
T.X. CRC BYTES 3, 2*	16
R.X. CRC BYTES 1, 0*	18
R.X. CRC BYTES 3, 2*	1A
R.X. TEMP MEMORY 1, 0*	1C
R.X. TEMP MEMORY 3, 2*	1E
R.X. TEMP MEMORY 5, 4*	20
LAST RECEIVED STATUS*	22
HASH REGISTER BYTES 1, 0*	24
HASH REGISTER BYTES 3, 2*	26
HASH REGISTER BYTES 5, 4*	28
HASH REGISTER BYTES 7, 6*	2A
SLOT TIME COUNTER*	2C
WAIT TIME COUNTER*	2E
MICRO MACHINE**	30
REGISTER FILE	.
	.
60 BYTES	6A
MICRO MACHINE LFSR**	6C
MICRO MACHINE**	6E
FLAG ARRAY	.
	.
14 BYTES	7A
QUEUE MEMORY**	7C
CU PORT	.
8 BYTES	82
MICRO MACHINE ALU**	84
RESERVED**	86
M.M. TEMP A ROTATE R**	88
M.M. TEMP A**	8A
T.X. DMA BYTE COUNT**	8C
M.M. INPUT PORT ADDRESS**	8E
T.X. DMA ADDRESS	90
M.M. OUTPUT PORT**	92
R.X. DMA BYTE COUNT**	94
R.U. DMA ADDRESS**	96
M.M. OUTPUT PORT ADDRESS REGISTER**	98
RESERVED**	9A
BUS THROTTLE TIMERS	9C
DIU CONTROL REGISTER**	9E
RESERVED**	A0
DMA CONTROL REGISTER**	A2
BIU CONTROL REGISTER**	A4
M.M. DISPATCHER REG.**	A6
M.M. STATUS REGISTER**	A8

*The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586.

**These bytes are not user defined, results may vary from Dump command to Dump command.

Figure 34. Dump Area Format—82586 Mode

31	0	
CONFIGURE BYTES 5, 4, 3, 2		00
CONFIGURE BYTES 9, 8, 7, 6		04
CONFIGURE BYTES 13, 12, 11, 10		08
I.A. BYTES 1, 0	X X X X X X X X	0C
I.A. BYTES 5, 2		10
TX CRC BYTES 0, 1	LAST T.X. STATUS	14
RX CRC BYTES 0, 1	TX CRC BYTES 3, 2	18
RX TEMP MEMORY 1, 0	RX CRC BYTES 3, 2	1C
R.X. TEMP MEMORY 5, 2		20
HASH REGISTERS 1, 0	LAST R.X. STATUS	24
HASH REGISTER BYTES 5, 2		28
SLOT TIME COUNTER	HASH REGISTERS 7, 6	2C
RECEIVE FRAME LENGTH	WAIT-TIME COUNTER	30
MICRO MACHINE** REGISTER FILE 128 BYTES		34 . . . B0
MICRO MACHINE LFSR**		B4
MICRO MACHINE** FLAG ARRAY 28 BYTES		B8 . . . D0
M.M. INPUT PORT** 16 BYTES		D4 . E0
MICRO MACHINE ALU**		E4
RESERVED**		E8
M.M. TEMP A ROTATE R.**		EC
M.M. TEMP A**		F0
T.X. DMA BYTE COUNT**		F4
M.M. INPUT PORT ADDRESS REGISTER**		F8
T.X. DMA ADDRESS**		FC
M.M. OUTPUT PORT REGISTER**		100
R.X. DMA BYTE COUNT**		104
M.M. OUTPUT PORT ADDRESS REGISTER**		108
R.X. DMA ADDRESS REGISTER**		10C
RESERVED**		110
BUS THROTTLE TIMERS		114
DIU CONTROL REGISTER**		118
RESERVED**		11C
DMA CONTROL REGISTER**		120
BIU CONTROL REGISTER**		124
M.M. DISPATCHER REG.**		128
M.M. STATUS REGISTER**		12C

*The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586.
**These bytes are not user defined, results may vary from Dump command to Dump command.

Figure 35. Dump Area Format—Linear and 32-Bit Segmented Mode

Diagnose

The Diagnose Command triggers an internal self-test procedure that checks internal 82596 hardware, which includes:

- Exponential Backoff Random Number Generator (Linear Feedback Shift Register).
- Exponential Backoff Timeout Counter.
- Slot Time Period Counter.
- Collision Number Counter.
- Exponential Backoff Shift Register.
- Exponential Backoff Mask Logic.
- Timer Trigger Logic.

This procedure checks the operation of the Backoff block, which resides in the serial side and is not easily controlled. The Diagnose command is performed in two phases.

The format of the 82596 Diagnose command is:

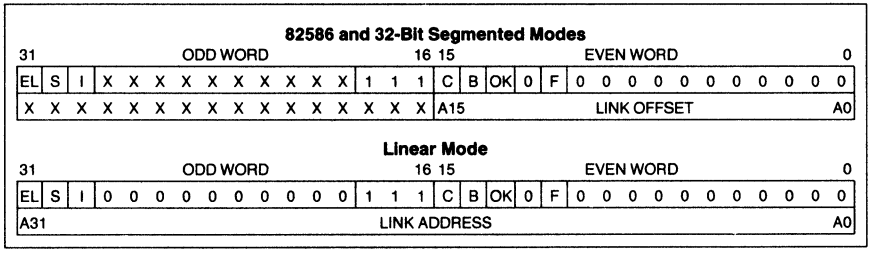


Figure 36. Diagnose

where:

LINK ADDRESS, EL, B, C, I, S	— As per standard Command Block (see the NOP command for details).
Bits 19–28	— Reserved (0 in the 32-bit Segmented and Linear Modes).
CMD (bits 16–18)	— The Diagnose command. Value: 7h.
OK (bit 13)	— Indicates error free completion.
F (bit 11)	— Indicates that the self-test procedure has failed.

RECEIVE FRAME DESCRIPTOR

Each received frame is described by one Receive Frame Descriptor (see Figure 37). Two new memory structures are available for the received frames. The structures are available only in the Linear and 32-bit Segmented modes.

Simplified Memory Structure

The first is the Simplified memory structure, the data section of the received frame is part of the RFD and is located immediately after the Length Field. Receive Buffer Descriptors are not used with the Simplified structure, it is primarily used to make programming easier. If the length of the data area described in the Size Field is smaller than the incoming frame, the following happens.

1. The received frame is truncated.
2. The No Resource error counter is updated.
3. If the 82596 is configured to Save Bad Frames the RFD is not reused; otherwise, the same RFD is used to hold the next received frame, and the only action taken regarding the truncated frame is to update the counter.
4. The 82596 continues to receive the next frame in the next RFD.

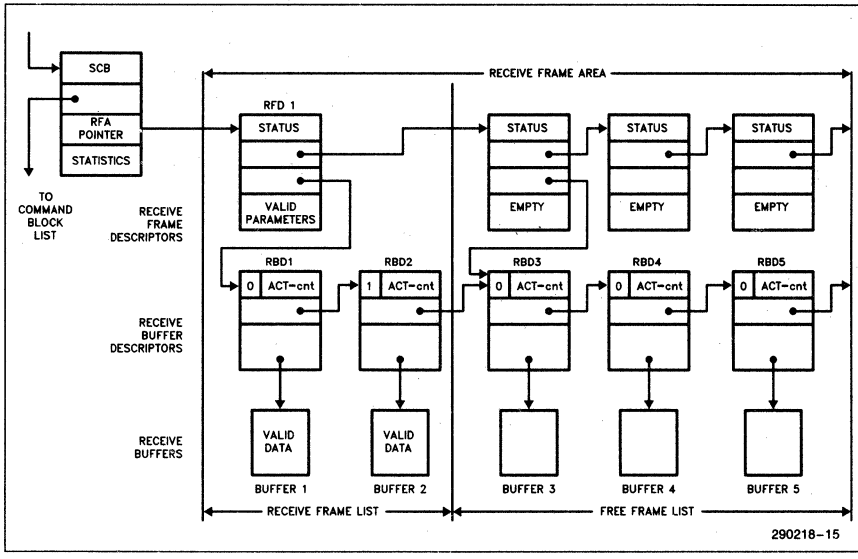


Figure 37. The Receive Frame Area

Note that this sequence is very useful for monitoring. If the 82596 is configured to Save Bad Frames, to receive in Promiscuous mode, and to use the Simplified memory structure, any programmed length of received data can be saved in memory.

The Simplified memory structure is shown in Figure 38.

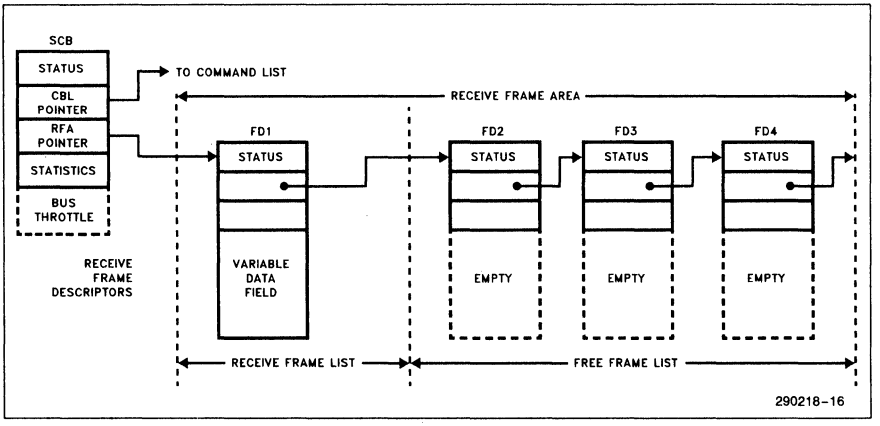


Figure 38. RFA Simplified Memory Structure

Flexible Memory Structure

The second structure is the Flexible memory structure, the data structure of the received frame is stored in both the RFD and in a linked list of Receive Buffers—Receive Buffer Descriptors. The received frame is placed in the RFD as configured in the Size field. Any remaining data is placed in a linked list of RBDs.

The Flexible memory structure is shown in Figure 39.

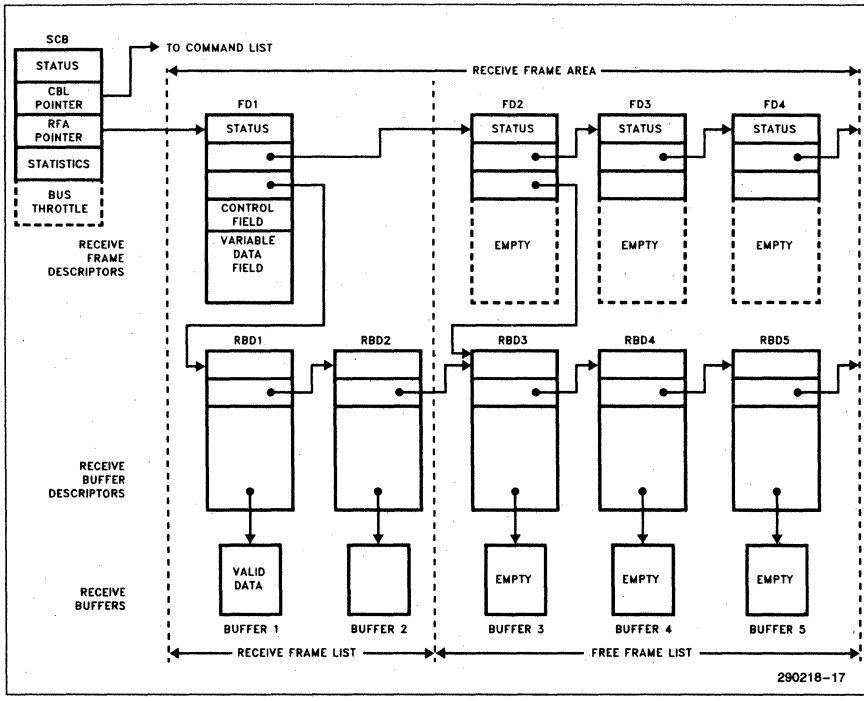


Figure 39. RFA Flexible Memory Structure

Buffers on the receive side can be different lengths. The 82596 will not place more bytes into a buffer than indicated in the associated RBD. The 82596 will fetch the next RBD before it is needed. The 82596 will attempt to receive frames as long as the FBL is not exhausted. If there are no more buffers, the 82596 Receive Unit will enter the No Resources state. Before starting the RU, the CPU must place the FBL pointer in the RBD pointer field of the first RFD. All remaining RBD pointer fields for subsequent RFDs should be "1s." If the Receive Frame Descriptor and the associated Receive Buffers are not reused (e.g., the frame is properly received or the 82596 is configured to Save Bad Frames), the 82596 writes the address of the next free RBD to the RBD pointer field of the next RFD.

Receive Buffer Descriptor (RBD)

The RBDs are used to store received data in a flexible set of linked buffers. The portion of the frame's data field that is outside the RFD is placed in a set of buffers chained by a sequence of RBDs. The RFD points to the first RBD, and the last RBD is flagged with an EOF bit set to 1. Each buffer in the linked list of buffers related to a particular frame can be any size up to 2^{14} bytes but must be word aligned (begin on an even numbered byte). This ensures optimum use of the memory resources while maintaining low overhead. All buffers in a frame are filled with the received data except for the last, in which the actual count can be smaller than the allocated buffer space.

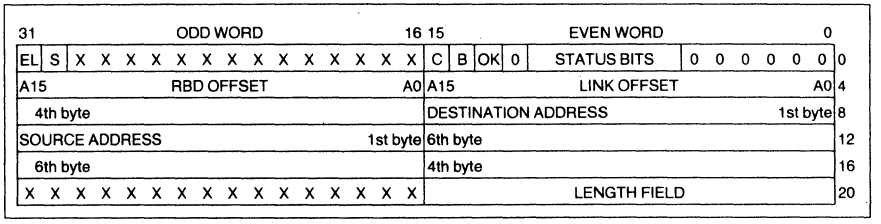


Figure 40. Receive Frame Descriptor—82586 Mode

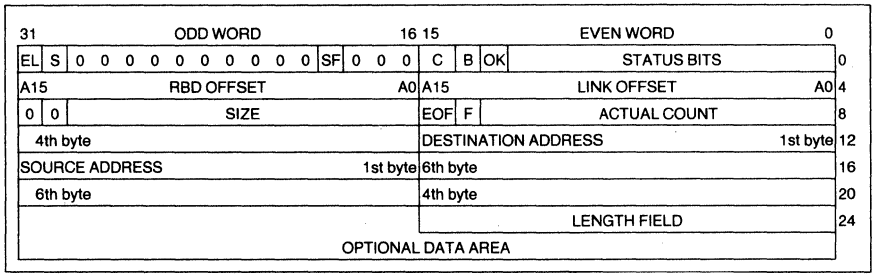


Figure 41. Receive Frame Descriptor—32-Bit Segmented Mode

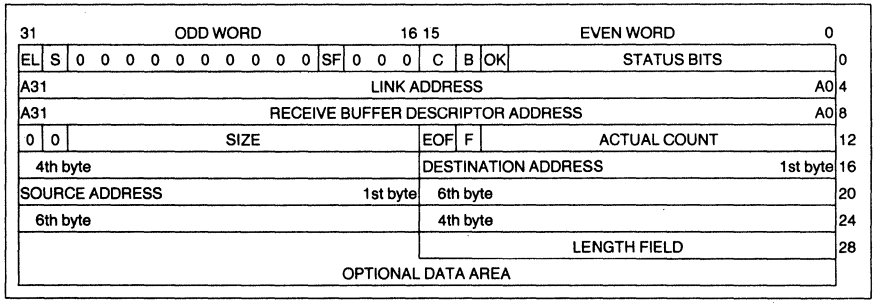


Figure 42. Receive Frame Descriptor—Linear Mode

where:

EL — When set, this bit indicates that this RFD is the last one on the RDL.

S — When set, this bit suspends the RU after receiving the frame.

SF — This bit selects between the Simplified or the Flexible mode.

0 — Simplified mode, all the RX data is in the RFD. RBD ADDRESS field is all "1s."

1 — Flexible mode. Data is in the RFD and in a linked list of Receive Buffer Descriptors.

C — This bit indicates the completion of frame reception. It is set by the 82596.

B — This bit indicates that the 82596 is currently receiving this frame, or that the 82596 is ready to receive the frame. It is initially set to 0 by the CPU. The 82596 sets it to 1 when reception set up begins, and to 0 upon completion. The C and B bits are set during the same operation.

OK (bit 13) — Frame received successfully, without errors. RFDs with bit 13 equal to 0 are possible only if the save bad frames, configuration option is selected. Otherwise all frames with errors will be discarded, although statistics will be collected on them.

STATUS — The results of the Receive operation. Defined bits are,

Bit 12: Length error if configured to check length

Bit 11: CRC error in an aligned frame

Bit 10: Alignment error (CRC error in misaligned frame)

Bit 9: Ran out of buffer space—no resources

Bit 8: DMA Overrun failure to acquire the system bus.

Bit 7: Frame too short.

Bit 6: No EOP flag (for Bit stuffing only)

Bit 5: When the SF bit equals zero, and the 82596 is configured to save bad frames, this bit signals that the receive frame was truncated. Otherwise it is zero.

Bits 2-4: Zeros

Bit 1: When it is zero, the destination address of the received frame matches the IA address. When it is a 1, the destination address of the received frame did not match the individual address. For example, a multicast address or broadcast address will set this bit to a 1.

Bit 0: Receive collision, a collision is detected during reception.

LINK ADDRESS — A 16-bit offset (32-bit address in the Linear mode) to the next Receive Frame Descriptor. The Link Address of the last frame can be used to form a cyclical list.

RBD POINTER — The offset (address in the Linear mode) of the first RBD containing the received frame data. An RBD pointer of all ones indicates no RBD.

EOF
F
SIZE
ACT COUNT — These fields are for the Simplified and Flexible memory models. They are exactly the same as the respective fields in the Receive Buffer Descriptor. See the next section for detailed explanation of their functions.

MC — Multicast bit.

DESTINATION ADDRESS — The contents of the destination address of the receive frame. The field is 0 to 6 bytes long.

SOURCE ADDRESS — The contents of the Source Address field of the received frame. It is 0 to 6 bytes long.

LENGTH FIELD — The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is received, i.e., most significant byte first, least significant byte second.

NOTES
1. The Destination address, Source address and Length fields are packed, i.e., one field immediately follows the next.
2. The affect of Address/Length Location (No Source Address Insertion) configuration parameter while receiving is as follows:

— 82586 Mode: The Destination address, Source address and Length field are not used, they are placed in the RX data buffers.

— 32-Bit Segmented and Linear Modes: when the Simplified memory model is used, the Destination address, Source address and Length fields reside in their respective fields in the RFD. When the Flexible memory strucrture is used the Destination address, Source address, and Length field locations depend on the SIZE field of the RFD. They can be placed in the RFD, in the RX data buffers, or partially in the RFD and the rest in the RX data buffers, depending on the SIZE field value.

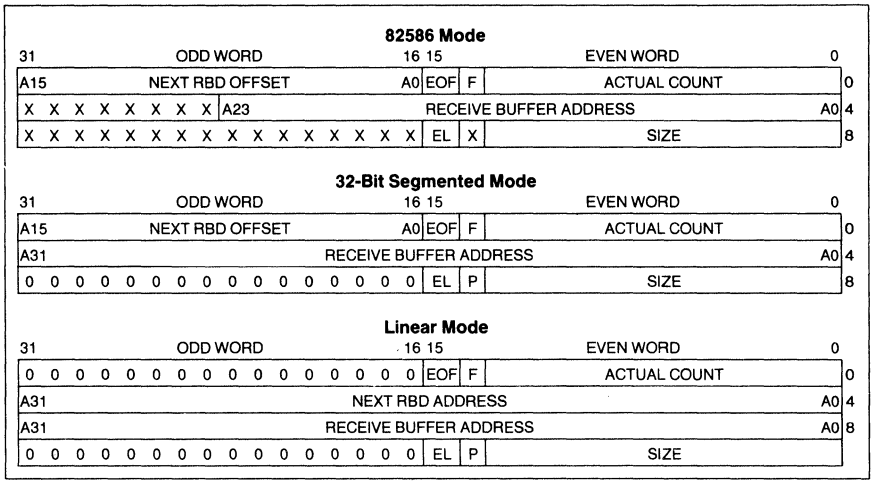


Figure 43. Receive Buffer Descriptor

where:

EOF — Indicates that this is the last buffer related to the frame. It is cleared by the CPU before starting the RU, and is written by the 82596 at the end of reception of the frame.

F — Indicates that this buffer has already been used. The Actual Count has no meaning unless the F bit equals one. This bit is cleared by the CPU before starting the RU, and is set by the 82596 after the associated buffer has been. This bit has the same meaning as the Complete bit in the RFD and CB.

ACT COUNT — This 14-bit quantity indicates the number of meaningful bytes in the buffer. It is cleared by the CPU before starting the RU, and is written by the 82596 after the associated buffer has already been used. In general, after the buffer is full, the Actual Count value equals the size field of the same buffer. For the last buffer of the frame, Actual Count can be less than the buffer size.

NEXT BD ADDRESS — The offset (absolute address in the Linear mode) of the next RBD on the list. It is meaningless if EL = 1.

BUFFER ADDRESS — The starting address of the memory area that contains the received data. In the 82586 mode, this is a 24-bit address (with pins A24–A31 = 0). In the 32-bit Segmented and Linear modes this is a 32-bit address.

EL — Indicates that the buffer associated with this RBD is last in the FBL.

P — This bit indicates that the 82596 has already prefetched the RBDs and any change in the RBD data will be ignored. This bit is valid only in the new 82596 memory modes, and if this feature has been enabled during configure command. The 82596 Prefetches the RBDs in locked cycles; after prefetching the RBD the 82596 performs a write cycle where the P bit is set to one and the rest of the data remains unchanged. The CPU is responsible for resetting it in all RBDs. The 82596 will not check this bit before setting it.

SIZE — This 14-bit quantity indicates the size, in bytes, of the associated buffer. This quantity must be an even number.

ELECTRICAL AND TIMING CHARACTERISTICS

DC Characteristics

T_C = 0°C–85°C, V_{CC} = 5V ±10% LE/$\overline{BE}$ have MOS levels (see V_{MIL}, V_{MIH}).
All other signals have TTL levels (see V_{IL}, V_{IH}, V_{OL}, V_{OH}).

Symbol	Parameter	Min	Max	Units	Notes
V_{IL}	Input Low Voltage (TTL)	−0.3	+0.8	V	
V_{IH}	Input High Voltage (TTL)	2.0	V_{CC} + 0.3	V	
V_{MIL}	Input Low Voltage (MOS)	−0.3	+0.8	V	
V_{MIH}	Input High Voltage (MOS)	3.7	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage (TTL)		0.45	V	I_{OL} = 4.0 mA(1)
V_{CIL}	$\overline{RXC}$, $\overline{TXC}$ Input Low Voltage	−0.5	0.6	V	
V_{CIH}	$\overline{RXC}$, $\overline{TXC}$ Input High Voltage	3.3	V_{CC}+0.5	V	
V_{OH}	Output High Voltage (TTL)	2.4		V	I_{OH} = 0.9mA–1 mA(1)
I_{LI}	Input Leakage Current		±15	µA	0 ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±15	µA	0.45 < V_{OUT} < V_{CC}
C_{IN}	Capacitance of Input Buffer		10	pF	FC = 1 MHz
C_{OUT}	Capacitance of Input/Output Buffer		12	pF	FC = 1 MHz
C_{CLK}	CLK Capacitance		20	pF	FC = 1 MHz
I_{CC}	Power Supply		200	mA	At 25 MHz
I_{CC}	Power Supply		300	mA	At 33 MHz

AC Characteristics

82596CA INPUT/OUTPUT SYSTEM TIMINGS

T_C = 0°C–85°C, V_{CC} = 5V ±10%. These timing assume the C_L on all outputs is 50 pF unless otherwise specified. C_L can be 20 pF to 120 pF however timings must be derated. All timing requirements are given in nanoseconds.

Symbol	Parameter	25 MHz		Notes
		Min	Max	
	Operating Frequency	12.5 MHz	25 MHz	1X CLK Input
T1	CLK Period	40	80	
T1a	CLK Period Stability		0.1%	Adjacent CLK Δ
T2	CLK High	14		2.0V
T3	CLK Low	14		0.8V
T4	CLK Rise Time		4	0.8V to 2.0V
T5	CLK Fall Time		4	2.0V to 0.8V
T6	$\overline{BEn}$, $\overline{LOCK}$, and A2–A31 Valid Delay	3	22	
T6a	$\overline{BLAST}$, $\overline{PCHK}$ Valid Delay	3	27	
T7	$\overline{BEn}$, $\overline{LOCK}$, $\overline{BLAST}$, A2–A31 Float Delay	3	30	
T8	W/$\overline{R}$ and $\overline{ADS}$ Valid Delay	3	22	
T9	W/$\overline{R}$ and $\overline{ADS}$ Float Delay	3	30	
T10	D0–D31, DPn Write Data Valid Delay	3	22	
T11	D0–D31, DPn Write Data Float Delay	3	30	
T12	HOLD Valid Delay	3	22	
T13	CA and BREQ Setup Time	7		1, 2
T14	CA and BREQ Hold Time	3		1, 2
T15	$\overline{BS16}$ Setup Time	8		2
T16	$\overline{BS16}$ Hold Time	3		2
T17	$\overline{BRDY}$, $\overline{RDY}$ Setup Time	8		2
T18	$\overline{BRDY}$, $\overline{RDY}$ Hold Time	3		2
T19	D0–D31, DPn READ Setup Time	5		2
T20	D0–D31, DPn READ Hold Time	3		2
T21	AHOLD and HLDA Setup Time	10		1, 2
T22	AHOLD Hold Time	3		1, 2
T22a	HLDA Hold Time	3		1, 2
T23	RESET Setup Time	10		1, 2
T24	RESET Hold Time	3		1, 2
T25	INT/$\overline{INT}$ Valid Delay	1	26	
T26	CA and BREQ, $\overline{PORT}$ Pulse Width	2 T1		1, 2, 3
T27	D0–D31 CPU $\overline{PORT}$ Access Setup Time	5		2
T28	D0–D31 CPU $\overline{PORT}$ Access Hold Time	3		2
T29	$\overline{PORT}$ Setup Time	7		2
T30	$\overline{PORT}$ Hold Time	3		2
T31	$\overline{BOFF}$ Setup Time	10		2
T32	$\overline{BOFF}$ Hold Time	3		2

AC Characteristics (Continued)

82596CA INPUT/OUTPUT SYSTEM TIMINGS

T_C = 0°C–85°C, V_{CC} = 5V ±5%. These timing assume the C_L on all outputs is 50 pF unless otherwise specified. C_L can be 20 pF to 120 pF, however timings must be derated. All timing requirements are given in nanoseconds.

Symbol	Parameter	33 MHz		Notes
		Min	Max	
	Operating Frequency	12.5 MHz	33 MHz	1X CLK Input
T1	CLK Period	30	80	
T1a	CLK Period Stability		0.1%	Adjacent CLK Δ
T2	CLK High	11		2.0V
T3	CLK Low	11		0.8V
T4	CLK Rise Time		3	0.8V to 2.0V
T5	CLK Fall Time		3	2.0V to 0.8V
T6	$\overline{BEn}$, $\overline{LOCK}$, and A2–A31 Valid Delay	3	19	
T6a	$\overline{BLAST}$, $\overline{PCHK}$ Valid Delay	3	22	
T7	$\overline{BEn}$, $\overline{LOCK}$, $\overline{BLAST}$, A2–A31 Float Delay	3	20	
T8	W/$\overline{R}$ and $\overline{ADS}$ Valid Delay	3	19	
T9	W/$\overline{R}$ and $\overline{ADS}$ Float Delay	3	20	
T10	D0–D31, DPn Write Data Valid Delay	3	19	
T11	D0–D31, DPn Write Data Float Delay	3	20	
T12	HOLD Valid Delay	3	19	
T13	CA and $\overline{BREQ}$ Setup Time	7		1, 2
T14	CA and $\overline{BREQ}$ Hold Time	3		1, 2
T15	$\overline{BS16}$ Setup Time	6		2
T16	$\overline{BS16}$ Hold Time	3		2
T17	$\overline{BRDY}$, $\overline{RDY}$ Setup Time	6		2
T18	$\overline{BRDY}$, $\overline{RDY}$ Hold Time	3		2
T19	D0–D31, DPn READ Setup Time	5		2
T20	D0–D31, DPn READ Hold Time	3		2
T21	AHOLD and HLDA Setup Time	8		1, 2
T22	AHOLD Hold Time	3		1, 2

AC Characteristics (Continued)

82596CA INPUT/OUTPUT SYSTEM TIMINGS

C_L on all outputs is 50 pF unless otherwise specified.
All timing requirements are given in nanoseconds.

Symbol	Parameter	33 MHz		Notes
		Min	Max	
T22a	HLDA Hold Time	3		1, 2
T23	RESET Setup Time	8		1, 2
T24	RESET Hold Time	3		1, 2
T25	INT/$\overline{\text{INT}}$ Valid Delay	1	20	
T26	CA and BREQ, $\overline{\text{PORT}}$ Pulse Width	2T1		1, 2, 3
T27	D0–D31 CPU $\overline{\text{PORT}}$ Access Setup Time	5		2
T28	D0–D31 CPU $\overline{\text{PORT}}$ Access Hold Time	3		2
T29	$\overline{\text{PORT}}$ Setup Time	7		2
T30	$\overline{\text{PORT}}$ Hold Time	3		2
T31	$\overline{\text{BOFF}}$ Setup Time	8		2
T32	$\overline{\text{BOFF}}$ Hold Time	3		2

NOTES:
1. RESET, HLDA, and CA are internally synchronized. This timing is to guarantee recognition at next clock for RESET, HLDA and CA.
2. All set-up, hold and delay timings are at maximum frequency specification Fmax, and must be derated according to the following equation for operation at lower frequencies:
   Tderated = (Fmax/Fopr) × T
   where:
   Tderate = Specifies the value to derate the specification.
   Fmax = Maximum operating frequency.
   Fopr = Actual operating frequency.
   T = Specification at maximum frequency.
   This calculation only provides a rough estimate for derating the frequency. For more detailed information, contact your Intel Sales Office for the data sheet supplement.
3. CA pulse width need only be 1 T1 wide if the set up and hold times are met; BREQ must meet setup and hold times and need only be 1 T1 wide.

TRANSMIT/RECEIVE CLOCK PARAMETERS

Symbol	Parameter	20 MHz		Notes
		Min	Max	
T36	$\overline{\text{TxC}}$ Cycle	50		1, 3
T38	$\overline{\text{TxC}}$ Rise Time		5	1
T39	$\overline{\text{TxC}}$ Fall Time		5	1
T40	$\overline{\text{TxC}}$ High Time	19		1, 3
T41	$\overline{\text{TxC}}$ Low Time	18		1, 3
T42	TxD Rise Time		10	4
T43	TxD Fall Time		10	4
T44	TxD Transition	20		2, 4
T45	$\overline{\text{TxC}}$ Low to TxD Valid		25	4, 6
T46	$\overline{\text{TxC}}$ Low to TxD Transition		25	2, 4
T47	$\overline{\text{TxC}}$ High to TxD Transition		25	2, 4
T48	$\overline{\text{TxC}}$ Low to TxD High (At End of Transition)		25	4

TRANSMIT/RECEIVE CLOCK PARAMETERS (Continued)

Symbol	Parameter	20 MHz		Notes
		Min	Max	
$\overline{\text{RTS}}$ AND $\overline{\text{CTS}}$ PARAMETERS				
T49	$\overline{\text{TxC}}$ Low to $\overline{\text{RTS}}$ Low, Time to Activate $\overline{\text{RTS}}$		25	5
T50	$\overline{\text{CTS}}$ Low to $\overline{\text{TxC}}$ Low, $\overline{\text{CTS}}$ Setup Time		20	
T51	$\overline{\text{TxC}}$ Low to $\overline{\text{CTS}}$ Invalid, $\overline{\text{CTS}}$ Hold Time	10		7
T52	$\overline{\text{TxC}}$ Low to $\overline{\text{RTS}}$ High		25	5
RECEIVE CLOCK PARAMETERS				
T53	$\overline{\text{RXC}}$ Cycle	50		1, 3
T54	$\overline{\text{RXC}}$ Rise Time		5	1
T55	$\overline{\text{RXC}}$ Fall Time		5	1
T56	$\overline{\text{RXC}}$ High Time	19		1
T57	$\overline{\text{RXC}}$ Low Time	18		1
RECEIVED DATA PARAMETERS				
T58	RXD Setup Time	20		6
T59	RXD Hold Time	10		6
T60	RXD Rise Time		10	
T61	RXD Fall Time		10	
$\overline{\text{CRS}}$ AND $\overline{\text{CDT}}$ PARAMETERS				
T62	$\overline{\text{CDT}}$ Low to $\overline{\text{TXC}}$ HIGH External Collision Detect Setup Time	20		
T63	$\overline{\text{TXC}}$ High to $\overline{\text{CDT}}$ Inactive, $\overline{\text{CDT}}$ Hold Time	10		
T64	$\overline{\text{CDT}}$ Low to Jam Start			10
T65	$\overline{\text{CRS}}$ Low to $\overline{\text{TXC}}$ High, Carrier Sense Setup Time	20		
T66	$\overline{\text{TXC}}$ High to $\overline{\text{CRS}}$ Inactive, $\overline{\text{CRS}}$ Hold Time (Internal Collision Detect)	10		
T67	$\overline{\text{CRS}}$ High to Jamming Start,			12
T68	Jamming Period			11
T69	$\overline{\text{CRS}}$ High to $\overline{\text{RXC}}$ High, $\overline{\text{CRS}}$ Inactive Setup Time	30		
T70	$\overline{\text{RXC}}$ High to $\overline{\text{CRS}}$ High, $\overline{\text{CRS}}$ Inactive Hold Time	10		

TRANSMIT/RECEIVE CLOCK PARAMETERS (Continued)

Symbol	Parameter	20 MHz		Notes
		Min	Max	
INTERFRAME SPACING PARAMETERS				
T71	Interframe Delay			9
EXTERNAL LOOPBACK-PIN PARAMETERS				
T72	$\overline{TXC}$ Low to $\overline{LPBK}$ Low		T36	4
T73	$\overline{TXC}$ Low to $\overline{LPBK}$ High		T36	4

NOTES:
1. Special MOS levels. V_{CIL} = 0.9V and V_{CIH} = 3.0V.
2. Manchester only.
3. Manchester. Needs 50% duty cycle.
4. 1 TTL load + 50 pF.
5. 1 TTL load + 100 pF.
6. NRZ only.
7. Abnormal end of transmission—CTS expires before RTS.
8. Normal end to transmission.
9. Programmable value:
   T71 = N_{IFS} • T36
   where: N_{IFS} = the IFS configuration value
   (if N_{IFS} is less than 12 then N_{IFS} is forced to 12).
10. Programmable value:
    T64 = (N_{CDF} • T36) + x • T36
    (If the collision occurs after the preamble)
    where:
    N_{CDF} = the collision detect filter configuration value,
    and
    x = 12, 13, 14, or 15
11. T68 = 32 • T36
12. Programmable value:
    T67 = (N_{CSF} • T36) + x • T36
    where: N_{CSF} = the Carrier Sense Filter configuration value, and
    x = 12, 13, 14, or 15
13. To guarantee recognition on the next clock.

82596CA BUS OPERATION

The following figures show the 82596CA basic bus cycle and basic burst cycle.

Please refer to the *32-Bit LAN Components Manual.*

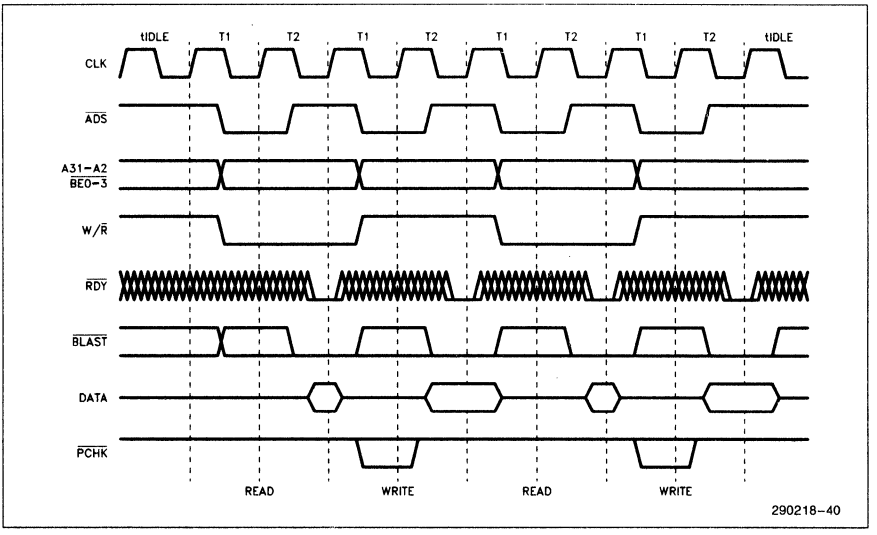


Figure 44. Basic 82596CA Bus Cycle

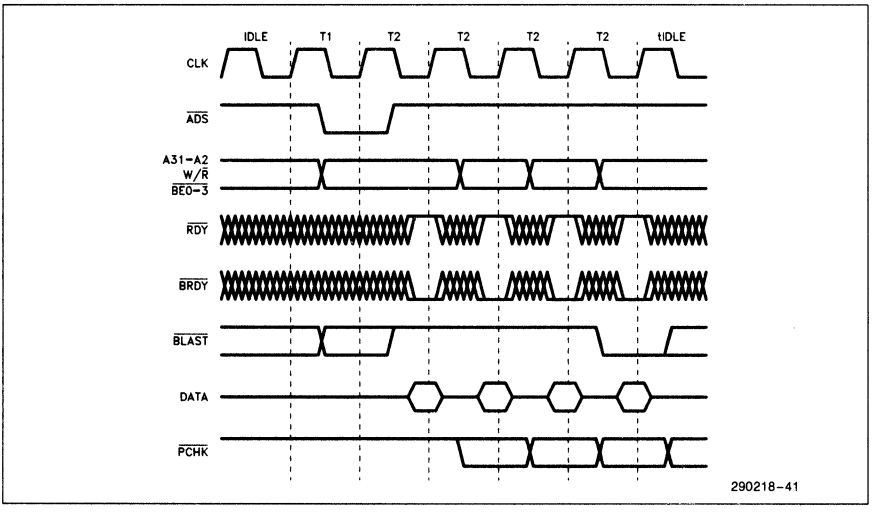


Figure 45. Basic 82596CA Burst Cycle

SYSTEM INTERFACE A.C. TIMING CHARACTERISTICS

The measurements should be done at:

- T_C = 0°C–85°C, V_{CC} = 5V ±10%, C = 50 pF unless otherwise specified.
- A.C. testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0".
- Timing measurements are made at 1.5V for both logic "1" and "0".
- Rise and Fall time of inputs and outputs signals are measured between 0.8V and 2.0V respectively unless otherwise specified.
- All timings are relative to CLK crossing the 1.5V level.
- All A.C. parameters are valid only after 100 μs from power up.

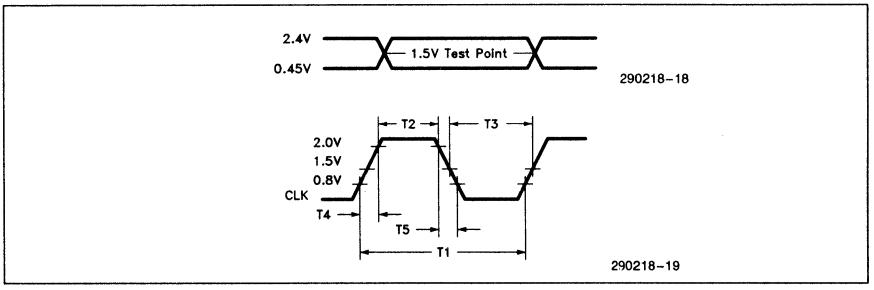


Figure 46. CLK Timings

Two types of timing specifications are presented below:

1. Input Timing—minimum setup and hold times.
2. Output Timings—output delays and float times from CLK rising edge.

Figure 47 defines how the measurements should be done:

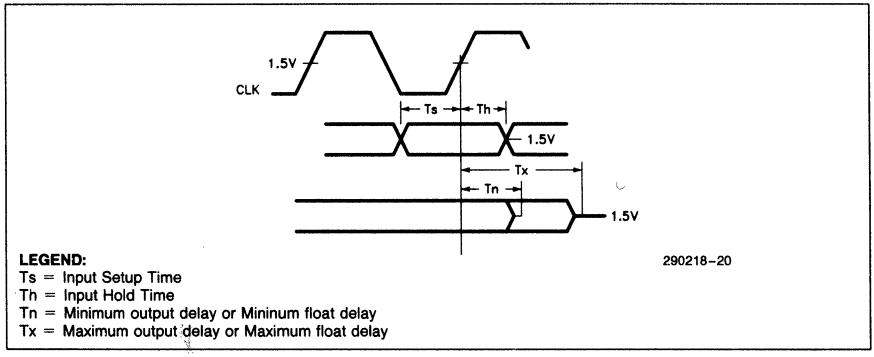


Figure 47. Drive Levels and Measurements Points for A.C. Specifications

Ts = T13, T15, T17, T19, T21, T23, T27, T29, T31
Th = T14, T16, T18, T20, T22, T22a, T24, T28, T30, T32
Tn = T6, T6a, T7, T8, T9, T10, T11, T12, T25
Tx = T6, T6a, T7, T8, T9, T10, T11, T12, T25

INPUT WAVEFORMS

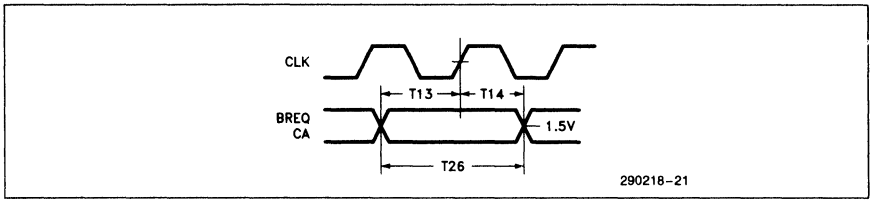


Figure 48. CA and BREQ Input Timing

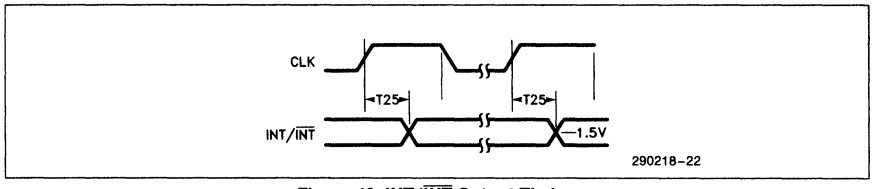


Figure 49. INT/$\overline{INT}$ Output Timing

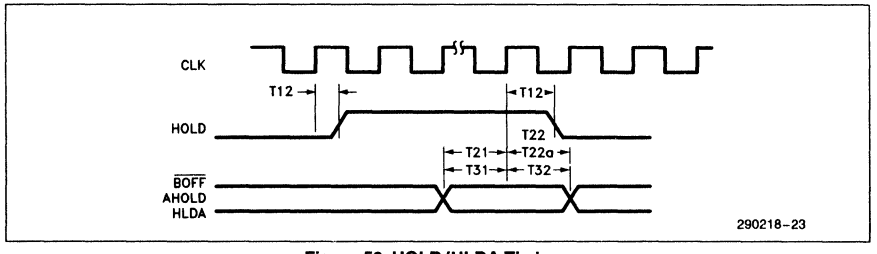


Figure 50. HOLD/HLDA Timings

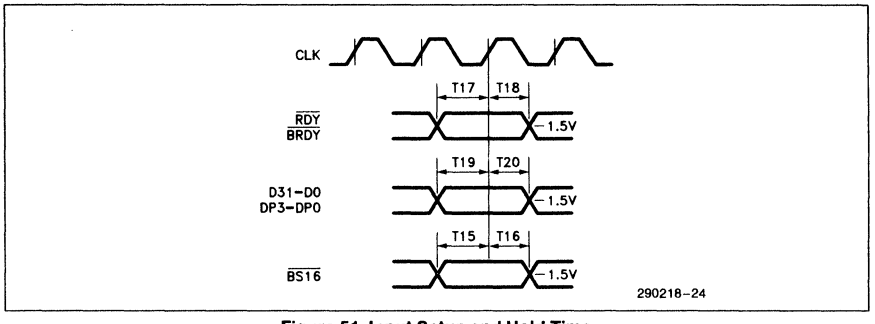


Figure 51. Input Setup and Hold Time

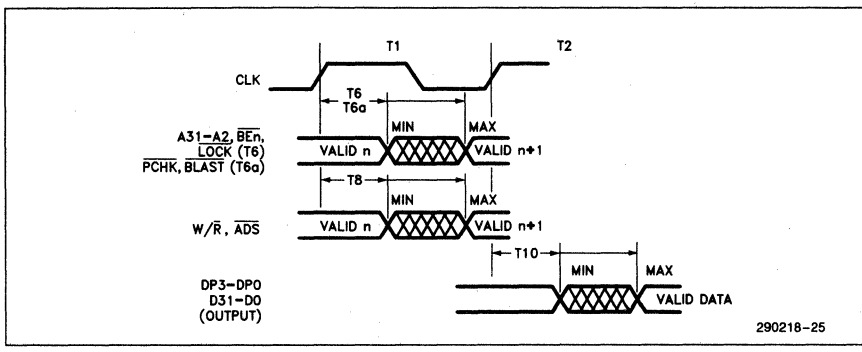


Figure 52. Output Valid Delay Timing

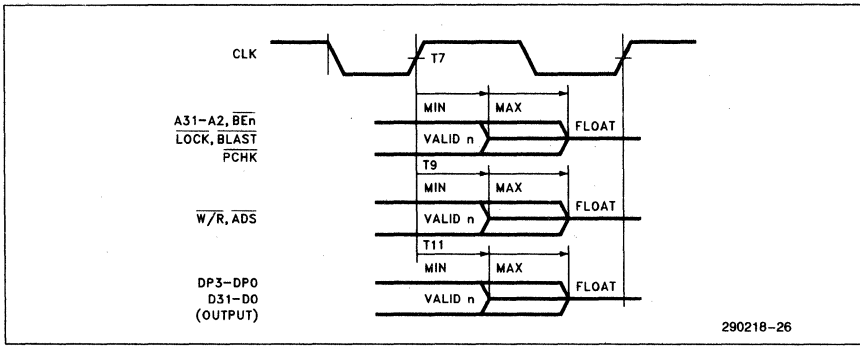


Figure 53. Output Float Delay Timing

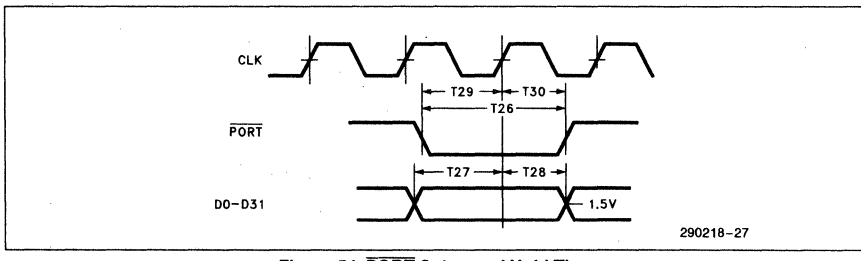


Figure 54. PORT Setup and Hold Time

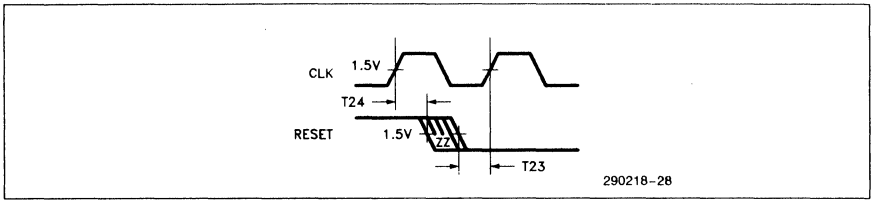


Figure 55. RESET Input Timing

SERIAL AC TIMING CHARACTERISTICS

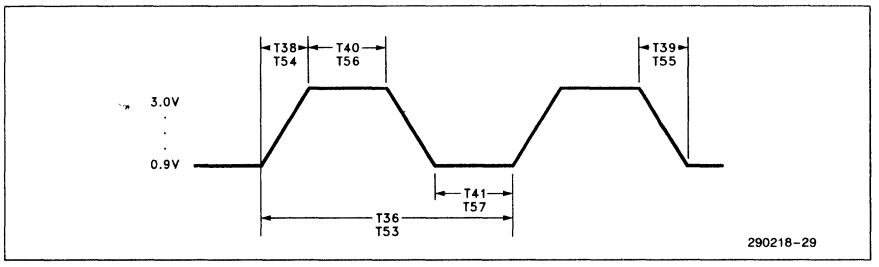


Figure 56. Serial Input Clock Timing

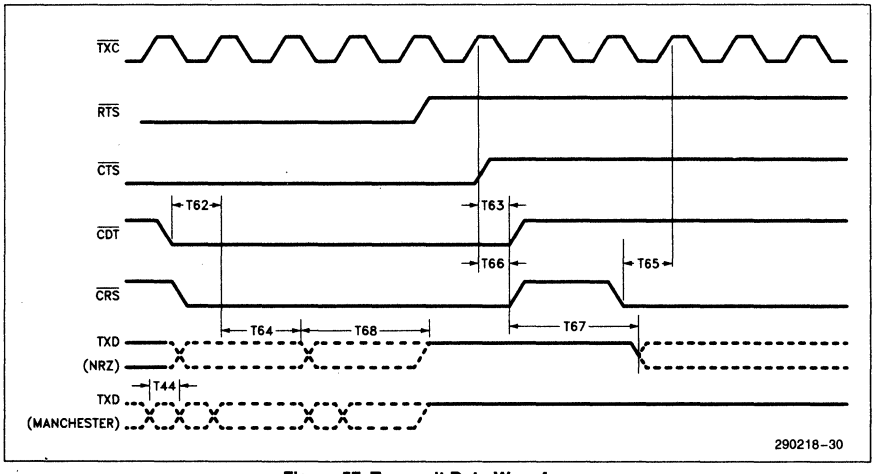


Figure 57. Transmit Data Waveforms

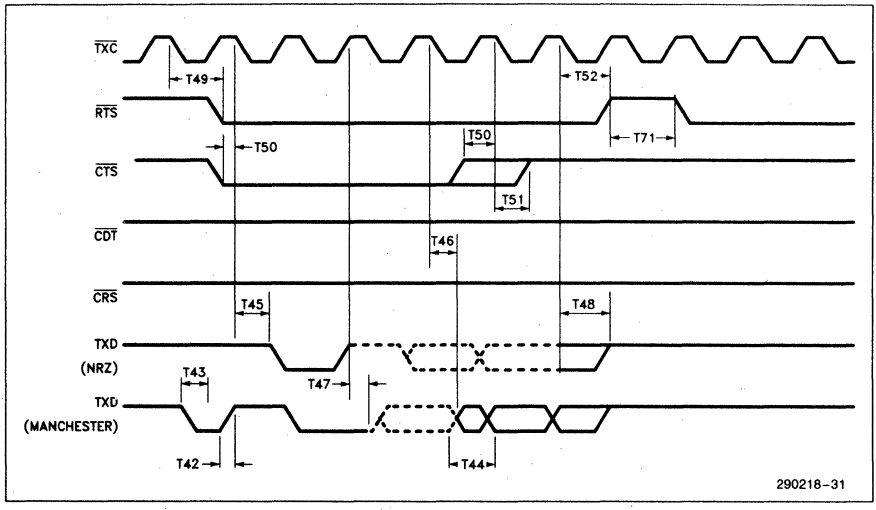


Figure 58. Transmit Data Waveforms

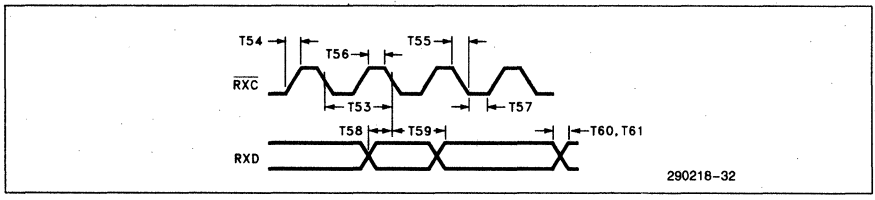


Figure 59. Receive Data Waveforms (NRZ)

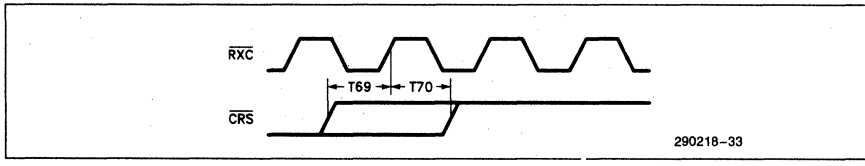
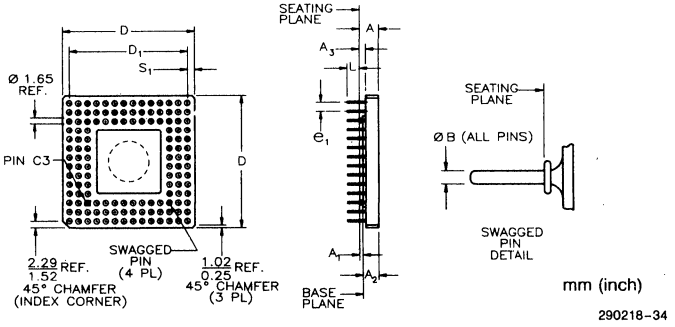


Figure 60. Receive Data Waveforms (CRS)

OUTLINE DIAGRAMS

132 LEAD CERAMIC PIN GRID ARRAY PACKAGE INTEL TYPE A



Family: Ceramic Pin Grid Array Package						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	3.56	4.57		0.140	0.180	
A_1	0.76	1.27	Solid Lid	0.030	0.050	Solid Lid
A_2	2.67	3.43	Solid Lid	0.105	0.135	Solid Lid
A_3	1.14	1.40		0.045	0.055	
B	0.43	0.51		0.017	0.020	
D	36.45	37.21		1.435	1.465	
D_1	32.89	33.15		1.295	1.305	
e_1	2.29	2.79		0.090	0.110	
L	2.54	3.30		0.100	0.130	
N		132			132	
S_1	1.27	2.54		0.050	0.100	
ISSUE	IWS 10/12/88					

Intel Case Outline Drawings
Plastic Quad Flat Pack (PQFP)
0.025 Inch (0.635mm) Pitch

Symbol	Description	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
N	Leadcount	68		84		100		132		164		196	
A	Package Height	0.160	0.170	0.160	0.170	0.160	0.170	0.160	0.170	0.160	0.170	0.160	0.170
A1	Standoff	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030
D, E	Terminal Dimension	0.675	0.685	0.775	0.785	0.875	0.885	1.075	1.085	1.275	1.285	1.475	1.485
D1, E1	Package Body	0.547	0.553	0.647	0.653	0.747	0.753	0.947	0.953	1.147	1.153	1.347	1.353
D2, E2	Bumper Distance	0.697	0.703	0.797	0.803	0.897	0.903	1.097	1.103	1.297	1.303	1.497	1.503
D3, E3	Lead Dimension	0.400 REF		0.500 REF		0.600 REF		0.800 REF		1.000 REF		1.200 REF	
D4, E4	Foot Radius Location	0.623	0.637	0.723	0.737	0.823	0.837	1.023	1.037	1.223	1.237	1.423	1.437
L1	Foot Length	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030
Issue	IWS Preliminary 12/12/88												INCH

Symbol	Description	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
N	Leadcount	68		84		100		132		164		196	
A	Package Height	4.06	4.32	4.06	4.32	4.06	4.32	4.06	4.32	4.06	4.32	4.06	4.32
A1	Standoff	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76
D, E	Terminal Dimension	17.15	17.40	19.69	19.94	22.23	22.48	27.31	27.56	32.39	32.64	37.47	37.72
D1, E1	Package Body	13.89	14.05	16.43	16.59	18.97	19.13	24.05	24.21	29.13	29.29	34.21	34.37
D2, E2	Bumper Distance	17.70	17.85	20.24	20.39	22.78	22.93	27.86	28.01	32.94	33.09	38.02	38.18
D3, E3	Lead Dimension	10.16 REF		12.70 REF		15.24 REF		20.32 REF		25.40 REF		30.48 REF	
D4, E4	Foot Radius Location	15.82	16.17	18.36	18.71	21.25	21.25	25.89	26.33	31.06	31.41	36.14	36.49
L1	Foot Length	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76
Issue	IWS Preliminary 12/12/88												mm

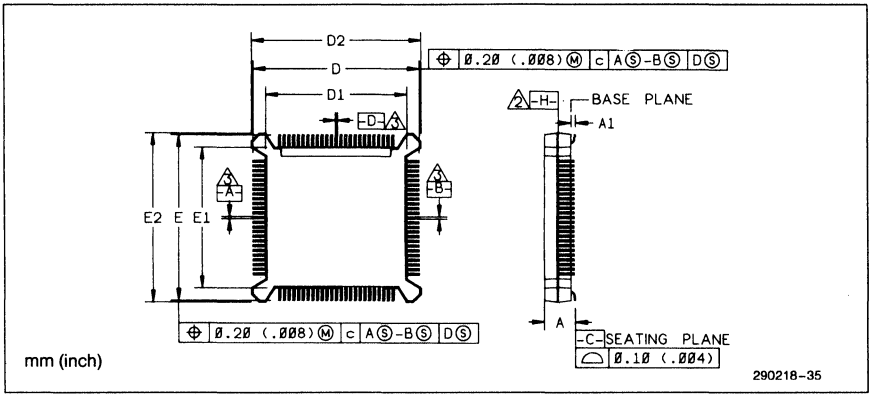


Figure 61. Principal Dimensions and Datums

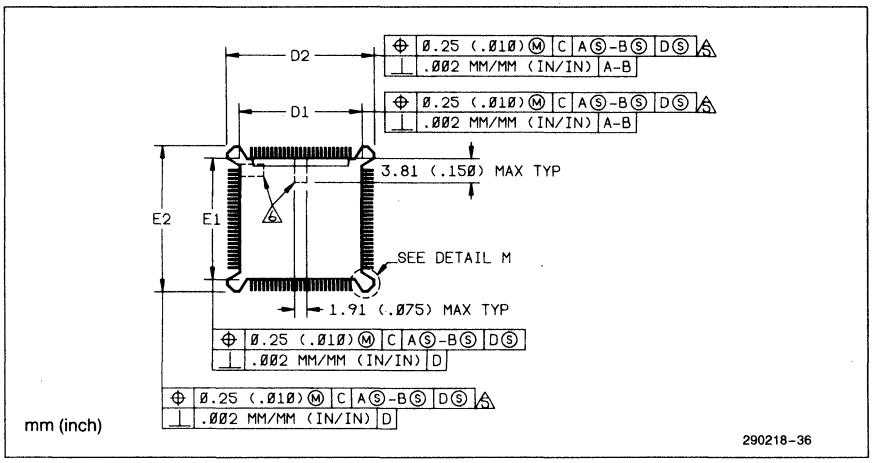


Figure 62. Molded Details

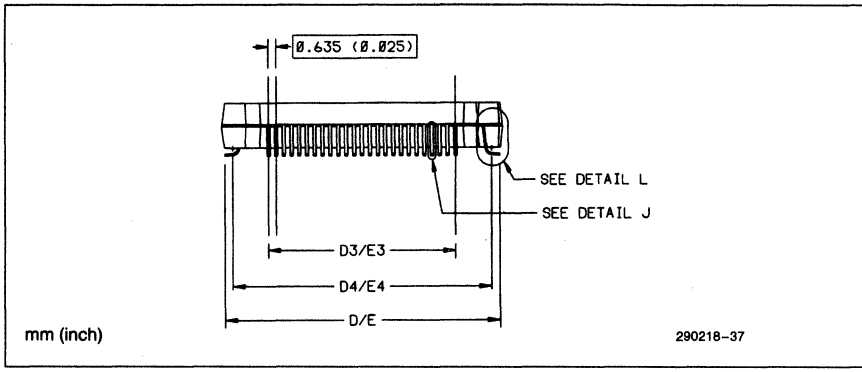


Figure 63. Terminal Details

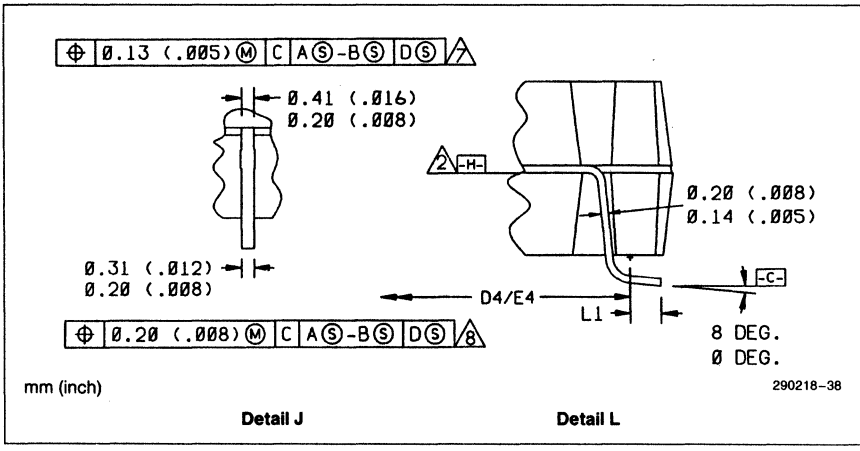
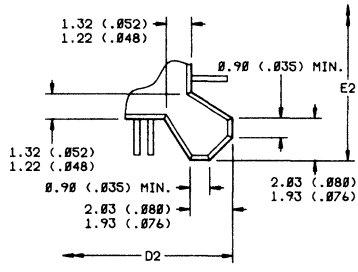


Figure 64. Typical Lead



mm (inch)

Figure 65. Detail M

REVISION HISTORY

The 82596 LAN Coprocessor data sheet version -003 contains updates and improvements to previous versions.

1. Added Pin Cross Reference table.
2. Added Bus Cycle figures.

Appendix

D

82588 High-Integration LAN Controller*

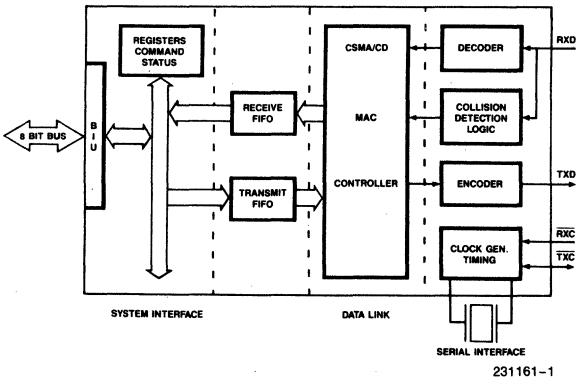
*Reprinted by permission of Intel Corporation. Copyright/Intel Corporation 1990.

- **Integrates ISO Layers 1 and 2**
  - **— CSMA/CD Medium Access Control (MAC)**
  - **— On-Chip Manchester, NRZI Encoding/Decoding**
  - **— On-Chip Logic Based Collision Detect and Carrier Sense**
- **Supports Mid-Range Industry Standard LANs**
  - **— StarLAN (IEEE 802.3 1BASE5)**
  - **— IBM/PC Network-Baseband and Broadband**
- **High Level Command Interface Offloads the CPU**
- **Efficient Memory Use Via Multiple Buffer Reception**
- **2 Clocks per Data Transfer**
- **User Configurable**
  - **— Up to 2 Mb/s Bit Rates with On-chip Encoder/Decoder (High Integration Mode)**
  - **— Up to 5 Mb/s with External Encoder/ Decoder (High Speed Mode)**
- **No TTL Glue Required with iAPX 186 and 188 Microprocessors**
- **Network Management and Diagnostics**
  - **— Short or Open Circuit Localization**
  - **— Station Diagnostics (External Loopback)**
  - **— Self Test Diagnostics Internal Loopback User Readable Registers**

The 82588 is a highly integrated CSMA/CD controller designed for cost sensitive, mid-range Local Area Network (LAN) applications, such as personal computer networks.

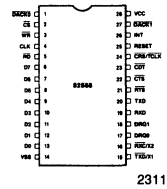
At data rates of up to 2 Mb/s, the 82588 provides a highly integrated interface and performs: CSMA/CD Data Link Control, Manchester, Differential Manchester or NRZI encoding/decoding, clock recovery; Carrier Sense, and Collision Detection. This mode is called "High Integration Mode." In the 82588 "High Speed Mode", the user can transfer data at a rate of up to 5 Mb/s. In this mode the physical link functions are done external to the 82588.

The 82588 is available in a 28 pin DIP and 44 lead PLCC package and fabricated in Intel's reliable HMOS II 5 volt technology.



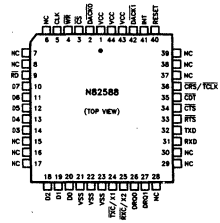
231161-1

Figure 1. 82588 Block Diagram



231161-2

Figure 2. 82588 Pin Configuration (DIP)



231161-31

Figure 3. 82588 Pin Configuration (PLCC)

Table 1. Pin Description

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
D7 D6 D5 D4 D3 D2 D1 D0	6 7 8 9 10 11 12 13	10 11 12 13 14 18 19 20	I/O	**DATA BUS:** The Data Bus lines are bi-directional three state lines connected to the system's Data Bus for the transfer of data, commands, status and parameters.
$\overline{RD}$	5	9	I	**READ:** Together with $\overline{CS}$, $\overline{DACK0}$ or $\overline{DACK1}$, Read controls data or status transfers out of the 82588 registers.
$\overline{WR}$	3	4	I	**WRITE:** Together with $\overline{CS}$, $\overline{DACK0}$ or $\overline{DACK1}$, Write controls data or command transfers into the 82588 registers.
$\overline{CS}$	2	3	I	**CHIP SELECT:** When this signal is LOW, the 82588 is selected by the CPU for transfer of command or status. The direction of data flow is determined by the $\overline{RD}$ or $\overline{WR}$ inputs.
CLK	4	5	I	**CLOCK:** System clock. TTL compatible signal.
RESET	25	40	I	**RESET:** A HIGH signal on this pin will cause the 82588 to terminate current activity. This signal is internally synchronized and must be held HIGH for at least four Clock cycles.
INT	26	41	O	**INTERRUPT:** Active HIGH signal indicates to the CPU that the 82588 is requesting an interrupt.
DRQ0	17	26	O	**DMA REQUEST (CHANNEL 0):** This pin is used by the 82588 to request a DMA transfer. DRQ0 remains HIGH as long as 82588 requires data transfers. Burst transfers are done by having the signal active for multiple transfers.
DRQ1	18	27	O	**DMA REQUEST (CHANNEL 1):** This pin is used by the 82588 to request a DMA transfer. DRQ1 remains HIGH as long as 82588 requires data transfers. Burst transfers are done by having the signal active or multiple transfers.
$\overline{DACK0}$	1	2	I	**DMA ACKNOWLEDGE (CHANNEL 0):** When LOW, this input signal from the DMA Controller notifies the 82588 that the requested DMA cycle is in progress. This signal acts like chip select for data and parameter transfer using DMA channel 0.
$\overline{DACK1}$	27	42	I	**DMA ACKNOWLEDGE (CHANNEL 1):** When LOW, this input signal from the DMA controller notifies the 82588 that the requested DMA cycle is in progress. This signal acts like chip select for data and parameter transfer using DMA channel 1.

Table 1. Pin Description (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
				High Integration Mode
X1/X2	15/16	24/25	I	**OSCILLATOR INPUTS:** These inputs may be used to connect a quartz crystal that controls the internal clock generator for the serial unit. X1 may also be driven by a MOS level clock whose frequency is 8 or 16 times the bit rate of Transmit/Receive data. X2 must be left floating if X1 has an external MOS clock.
				High Speed Mode
$\overline{TxC}$	15	24	I	**TRANSMIT CLOCK:** This signal provides timing information to the internal serial logic, depending upon the mode of data transfer. For NRZ encoding, data is transferred to the TxD pin on the HIGH to LOW clock transition. For Manchester encoding the transmitted bit center is aligned with the $\overline{TxC}$ LOW to HIGH transition.
$\overline{RxC}$	16	25	I	**RECEIVE CLOCK:** This signal provides timing information to the internal serial logic. NRZ data should be provided for reception (RxD). The state of the RxD pin is sampled on the HIGH to LOW transition of $\overline{RxC}$. The operating mode of the 82588 is defined when configuring the chip.
$\overline{TCLK}/\overline{CRS}$	24	36	I	In High Speed Mode, this pin is Carrier Sense, input CRS, and is used to notify the 82588 that there is activity on the serial link.
			O	In High Integration Mode, this pin is Transmit Clock, $\overline{TCLK}$, and is used to output the transmit clock.
$\overline{CDT}$	23	35	I	**COLLISION DETECT:** This input notifies the 82588 that a collision has occurred. It is sensed only if the 82588 is configured for external Collision Detect (external circuitry is then required for detecting the collision).
RxD	19	31	I	**RECEIVE DATA:** This pin receives serial data.
TxD	20	32	O	**TRANSMIT DATA:** This pin transmits data to the Serial Link. This signal is HIGH when not transmitting.
$\overline{RTS}$	21	33	O	**REQUEST TO SEND:** When this signal is LOW, the 82588 notifies an external interface that it has data to transmit. It is forced HIGH after a reset and when transmission is stopped.
$\overline{CTS}$	22	34	I	**CLEAR TO SEND:** CTS enables the 82588 to start transmitting data. Raising this signal to HIGH stops the transmission.
VCC	28	1, 43, 44		**POWER:** + 5V Supply
VSS	14	21, 22, 23		Ground

Table 1. Pin Description (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
NC		6 7 8 15 16 17 28 29 30 37 38 39		**NO CONNECT:** These pins are reserved for future use.

FUNCTIONAL DESCRIPTION

High Integration Mode

The 82588 LAN Controller is a highly integrated CSMA/CD controller for cost sensitive LAN applications such as personal computer networks. Included on chip is a programmable CSMA/CD controller, an NRZI and Manchester encoder/decoder with clock recovery, and two collision detection mechanisms. With the addition of simple transceiver line drivers or RF Modem, the 82588 performs all the major functions of the ISO Physical and Data Link Layers.

CSMA/CD Controller

The 82588 on-chip CSMA/CD controller is programmable, which allows it to operate in a variety of LAN environments, including industry standards such as StarLAN (IEEE 802.3 1BASE5) and the 2 Mb/s IBM PC Network (both baseband and broadband transmission). Programmable parameters include:

- Framing (End of Carrier of SDLC)
- Address field length
- Station priority
- Interframe spacing
- Slot time
- CRC-32 OR CRC-16

Encoder/Decoder

The on-chip NRZI and Manchester encoder/decoder supports data rates up to 2 Mb/s. Manchester encoding is typically used in baseband applications and NRZI is used in broadband applications.

Collision Detection

One of the 82588's unique features is its on-chip logic based collision detection. To ensure a high probability of collision detection two mechanisms are provided. The Code Violation method defines a collision when a transition edge occurs outside the area of normal transitions as specified by either the Manchester or NRZI encoding methods. Bit Comparison method compares the signature of the transmitted frame to the received frame signature (re-calculated by the 82588 while listening to itself). If the signatures are identical the frame is assumed to have been transmitted without a collision.

System Interface

In addition to providing the functions necessary for interfacing to the LAN, the 82588 has a friendly system interface that eases the design effort. First, the 82588 has a high level command interface; that is the CPU sends the 82588 commands such as Transmit or Configure. This means the designer does not have to write low level software to perform these tasks, and it offloads the CPU in the application. Second, the 82588 supports an efficient memory structure called Multiple Buffer Reception in which buffers are chained together while receiving frames. This is an important feature in applications with limited memory, such as personal computers. Third, the 82588 has two independent sixteen byte FIFO's, one for reception and one for transmission. The FIFO's allow the 82588 to tolerate bus latency. Finally the 82588 provides an eight byte data path that supports up to 4 Mbytes/second using external DMA.

Network Management & Diagnostics

The 82588 provides a rich set of diagnostic and network management functions including: internal and external loopback, channel activity indicators, optional capture of all frames regardless of destination address (Promiscuous Mode), capture of collided frames, (if address matches), and time domain reflectometry for locating fault points in the network cable. The 82588 register Dump command ensures reliable software by dumping the content of the 82588 registers into the system memory.

The next section will describe the 82588 system bus interface, the 82588 network interface, and the 82588 internal architecture.

82588/Host CPU Interaction

The CPU communicates with the 82588 through the system's memory and 82588's on-chip registers. The CPU creates a data structure in the memory, programs the external DMA controller with the start address and byte count of the block, and issues the command to the 82588.

The 82588 is optimized for operating with the iAPX 186/188, but due to the small number of hardware signals between the 82588 and the CPU, the 82588 can operate easily with other processors. The data bus is 8 bits wide and there is no address bus.

Chip Select and Interrupt lines are used to communicate between the 82588 and the host as shown in the Figure 3. Interrupt is used by the 82588 to draw the CPU's attention. The Chip Select is used by the CPU to draw the 82588's attention.

There are two kinds of transfer over the bus: Command/Status and data transfers. Command/Status transfers are always performed by the CPU. Data transfers are requested by the 82588, and are typically performed by a DMA controller. The table given in Figure 4 shows the Command/Status and data transfer control signals.

The CPU writes to 82588 using $\overline{CS}$ and $\overline{WR}$ signals. The CPU reads the 82588 status register using $\overline{CS}$ and $\overline{RD}$ signals.

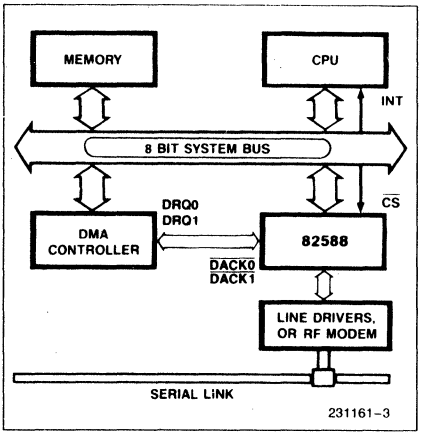


Figure 3. 82588/HOST CPU Interaction

To initiate an operation like Transmit or Configure (see Figure 5), a Write command from CPU to 82588 is issued by the CPU. A Read operation from CPU gives the status of the 82588. Although there are four status registers they're read using the same port in a round robin fashion (Figure 6).

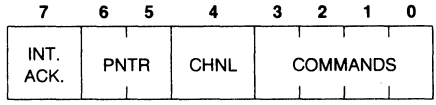
Any parameters or data associated with a command are transferred between the memory and 82588 using DMA. The 82588 has two data channels, each having Request and Acknowledge lines. Typically one channel is used to receive data and other to transmit data and perform all the other initialization and maintenance operations like Configure, Address Set-Up, Diagnose, etc. The channels are identical and can be used interchangeably.

When the 82588 requires access to the memory for parameter or data transfer it activates the DMA request lines and uses the DMA controller to achieve the data transfer. Upon the completion of an operation, the 82588 interrupts the CPU. The CPU then reads results of the operation (the status of the 82588).

Pin Name			Function
$\overline{CS}$*	$\overline{RD}$	$\overline{WR}$	
1	x	x	No transfer to/from Command/Status
0	1	1	
0	0	0	Illegal
0	0	1	Read from status register
0	1	0	Write to Command register
$\overline{DACK0}$[$\overline{DACK1}$]*	$\overline{RD}$	$\overline{WR}$	
1	x	x	No DMA transfer
0	1	1	
0	0	0	Illegal
0	0	1	Data Read from DMA channel 0 [or 1]
0	1	0	Data Write to DMA channel 0 [or 1]

* Only one of $\overline{CS}$, $\overline{DACK0}$ and $\overline{DACK1}$ may be active at any time.

Figure 4. Databus Control Signals and Their Functions



COMMAND REGISTER

COMMANDS		VALUE	COMMANDS		VALUE
NOP	—	0	ABORT	—	13
IA-SETUP	—	1	RECEIVER-ENABLE	—	8
CONFIGURE	—	2	ASSIGN NEXT BUF	—	9
MC-SETUP	—	3	RECEIVE-DISABLE	—	10
TRANSMIT	—	4	STOP-RECEPTION	—	11
TDR	—	5	RESET	—	14
DUMP	—	6	FIX PTR	—	15 (CHNL = 1)
DIAGNOSE	—	7	RLS PTR	—	15 (CHNL = 0)
RETRANSMIT	—	12			

Figure 5. Command Format and Operation Values

	7	6	5	4	3	2	1	0
Status 0	INT	RCV	EXEC	CHNL	EVENT			
Status 1				RESULT 1				
Status 2				RESULT 2				
Status 3	RCV CHNL	RCV STATE		BUFF CHNG NO. OF BUF		EXEC CHNL	EXEC STATE	

EVENTS	VALUE (STATUS 0)
IA-SETUP-DONE	— 1
CONFIGURE-DONE	— 2
MC-SETUP-DONE	— 3
TRANSMIT-DONE	— 4
TDR-DONE	— 5
DUMP-DONE	— 6
DIAGNOSE-PASSED	— 7
END OF FRAME	— 8
REQUEST NEXT BUFFER	— 9
RECEPTION ABORTED	— 10
RETRANSMIT-DONE	— 12
EXECUTION-ABORTED	— 13
DIAGNOSE-FAILED	— 15

Figure 6. Status Registers and Event Values

Transmitting a Frame

To transmit a frame, the CPU prepares a Transmit Data Block in memory as shown in Figure 7. Its first two bytes specify the length of the rest of the block. The next few bytes (Up to 6 bytes long) contain the destination address of the node it is being sent to. The rest of the block is the data field. The CPU programs the DMA controller with the start address of the block, length of the block and other control information and then issues the Transmit command to the 82588.

Upon receiving the command, the 82588 fetches the first two bytes of the block to determine the length of the block. If the link is free, and the first data byte was fetched, the 82588 begins transmitting the preamble and concurrently fetches the bytes from the Transmit Data Block and loads them into a 16 byte FIFO to keep them ready for transmitting. The FIFO is a buffer between the serial and parallel part of the 82588. The on-chip FIFOs help the 82588 to tolerate system bus latency as well as provide efficient usage of system bandwidth.

The destination address is sent out after the preamble. This is followed by the source or the station individual address, which is stored earlier on the 82588 using the IA-SETUP command. After that, the entire information field is transmitted followed by a CRC field calculated by the 82588. If during the transmission of the frame, a collision is encountered, then the transmission is aborted and a jam pattern is sent out after completion of the preamble. The 82588 generates an Interrupt indicating the experience of a collision and the frame has to be re-transmitted. Retransmission is done by the CPU exactly as the Transmit command except the Re-Transmit command keeps track of the number of collisions encountered. When the 82588 gets the Retransmit command and the exponential back-off time is expired, the 82588 transmits the frame again. The transmitted frame can be coded to either Manchester, Differential Manchester or NRZI methods.

Collision Detection

The 82588 eliminates the need for external collision detection logic, in most applications, while easing or eliminating the need for complex transceivers. Two algorithms are used for collision detection: Bit Comparison and Code Violation. The Bit Comparison Method is useful in Broadband networks where there are separate transmit and receive channels. Bit Comparison compares the "signature" of the transmitted data and received data at the end of the collision window in any network configuration. This algorithm calculates the CRC after a programmable number of transmitted bits, holds this CRC in a register, and compares it with received data's CRC. A CRC or "signature" difference indicates a collision. The code violation is detected if the encoding of the received data has any bit that does not fit the encoding rules. The code violation method is useful in short bus topology and serial backplane applications where bit attenuation over the bus is negligible.

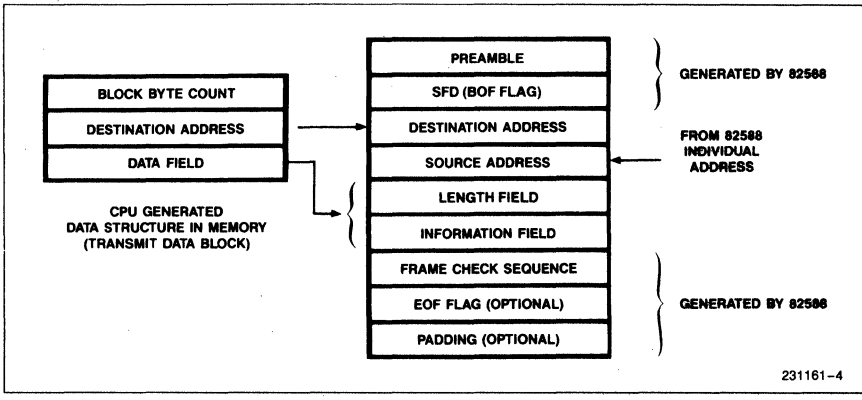


Figure 7. The 82588 Frame Structure and location of Data element in System Memory

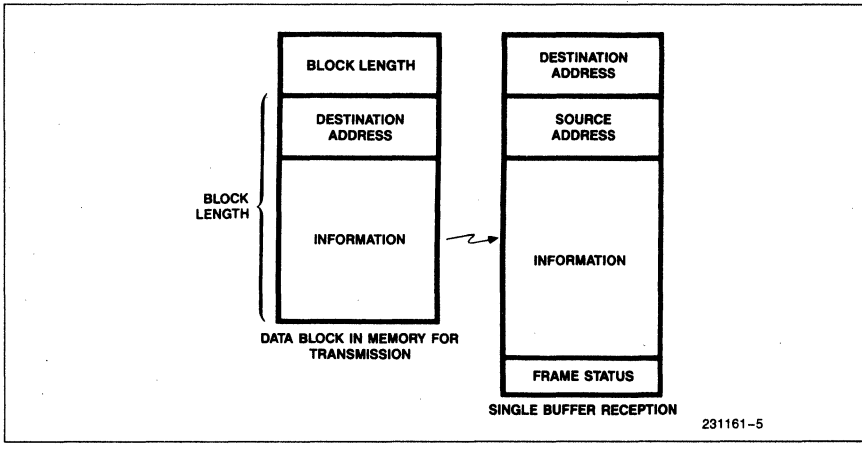


Figure 8. Single Buffer Reception

Receiving a Frame

The 82588 can receive a frame when its receiver has been enabled. The received frame is decoded by either on-chip Manchester, Differential Manchester or NRZI decoders in High Integration Mode and NRZI in High Speed Mode. The 82588 checks for an address match for an individual address, a Mulitcast address or a Broadcast address. In the Promiscuous mode the 82588 receives all frames. Only when the address match is successful does the 82588 transfer the frame to the memory using the DMA controller. Before enabling the receiver, the CPU makes a memory buffer area available to the Receive Unit and programs the starting address of the DMA controller. The received frame is transferred to the memory buffer in the format shown in Figure 8. This method of reception is called "Single Buffer" reception. The entire frame is contained in one continuous buffer. Upon completion of reception the total number of bytes written into the memory buffer is loaded into status registers 1 and 2 and the status of the reception itself is appended to the received frame. An interrupt to the CPU follows.

If the frame size is unknown, memory usage can be optimized by using "Multiple Buffer" reception.

This way the user does not have to allocate large memory space for short frames. Instead, the 82588 can dynamically allocate memory space as it receives frames. This method requires both DMA channels alternately to receive the frame. As the frame reception starts, the 82588 interrupts the CPU and automatically requests assignment of the next sequential buffer. The CPU does this and loads the second DMA channel with the next buffer information so that the 82588 can immediately switch to the other channel as soon as the current buffer is full. When the 82588 switches from the first to the second buffer it again interrupts the CPU requesting it to allocate another buffer on the other (previous) channel in advance. This process continues until the entire frame is received. The received frame is spread over multiple memory buffers. The link between the buffers is easily maintained by the CPU using a buffer chain descriptor structure in memory (see Figure 9).

This dynamic (pre) allocation of memory buffers results in efficient use of available storage when handling frames of widely differing sizes. Since the buffers are pre-allocated one block in advance, the system is not time critical.

80188 Based System

Figure 10 shows a high performance, high-integration configuration of the 82588 with the 80188 in a typical iAPX188-based microcomputer. The 80188 controls the 82588, as well as providing DMA control services for data transfer, using its on-chip two channel DMA controller.

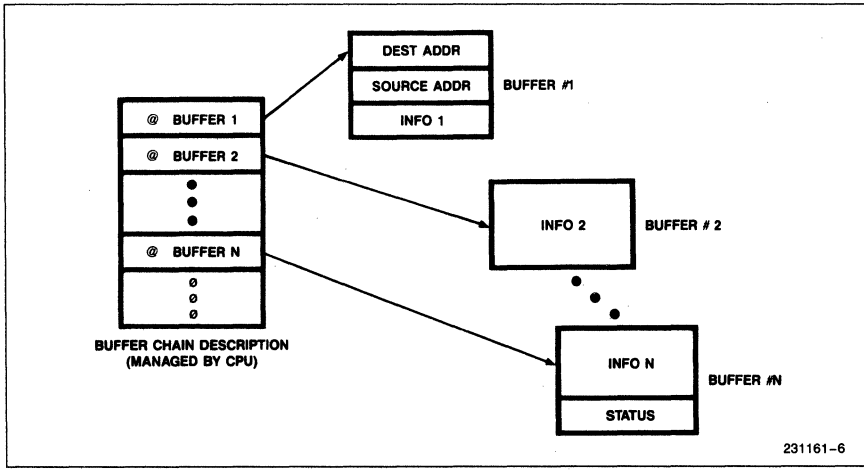


Figure 9. Multiple Buffer Reception

Link Interface

The Serial Interface Mode configuration parameter selects either a highly integrated Direct Link interface (High Integration Mode) or a highly flexible Transceiver Interface (High Speed Mode).

Application

In the High Integration Mode it is possible to connect the 82588 on a very short "Wired OR" link, on a longer twisted pair cable, or a broadband connection.

Twisted Pair Connection

The link consists of a twisted pair that interconnects the 82588. The transmit data pin is connected via a driver and the receive data pin is connected via a buffer. The twisted pair must be properly terminated to prevent reflections.

In the minimum configuration, TxD and RxD are connected to the twisted pair and $\overline{CTS}$ is grounded. The 82588 may control the driver with the $\overline{RTS}$ pin. It is also possible to use external circuitry for performing collision detection, and feeding it to the 82588 through the $\overline{CDT}$ pin.

Broadband Connection

The 82588 supports data communications over a broadband link in both its modes. Proper MODEM interface should be provided. Collision Detection by Bit Comparison, in High Interface Mode, can be applied to transmission over broadband links.

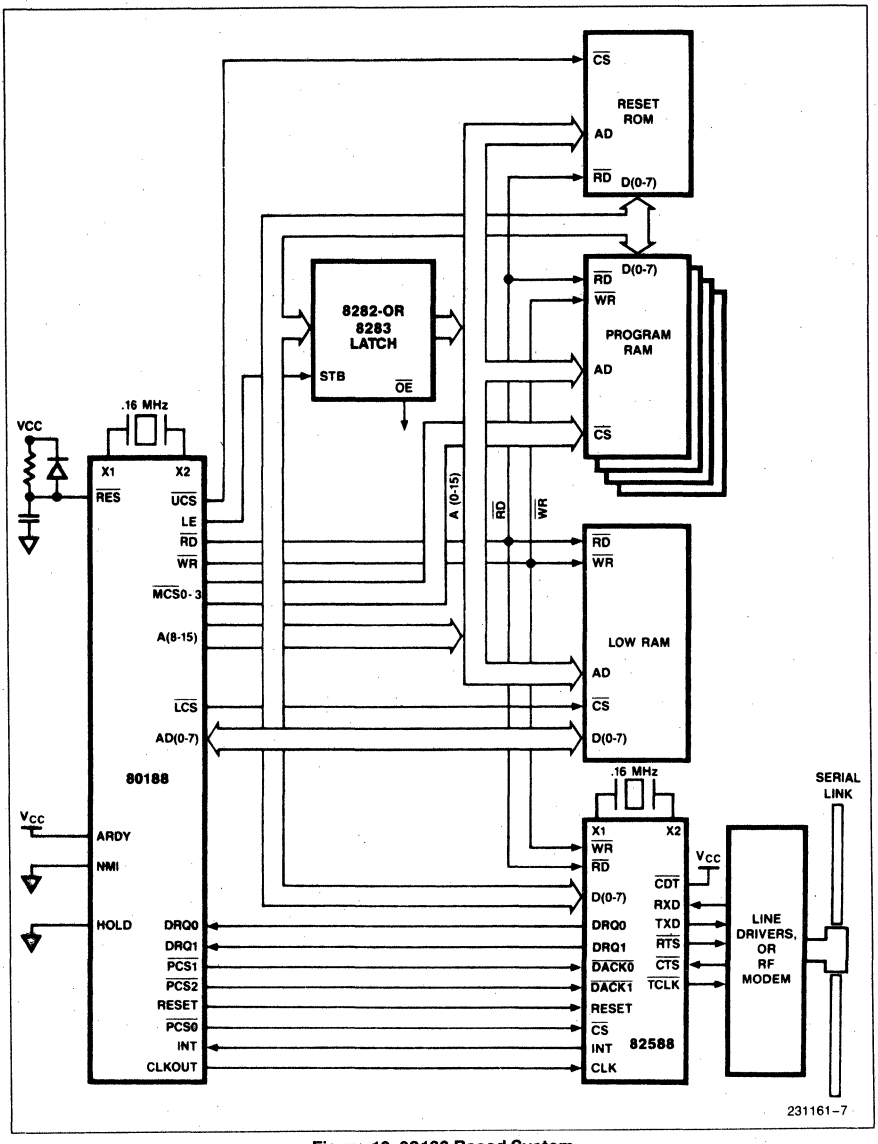


Figure 10. 80188 Based System

Absolute Maximum Ratings*

Ambient Temperature Under Bias ....0°C to +70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin With
Respect to Ground.....−1.0V to 7V
Power Dissipation.....1.7 Watts

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. Characteristics

(T_A = 0°C to +70°C; T_C (DIP) = 52°C to 108°C, T_C (PLCC) = 63°C to 116°C; VCC = +5V ±10%)

$\overline{TxC}$, $\overline{RxC}$ have MOS levels (See VMIL, VMIH). All other signals have TTL levels (See VIL, VIH, VOL, VOH).

Symbol	Parameter	Min	Max	Units	Test Conditions
VIL	Input Low Voltage (TTL)	−0.5	+0.8	V	
VIH	Input High Voltage (TTL)	2.0	VCC + 0.5	V	
VOL	Output Low Voltage (TTL)		0.45	V	IOL = 2.0 mA
VOH	Output High Voltage (TTL)	2.4		V	IOH = −400 μA
VMIL	Input Low Voltage (MOS)	−0.5	0.6	V	
VMIH	Input High Volatge (MOS)	3.9	VCC + 0.5	V	
ILI	Input Leakage Current		+10	μA	0 = VIN = VCC
ILO	Output Leakage Current		±10	μA	0.45 = VOUT = VCC
ICC	Power Supply Current		400 300	mA mA	T_A = 0°C T_A = +70°C

A.C. Characteristics

(T_A = 0°C to +70°C; T_C (DIP) = 52°C to 108°C, T_C (PLCC) = 63°C to 116°C; VCC = +5V ±10%)

System Clock Parameters

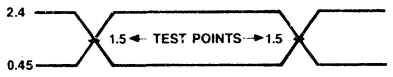
Symbol	Parameter	Min	Max	Units	Test Conditions
T1	CLK Cycle Period	125		ns	
T2	CLK Low Time	53	1000	ns	*5
T3	CLK High Time	53		ns	*6
T4	CLK Rise Time		15	ns	*1
T5	CLK Fall Time		15	ns	*2

A.C. Characteristics (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
Reset Parameters					
T6	Reset Active to Clock Low	20		ns	*3
T8	Reset Pulse Width	4T1		ns	
T9	Control Inactve After Reset		T1	ns	
Interrupt Timing Parameters					
T10	CLK High to Interrupt Active		85	ns	*4
T11	$\overline{WR}$ Idle to Interrupt Idle		85	ns	*4
Write Parameters					
T12	$\overline{CS}$ or $\overline{DACK0}$ or $\overline{DACK1}$ Setup to $\overline{WR}$ Low	0		ns	
T13	$\overline{WR}$ Pulse Width	95		ns	
T14	$\overline{CS}$ or $\overline{DACK0}$ or $\overline{DACK1}$ Hold After $\overline{WR}$ High	0		ns	
T15	Data Setup to $\overline{WR}$ High	75		ns	
T16	Data Hold After $\overline{WR}$ High	0		ns	
Read Parameters					
T17	$\overline{CS}$ or $\overline{DACK0}$ or $\overline{DACK1}$ Setup to $\overline{RD}$ Low	0		ns	
T18	$\overline{RD}$ Pulse Width	95		ns	
T19	$\overline{CS}$ or $\overline{DACK0}$ or $\overline{DACK1}$ Address Valid After $\overline{RD}$ High	0		ns	
T20	$\overline{RD}$ Low to Data Valid		80	ns	*7
T21	Data Float After $\overline{RD}$ High		55	ns	*7
DMA Parameters					
T22	CLK Low to DRQ0 or DRQ1 Active		85	ns	*4
T23	$\overline{WR}$ or $\overline{RD}$ Low to DRQ0 or DRQ1 Inactive		60	ns	*4

NOTES:
*1—0.8V–2.0V
*2—2.0V–0.8V
*3—to guarantee recognition at next clock
*4—CL = 50 pF
*5—measured at 1.5V
*6—measured at 1.5V
*7—CL = 20 pF–200 pF

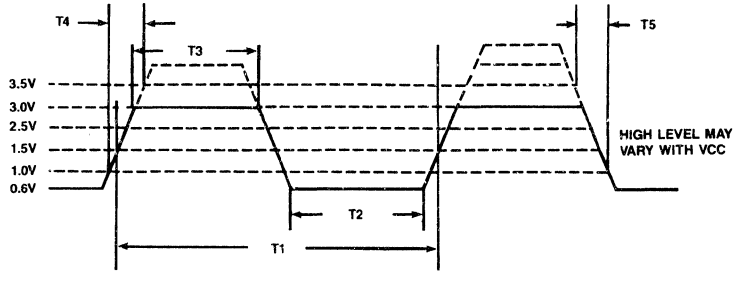
A.C. TESTING INPUT/OUTPUT WAVEFORM



231161-8

AC Testing Inputs are Driven at 2.4V for a Logic 1 and 0.45V for a Logic Q Timing Measurements are Made at 1.5V for Both a Logic 1 and Q:
Rise and Fall Time of Input/Output Signals are Measured Between 0.8V to 2.0V Respectively.

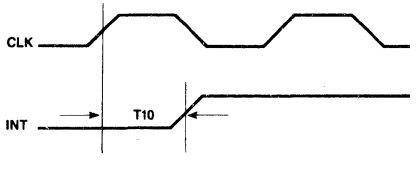
TTL Input/Output Voltage Levels for Timing Measurements



231161-9

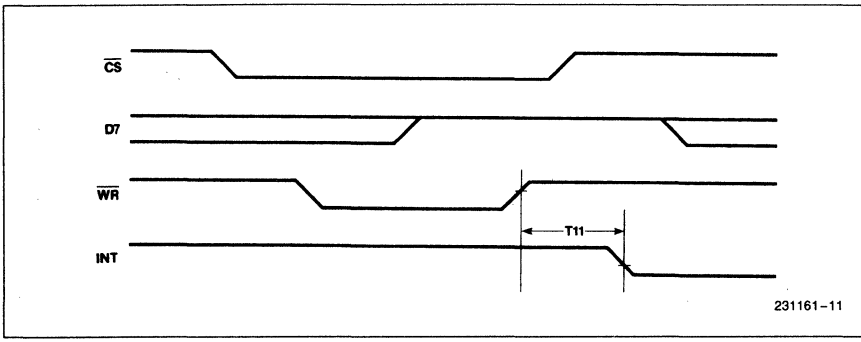
Rise and Fall Time of Input Signals are Measured Between 1.0V to 3.5V Respectively.

Clocks MOS Input Voltage Levels for Timing Measurements

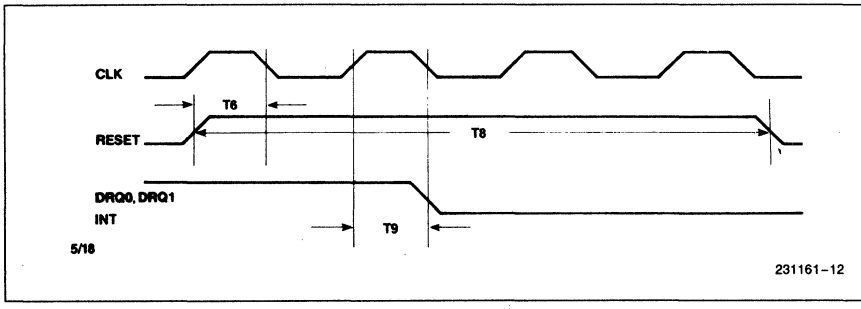


231161-10

Interrupt Timing (Going Active)



Interrupt Timing (Going Inactive)



Reset Timing

Serial Interface A.C. Timing Characteristics
High Integration Mode

$\overline{TFC}$ is the crystal or serial clock input at the X1 pin. When a serial clock is provided at the X1 pin, the maximum capacitive load allowed on the X2 pin is 15 pF.
$\overline{TFC}$ Frequency Range:

For Oscillator Frequency = 1 to 16 MHz (High)		
	×8 Sampling	×16 Sampling
$\overline{TCLK}$ Frequency	0.125 – 2 MHz	62.5 kHz – 1 MHz
T29 = $\overline{TCLK}$ Cycle Time	8 × T24	16 × T24
T30 = $\overline{TCLK}$ High Time	T24 (Typically)	T24 (Typically)
T31 = $\overline{TCLK}$ Low time	7 × T24 (Typically)	15 × T24 (Typically)

For Oscillator Frequency = 0 to 1 MHz (Low)*		
	×8 Sampling	×16 Sampling
$\overline{TCLK}$ Frequency	0 – 0.125 MHz	0 – 6.25 kHz
T29 = $\overline{TCLK}$ Cycle Time	8 × T24	16 × T24
T30 = $\overline{TCLK}$ High Time	T25 (Typically)	T25 (Typically)
T31 = $\overline{TCLK}$ Low Time	7 × T24 + T26 (Typically)	15 × T24 + T26 (Typically)
*A non-symmetrical clock should be provided so that T25 is less than 1000 ns. T24 = Serial Clock Period T25 = Serial Clock High Time T26 = Serial Clock Low Time		

High Speed Mode

- Applies for $\overline{TxC}$, $\overline{RxC}$
- f max = 5 MHz ±100 ppm
- For Manchester, symmetry is required: T_{63}, $T_{64} = \frac{1}{2f} \pm 5\%$

High Integration Mode

Symbol	Parameter	Min	Max	Units	Test Conditions
External (Fast) Clock Parameters					
T24	Fast Clock Cycle	62.5		ns	*1
T25	$\overline{TFC}$ High Time	18.5	1000	ns	*1, *14
T26	$\overline{TFC}$ Low Time	23.5		ns	*1
T27	$\overline{TFC}$ Rise Time		5	ns	*1
T28	$\overline{TFC}$ Fall Time		5	ns	*1
Transmit Clock Parameters					
T29	Transmit Clock Cycle	500		ns	*3, *12
T30	$\overline{TCLK}$ High Time	*8	1070	ns	*3
T31	$\overline{TCLK}$ Low Time	*9			*3
T32	$\overline{TCLK}$ Rise Time		15	ns	*3
T33	$\overline{TCLK}$ Fall Time		15	ns	*3

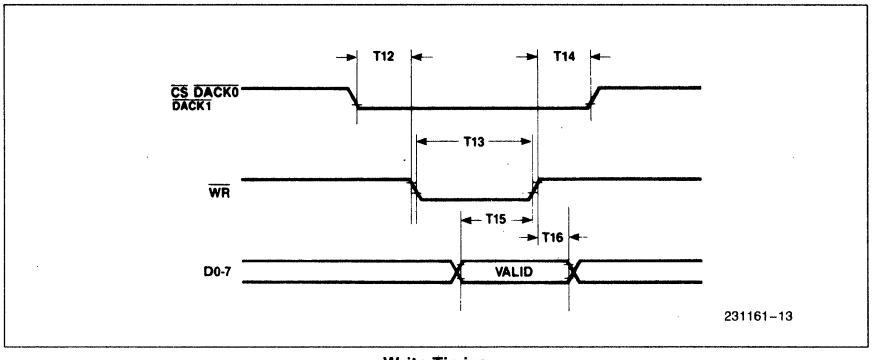
High Integration Mode (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
Transmit Data Parameters (Manchester, Differential Manchester)					
T34	TxD Transition-Transition	4T24-10		ns	*12
T35	$\overline{TCLK}$ Low to TxD Transition Half Bit Cell		*10		*2, *12
T36	$\overline{TCLK}$ Low to TxD Transition Full Bit Cell		*11		*2, *12
T37	TxD Rise Time		15	ns	*2
T38	TxD Fall Time		15	ns	*2
Transmit Data Parameters (NRZI)					
T39	TxD Transition-Transition	8T24-10		ns	*12
T40	$\overline{TCLK}$ Low to TxD Transition		*10		*2, *12
T41	$\overline{TxD}$ Rise Time		15	ns	*2
T42	TxD Fall Time		15	ns	*2
$\overline{RTS}$, $\overline{CTS}$, Parameters					
T43	$\overline{TCLK}$ Low To $\overline{RTS}$ Low		*10		*3, *12
T44	$\overline{CTS}$ Low to $\overline{TCLK}$ Low $\overline{CTS}$ Setup Time	65		ns	
T45	$\overline{TCLK}$ low to $\overline{RTS}$ High		*10		*3, *12
T46	$\overline{TCLK}$ Low to $\overline{CTS}$ Invalid. $\overline{CTS}$ Hold Time	20		ns	*4, *13
T47	$\overline{CTS}$ High to $\overline{TCLK}$ Low. $\overline{CTS}$ Setup Time to Stop Transmission	65		ns	*4
IFS Parameters					
T48	Interframe Delay	*5			
Collision Detect Parameter					
T49	$\overline{CDT}$ Low to $\overline{TCLK}$ High. External Collision Detect Setup Time	50		ns	*13
T50	$\overline{CDT}$ High to $\overline{TCLK}$ Low	50		ns	*13
T51	$\overline{TCLK}$ High to $\overline{CDT}$ Inactive. $\overline{CDT}$ Hold Time	20		ns	*13

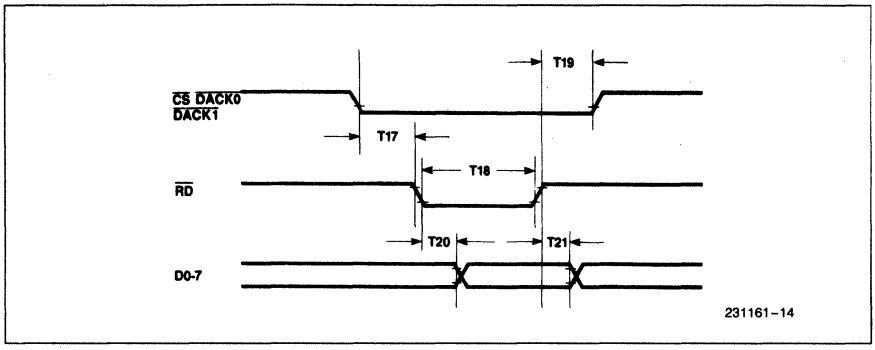
High Integration Mode (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
Collision Detect Parameters (Continued)					
T52	$\overline{CDT}$ Low to Jamming Start		*6		
T53	Jamming Period	*7			
Received Data Parameters (Manchester)					
T54	RxD Transition-Transition	4T24		ns	*12
Received Data Parameters (Manchester)					
T55	RxD Rise Time		10	ns	*1
T56	RxD Fall Time		10	ns	*1
Received Data Parameters (NRZI)					
T57	RxD Transition-Transition	8T24		ns	*12
T58	RxD Rise Time		10	ns	*1
T59	RxD Fall Time		10	ns	*1

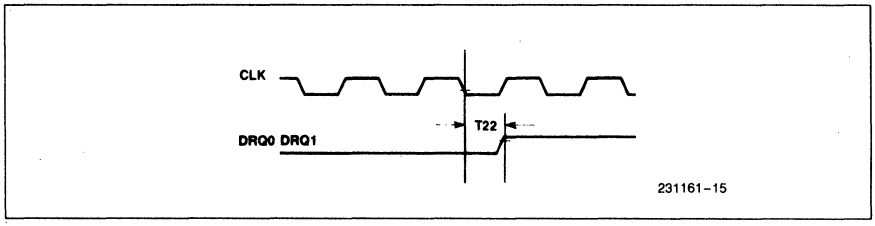
NOTES:
*1—MOS levels.
*2—1 TTL load + 50 pF.
*3—1 TTL load + 100 pF.
*4—Abnormal end to transmission: $\overline{CTS}$ expires before $\overline{RTS}$.
*5—Programmable value: T48 = NIFS × T29 (ns) NIFS—the IFS configuration value.
If NIFS is less than 12, then it is enforced to 12.
*6—Programmable value:
T52 = NCDF × T29 + (12 to 15) × T29 (if collision occurs after preamble).
*7—T53 = 32 × T29
*8—Depends on T24 frequency range:
High Range: T24 − 10
Low Range: T25 − 10
*9—T31 = T29 − T30 − T32 − T33
*10—2T24 + 40 ns
*11—6T24 + 40 ns
*12—For ×16 sampling clock parameter minimum value should be multiplied by a factor of 2.
*13—To guarantee recognition on the next clock.
*14—62.5 ns minimum in Low Range.



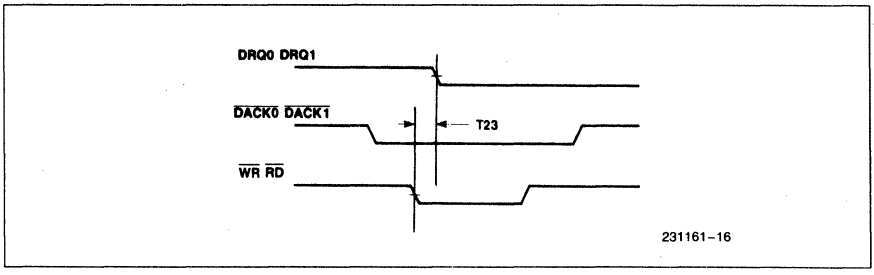
Write Timing



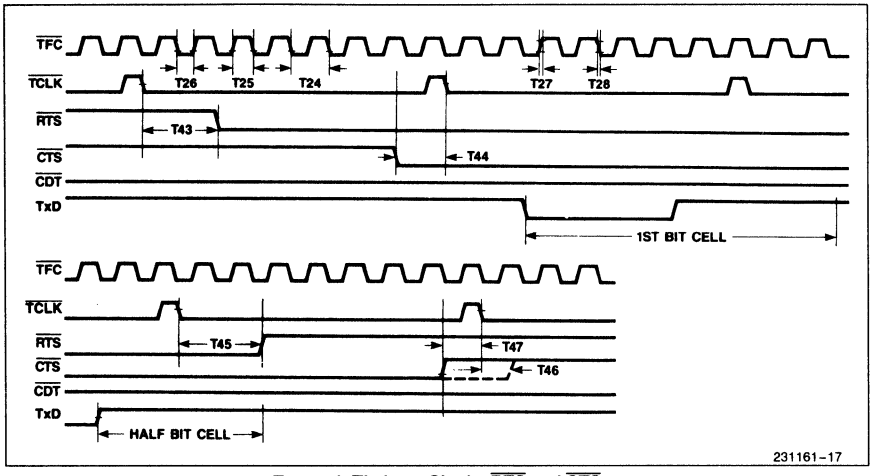
Read Timing



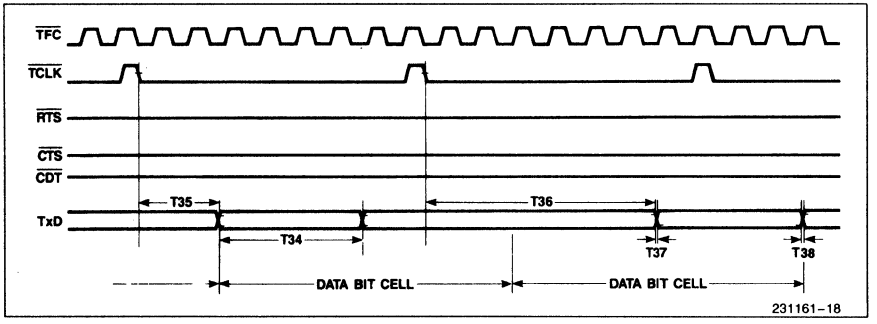
DMA Request (Going Active)



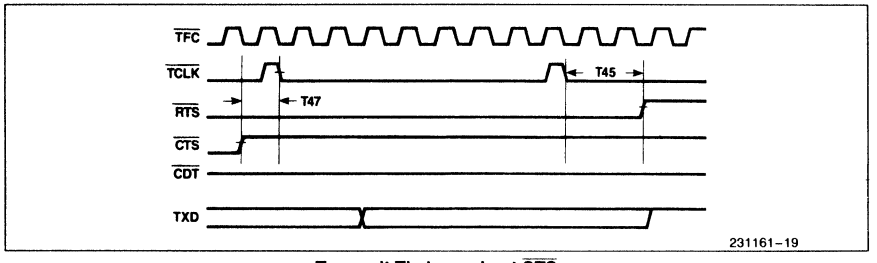
DMA Request (Going Inactive)



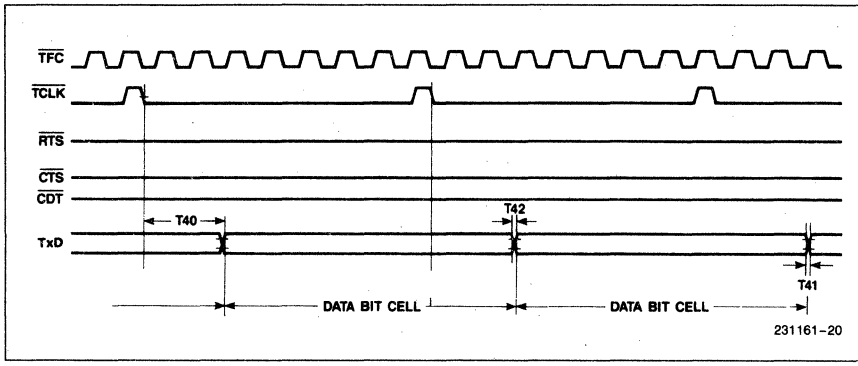
Transmit Timings: Clocks $\overline{RTS}$ and $\overline{CTS}$



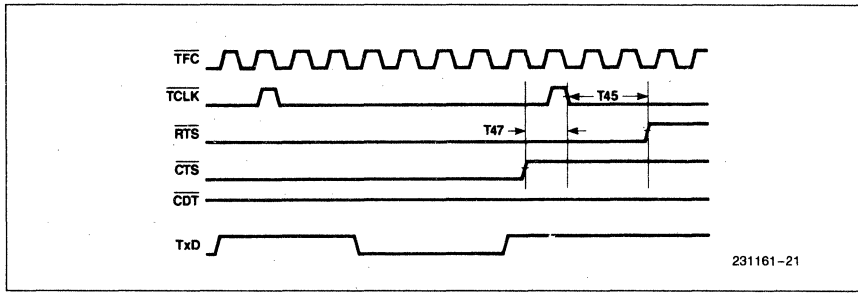
Transmit Timings—Manchester Data Encoding



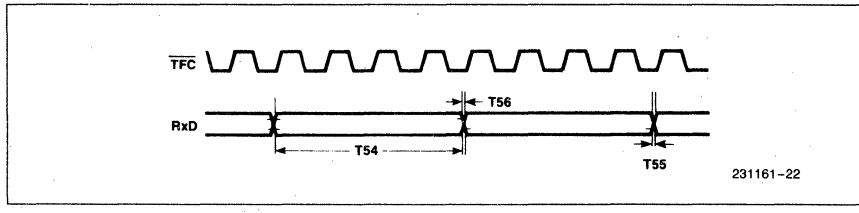
Transmit Timings—Lost $\overline{CTS}$



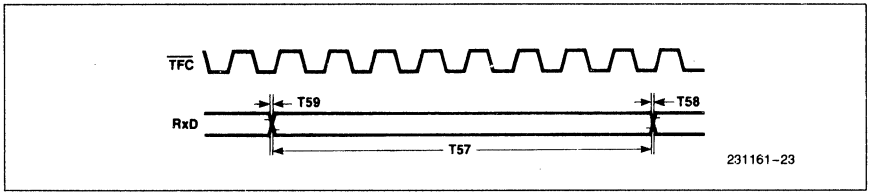
Transmit Timings—NRZI Data Encoding



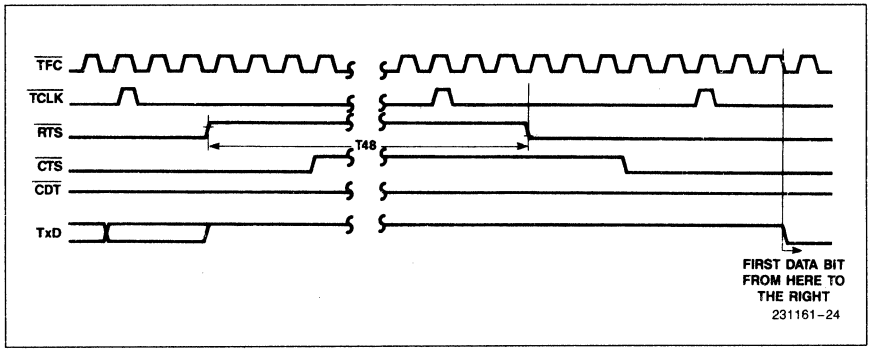
Transmit Timings—Lost CTS



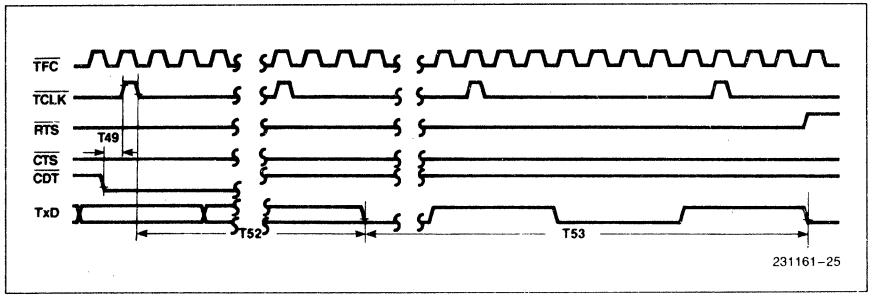
Receive Data Timings (Manchester)



Receive Data Timings (NRZI)



Transmit Timings—Interframe Spacing



Transmit Timings—Collision Detect and Jamming

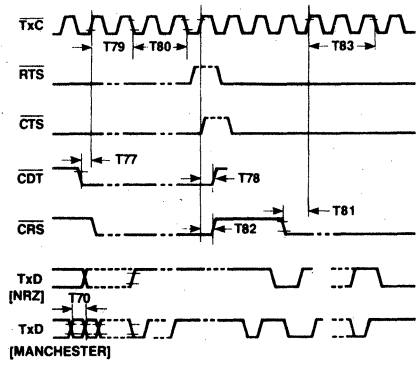
High Speed Mode

Symbol	Parameter	Min	Max	Units	Test Conditions
Transmit/Receive Clock Parameters					
T60	$\overline{RxC}$ $\overline{TxC}$ Cycle	200	*13	ns	
T61	$\overline{TxC}$ Rise Time		10	ns	*1
T62	$\overline{TxC}$ Fall Time		10	ns	*1
T63	$\overline{TxC}$ High	80	1000	ns	*1, *3
T64	$\overline{TxC}$ Low	80		ns	*1, *3
Transmit Data Parameters					
T65	TxD Rise Time		20	ns	*4
T66	TxD Fall Time		20	ns	*4
T67	$\overline{TxC}$ Low to TxD Valid		60	ns	*4, *6
T68	$\overline{TxC}$ Low to TxD Transition		60	ns	*2, *4
T69	$\overline{TxC}$ High to TxD Transition		60	ns	*2, *4
T70	TxD Transition— Transition	70			*2, *4
T71	$\overline{TxC}$ Low to TxD High (At the Transmission End)		60	ns	*4
$\overline{RTS}$, $\overline{CTS}$ Parameters					
T72	$\overline{TxC}$, Low to $\overline{RTS}$ Low Time to Activate $\overline{RTS}$		60	ns	*5
T73	$\overline{CTS}$ Low to $\overline{TxC}$ Low $\overline{CTS}$ Setup Time	65		ns	
T74	$\overline{TxC}$ Low to $\overline{RTS}$ High		60	ns	*5
T75	$\overline{TxC}$ Low to $\overline{CTS}$ Invalid	20		ns	
T75A	$\overline{CTS}$ High to $\overline{TxC}$ Low $\overline{CTS}$ Set-up Time to Stop Transmission	65		ns	*7
Interframe Spacing Parameters					
T76	Inter Frame Delay	*9			
$\overline{CRS}$, $\overline{CDT}$, Parameters					
T77	$\overline{CDT}$ Low to $\overline{TxC}$ High External Collision Detect Setup Time	45		ns	
T78	$\overline{TxC}$ High to $\overline{CDT}$ Inactive $\overline{CDT}$ Hold Time	20		ns	*14
T79	$\overline{CDT}$ Low to Jamming Start		*10		
T80	Jamming Period	*11			
T81	$\overline{CRS}$ Low to $\overline{TxC}$ High Carrier Sense Setup Time	45		ns	*14
T82	$\overline{TxC}$ High to $\overline{CRS}$ Inactive $\overline{CRS}$ Hold Time	20		ns	*14

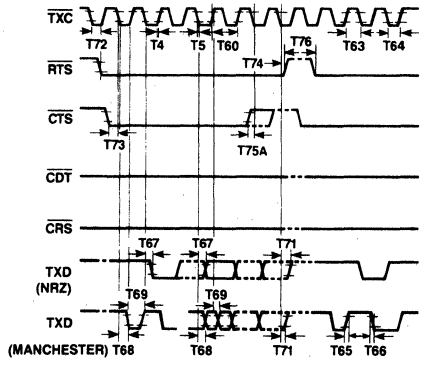
High Speed Mode (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
$\overline{CRS}$, $\overline{CDT}$, Parameters (Continued)					
T83	$\overline{CRS}$ High to Jaming (Internal Collision Detect)		*12		
T84	$\overline{CRS}$ High to $\overline{RxC}$ High. End of Receive Packet	80		ns	
T85	$\overline{RxC}$ High to $\overline{CRS}$ High. End of Receive Packet.	20		ns	
Receive Clock Parameters					
T86	$\overline{RxC}$ Rise Time		10	ns	*1
T87	$\overline{RxC}$ Fall Time		10	ns	*1
T88	$\overline{RxC}$ High Time	80		ns	*1
T89	$\overline{RxC}$ Low Time	80		ns	*1
Received Data Parameters					
T90	RxD Setup Time	45		ns	*1
T91	RxD Hold Time	45		ns	*1
T92	RxD Rise Time		20	ns	*1
T93	RxD Fall Time		20	ns	*1

NOTES:
*1 — MOS levels.
*2 — Manchester only.
*3 — Manchester. Needs 50% duty cycle.
*4 — 1 TTL load + 50 pF.
*5 — 1 TTL load + 100 pF.
*6 — NRZ only.
*7 — Abnormal end to transmissions: $\overline{CTS}$ expires before $\overline{RTS}$.
*8 — Normal end to transmission.
*9 — Programmable value.
T76 = NIFS × T60 (ns)
NIFS - the IFS configuration value.
If NIFS is less than 12, then NIFS is enforced to 12.
*10 — Programmable value:
T79 = NCDF × T60 + (12 to 15) × T60 (ns) (if collision occurs after preamble).
*11 — T80 = 32 × T60
*12 — Programmable value:
NCSF × TTRC + (12 to 15) × TTRC
T83 = NCSF × T60 + (12 to 15) × T60
NCDF - collision detect filter configuration value.
*13 — 2000 ns if configured for Manchester encoding.
*14 — To guarantee recognition on the next clock.

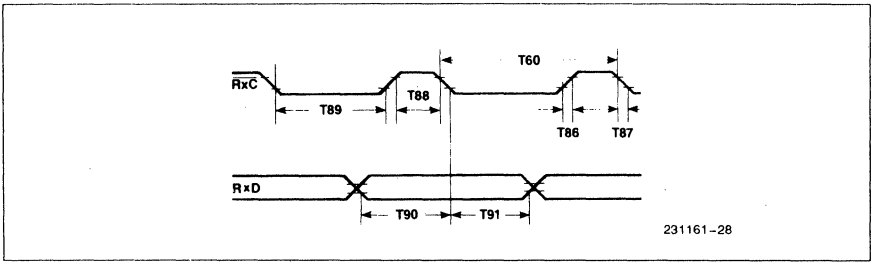


231161-26

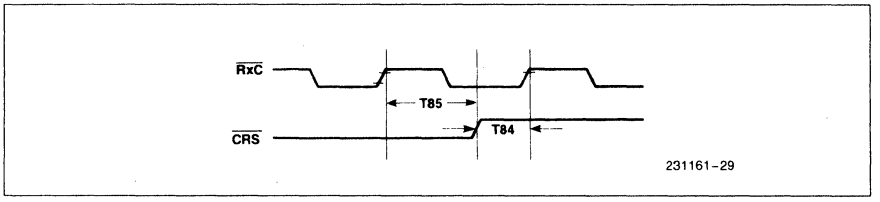


231161-27

Transmit Data Waveforms



Receive Data Waveforms (NRZ)



Receive Data Waveforms

Glossary

26LS30 A popular integrated circuit that converts a single-ended signal to a differential signal.

26LS32 A popular integrated circuit that converts a differential signal to a single-ended signal.

The 802 group The committees that are responsible for developing LAN protocols. The 802 group is divided into eight committees; each is responsible for dealing with a different LAN subject.

Access method A set of rules which describe the actions that a station must follow prior to transmitting its data onto the network cable. The access method also specifies the actions that a station should take upon discovering collisions on the network cable.

Analog front end An analog circuitry responsible for converting a digital signal to an analog signal prior to the transmission of the signal into the LAN cable. The circuit also accepts analog signals from the LAN cables and converts them to digital signals.

Application layer This is layer number 7 in the ISO seven layers model.

Asynchronous transmission A transmission where each character is preceded by a special bit called the start bit and followed by a special bit(s) called the stop bit(s).

Attenuation A process whereby a signal loses its quality due to the accumulative capacitance, resistance, and inductance of the cable. The attenuation is measured in decibels per unit length. As an example, a telephone twisted pair wire of 24 gauge has attenuation of 42.55 dB/mi at 1 MHz.

AUI Attachment unit interface. The transceiver cable in 10BASE5 (Ethernet) is also called AUI cable.

Balanced elliptical filter A type of filter utilized in front end circuits. As an example, the 10BASE-T filter is a seven-pole balanced elliptical filter with a 3-dB frequency between 17 and 19 MHz.

Baseband transmission In a baseband transmission the entire bandwidth of the network cable is dedicated for a single transmission.

Baud rate The rate at which data are transmitted serially. When using the telephone network for serial transmission, the common baud rates are 110, 300, 600, 1299, 2400, 4000, and 9600 baud.

BNC connector An off-the-shelf connector that provides the mechanical and electrical connection between coaxial cables.

Break character A special character used in asynchronous protocols. A break character is all 0s. It is used to attract the immediate attention of the receiver.

Broadcasting A feature in LANs that allows a station to transmit a message to all the stations in the system.

Bundle (twisted pairs) The unshielded twisted pairs of wires are usually purchased in bundles. Each bundle is composed of typically 25 or 50 pairs of unshielded pairs of twisted wire.

Bus topology A popular LAN topology, where one main cable is distributed among all the LAN's stations.

Cable capacitance corruption A process whereby a signal loses its quality during its transmission due to the accumulative capacitance of the cable. The cable capacitance is measured in farads per unit length. As an example, a telephone twisted pair wire of 24 gauge has a cable capacitance of 0.1 μF/mi.

Cable dc resistance corruption A process whereby a signal loses its quality during its transmission due to the accumulative resistance of the cable. The cable resistance is measured in ohms per unit length. As an example, a telephone twisted pair wire of 24 gauge has a cable dc resistance of 823.68 Ω/mi.

Cable inductance corruption A process whereby a signal loses its quality during its transmission due to the accumulative inductance of the cable. The cable inductance is measured in henries per unit length. As an example, a telephone twisted pair wire of 24 gauge has a cable inductance of 0.084 mH/mi.

CD Collision detect. A method used in the CSMA/CD access method where each station has the means to detect a collision.

Central controller A special LAN circuitry responsible for controlling and managing transmissions to the network cable. In a LAN that employs a central controller, a station is allowed to transmit only if it received direction to do so from the central controller. The station then transmits for the amount of time dictated by the controller.

Clock recovery A method of extracting the clock out of an incoming stream of bits. The clock recovery process is relatively easy to accomplish when the data stream is in a Manchester format.

Coaxial cable A well shielded cable designed to carry data at a rate of 10 Mbit/s, for distances ranging from 200 to 500 m without distorting the data. Coaxial cables are available as single-shielded or double-shielded cables. High-quality coaxial cables are expensive.

Collision A situation where two or more stations are transmitting at the same time.

Collision signal The collision signal is a special signal generated upon discovering that a collision exists in the network. This signal enables receiving stations to conclude that there is a collision in the system.

CRC The CRC field is a field in the 802.3 frame. It contains the result of a mathematical manipulation performed on the rest of the bytes in the frame.

Upon receiving a frame, the receiving node performs the same mathematical manipulations and compares the results to the transmitted CRC. If the CRC calculated is equal to the transmitted CRC, the receiving node concludes that the frame was received without transmission errors.

Cross talk A distortion phenomenon that occurs on a signal during its transmission over a cable due to nearby switching equipment and other nearby cables.

CS Carrier sense. A method used in the CSMA/CD access method where a station must sense a carrier on the network cable prior to transmission.

CSMA/CD access method Carrier sense multiple access with collision detection. The access method utilized by the 802.3 LANs.

CSMA/CA access method Carrier sense multiple access with collision avoidance. This access method utilizes carrier sense and multiple access, but unlike the CSMA/CD, it is designed with a collision avoidance mechanism.

Data communication network A collection of hardware components (boards, cables, connectors, etc.) and software components, which together allow the communication of data among several users. If the network cables are limited to relatively short distances, the network is called local area network.

Data link layer This is layer 2 in the ISO seven layers model.

Data packet A frame (or collection of frames) in 802.3 LAN is sometimes referred to as a data packet.

Data terminal equipment (DTE) A device (usually consists of a keyboard and monitor display) that allows the user to transmit and receive data from the LAN.

DC-to-dc converter (isolated) An electronic circuit that accepts a dc voltage at its input and outputs a dc voltage at its output. A dc-to-dc converter is used in the implementation of Ethernet and Cheapernet node boards. An isolated dc-to-dc converter is one that provides dc isolation between its input and output ports.

Deferring procedure The method used by a station to defer further transmission upon discovering a collision.

Destination address A field in the 802.3 frame. This field contains the destination address of the frame (6 bytes).

Deterministic LAN A LAN where the access method is such that each station is guaranteed a transmission permission at a predetermined time for a predetermined amount of time. The token ring family of LANs is an example of a deterministic LAN.

DIW D inside wiring. The telephone bundle that contains 25 or 50 pairs of unshielded twisted wires (bundles) is also called DIW.

DMA Direct memory access. A common feature of microprocessors, which allows fast transfers of data blocks between peripherals.

DPLL Digital phase lock loop. Circuitry that enables the extraction of the clock from the incoming stream of bits.

EMI Electromagnetic interference. The electronic noise generated by a current-carrying conductor.

Fiber optics Communication cables that are used for the transmission of data at a very high data rate. Unlike the regular metallic cables, the fiber cables do not carry electrical current. They transmit the data in a form of light. This type of transmission is not influenced by electrical or magnetic noise.

FIFO buffer First in, first out buffer. A memory buffer that stores incoming bytes. Upon issuing a request to the FIFO, the FIFO outputs the bytes in the same order that it received them. (The first byte to enter the FIFO is the first byte to leave the FIFO.)

File server A station in a LAN that is responsible for booting up and managing the operation for the rest of the stations in the network. The file server usually includes a large hard drive which holds application programs and databases.

FILO buffer First in, last out buffer. A memory buffer that stores incoming bytes. Upon issuing a request to the FILO, the FILO outputs the bytes in the reversed order that it received them. (The first byte to enter the FILO is the last byte to leave the FILO.)

Frame A basic unit of information in LANs. Data are sent in frames, and the LAN protocol dictates the length of the frame and the description and length of each of the fields in the frame.

Frequency division modulation (FDM) A technique used by some LANs where the network cable carries several transmission messages simultaneously. In this method, each transmission is assigned a different frequency channel.

Full-duplex communication A communication session in which transmitting and receiving is achieved simultaneously.

HHUB Header HUB. The HUB board located at the top level of a Starlan network is called a header hub.

Hub board A hub board is an electronic circuit board located at the center of the "star" in the 1BASE5 LAN topology. The board examines the transmissions from all the stations that are connected to its inputs and acts accordingly. If the hub discovers a single valid incoming signal at its inputs, it retransmits the signal to its output. If the hub discovers illegal transmissions at its output, it outputs the collision signal.

IBM bi-SYNC A synchronous communication protocol that requires two SYNC characters.

Idle signal The idle signal is a signal generated by 802.3 nodes at the end of the transmission.

IEEE IEEE is the abbreviation of institute of electrical and electronic engineers. This organization took upon itself the tasks of developing and publishing standard protocols.

IEEE 802.3 standard protocol The IEEE 802.3 is a set of LAN protocols for the implementation of layer 1 and the MAC layer. The access method used in the 802.3 set of LANs is called the CSMA/CD access method.

IEEE 802.3 10BASE5 (Ethernet) A standard LAN protocol with the following main characteristics: (1) data rate = 10 Mbit/s, (2) baseband transmission, (3) maximum permitted length of network cable = 500 m.

IEEE 802.3 1BASE5 (Starlan) A standard LAN protocol with the following main characteristics: (1) data rate = 1 Mbit/s, (2) baseband transmission, (3) maximum permitted length of network cable = 500 m.

IEEE 802.3 10BASE2 (Cheapernet) A standard LAN protocol with the following main characteristics: (1) data rate = 10 Mbit/s, (2) baseband transmission, (3) maximum permitted length of network cable = 200 m.

IEEE 802.3 10BASE2 (broadband Ethernet) A standard LAN protocol with the following main characteristics: (1) data rate = 10 Mbit/s, (2) broadband transmission, (3) maximum permitted length of network cable = 3600 m.

IEEE 802.3 10BASET (twisted pair Ethernet) A standard LAN protocol with the following main characteristics: (1) data rate = 10 Mbit/s, (2) broadband transmission, (3) maximum permitted length of network cable = 100 m.

The IEEE 802.4 standard LAN protocol The IEEE 802.4 is a set of LAN protocols for the implementation of layer 1 and the MAC layer. The access method used in the 802.4 set of LANs is called the token bus access method.

The IEEE 802.5 standard LAN protocol The IEEE 802.5 is a set of LAN protocols for the implementation of layer 1 and the MAC layer. The access method used in the 802.5 set of LANs is called the token ring access method.

The IEEE 802.6 standard LAN protocol The IEEE 802.6 is a set of LAN protocols for the implementation of layer 1 and the MAC layer. This LAN is called metropolitan area network (MAN), and it utilizes fiber optics cables.

IHUB Intermediate hub. Any hub that is not located at the top level of a Starlan network is called an IHUB.

Information field A field in the 802.3 frame. This field contains the actual information (data). The 802.3 protocol specifies a minimum of 46 bytes and a maximum of 1500 bytes in the information field.

Interrupt signal A signal that is applied to microprocessors and coprocessor devices. Upon receiving the interrupt signal, the device aborts its current operation and starts executing a special routine. The interrupt concept is implemented in cases where a special condition occurs that requires the processor to take immediate action.

I/O slot (PC) A connector residing on the PC motherboard, which enables an external circuit to be plugged and interfaced with the motherboard. The I/O slot provides all the signals that are necessary to accomplish the interface (i.e., data bus, address bus, control signals). Most available commercial node boards are designed as an I/O card to be plugged into the I/O slot.

ISO ISO is the abbreviation of International Standards Organization, an organization that took upon itself the tasks of developing and publishing various standard communication protocols.

Jabbering procedure A procedure that disables a node from transmitting. The node is disabled upon discovering that the node is transmitting longer than a predetermined amount of time.

Jamming procedure A node executes the jamming procedure upon discovering that it is transmitting simultaneously with another node. After discovering the collision, the node continues to transmit a special jam signal to ensure that all stations in the LAN are aware of the collision.

LAN Local area network, a communication system that is spread over a local (relatively small) geographical location. The network cables of the LAN are dedicated for the transmission of the LAN data and are not shared by other communication systems.

LAN controller An integrated circuit, designed specifically to implement a particular LAN protocol (e.g., the Intel's 82586, 82596 LAN controllers).

Length field A field in the 802.3 frame. This field contains the number of real data that the frame contains.

Line integrity test A feature implemented in the 10BASE-T repeaters and nodes. The line integrity pulses are periodic pulses transmitted by the node, which enables the repeater to conclude that a node is connected to its port.

Link segment The cable that connects a node to the MPR in a 10BASE-T LAN.

Loop-back mode A common technique used in communication for diagnostic purposes. The circuit is put in the loop-back mode by connecting the transmitter port of the circuit directly to the receiver port. The diagnostic software then causes the circuit to transmit and should observe that whatever was transmitted is received back.

MA Multiple access. A method used in the CSMA/CD access method, where a station is permitted to transmit one time after another (multiple access).

MAC layer The media access control (MAC) layer is the communication layer that is responsible for executing the access method of the LAN protocol.

Manchester code A special method of transmitting serial data. This method enables the receiver to detect a "bad" transmission by sensing a Manchester code violation in the incoming stream of bits. This method also enables the receiver to extract the clock out of the incoming bits.

MAU Medium attachment unit. The transceiver circuit in 10BASE5 (Ethernet) is also called MAU.

MDI Medium dependent interface. The connector that connects the link segment to the MAU circuit. In 10BASE-T LAN, the MDI is the RJ-45 8-pin connector.

Modem An electronic device that accepts a serial digital signal and converts it to analog signal compatible with the telephone network. The modem also accepts analog signals from the telephone network and converts it to a serial digital signal.

Motherboard (PC) The main board of the PC. This board contains the microprocessor (8086, '286, '386, '486), the memory chips (RAM, ROM), I/O slots, and other circuitries.

MPR Multiport repeater in 10BASE-T LAN.

Multicasting A feature in LANs that allows a station to transmit a message to a group of stations in the system.

Multiport repeater An electronic circuit in 10BASE-T LAN. All stations in the network are connected directly to the multiport repeater board (star topology).

Network layer This is layer 3 in the ISO seven layers model.

Node board An electronic board used in LAN stations. A node board plugged into a PC enables the PC to communicate with the rest of the LAN.

Node driver A software executed by the PC that enables the I/O node card to interface with the PC operating system and the higher level software.

Optical LED A special light emitting diode (LED) which outputs pulses of light at high frequency into a fiber optics cable.

Optical transistor A "baseless" transistor, which is able to be turned on by applying a light pulse (not current) to its BASE. This type of transistor is used in fiber optics systems.

Physical layer The physical layer is layer 1 in the ISO seven layers model. This layer specifies the physical parameters of the network.

Preamble field A field in the 802.3 frame. This field enables the receiving station to clock itself (to extract the clock) out of the incoming stream of bits.

Predistortion circuitry An electronic circuitry (usually an integral part of the analog front end circuit), which is responsible for distorting the signals prior to the transmission into the network cable. The predistortion compensates for the distortion that the transmitted signal undergoes during the transmission. The predistortion circuitry is especially important in 10BASE-T equipment where the signal is transmitted over unshielded wires.

Presentation layer This is layer 6 in the ISO seven layers model.

Probabilistic LAN A LAN where the access method is such that there is no guarantee when a station is able to transmit data into the network cable. The 802.3 family of LANs is an example of a probabilistic LAN.

Promiscuous mode A node may be configured to work in a promiscuous mode. In this mode, the node receives all frames, regardless of the destination address.

Propagation delay The amount of time it takes for signal to propagate from the source station to the destination station.

RG58 cable A coaxial cable used as the network cable in Cheapernet LANs.

RJ45 A connector used in the 10BASE-T LANs. The RJ45 is an 8-pin connector. Only 4 out of the 8 pins are utilized in the 10BASE-T LAN.

RxD RxD is a common notation for the pin of an integrated circuit which accepts an incoming transmitted signal.

Session layer This is layer 5 in the ISO seven layers model.

Seven layers model A communication network model (adapted by the ISO). This is a recommended model to be used when developing and implementing

communication networks. The concept of the model is to divide the network into seven distinct section (layers).

Signal jittering The jittering process is the process in which the edges of the transmitted serial signal are shifted from their nominal position. The LAN protocol specifies the maximum amount of jittering that the signal may undergo.

Single-ended signal A digital signal where a ground voltage represents a logical 0, and a high voltage represents a logical 1.

Source address A field in the 802.3 frame. This field contains the source address (the address of the station that sent this frame).

Space division multiplexing An inefficient method of establishing communication among several users. In this method, each cable in the network is dedicated for carrying data from a single user to another. The number of cables that are required to establish communication between N users is $N*(N-1)/2$.

Squelch circuit A squelch circuit has the responsibility of digitally filtering out noise signals.

Standard communication protocol A published set of rules that dictates the way and means by which a communication session is established. Standard protocols enable one to achieve successful communication even if the pieces of equipment (hardware and software) are made by different vendors.

Star topology A popular LAN topology, in which each of the LAN's stations is connected to a focal point (the center of the star).

Start bit A bit used in asynchronous communication protocol. The bit precedes the transmission of a single character and is used as a start-of-character indication.

Start of frame (SOF) A field in the 802.3 frame, an indication to the receiving station that following this field is actual useful information.

Station A DTE (data terminal equipment) together with its node is called a Station.

Stop bit A bit used in asynchronous communication protocol. The bit follows the transmission of a single character and is used as an end-of-character indication.

Synchronous transmission A transmission in which each string of characters is preceded by a special synchronization character(s) called the SYNC character(s).

Thick wire LAN The Ethernet is sometimes called a thick wire LAN (due to the thick coaxial cable that is utilized in this LAN).

Thin wire LAN The Cheapernet is sometimes called a thin wire LAN (due to the thin coaxial cable that is utilized in this LAN).

Transceiver An electronic circuit that is responsible for both transmitting and receiving (TRANSmitter + reCEIVER).

Transport layer This is layer 4 in the ISO seven layers model.

TPE Twisted pair Ethernet.

TxD TxD is a common notation for the pin of an integrated circuit that outputs the signal to be transmitted.

USART Universal synchronous asynchronous receiver transmitter. A device that accepts a serial digital signal and converts it to a parallel format. The USART also accepts a parallel format and converts it to a serial digital signal. The conversion is accomplished in accordance with standard serial communication protocols.

Unshielded twisted pairs This is the type of wire used by telephone companies to install telephone services. These wires are very inexpensive. The unshielded twisted pairs are also used as the network cables in Starlan and 10BASE-T LANs.

WAN Wide area network. A network that allows communication over a wide geographical area. Usually a network is considered a WAN if it utilizes the telephone network.

Index

26LS30, 49
26LS32, 52
80386, 120
80486, 120
8188, 62
82503, 141
82504, 133
82596, 119

Access method, 5
Alignment error, 80
Analog front end, 130
Asynchronous transmission, 156
AUI cable, 94

Back-off algorithm, 26
Baseband transmission, 6
Bit-stuffing frame, 79
Broadcasting, 21

Cables, 7
Carrier sense, 29
Central controller, 6
Coaxial cable, 95
Collision, 5
CRC, 20
CRC field, 77
CSMA/CD, 17, 123

Decoder, 28
Deferring procedures, 5
Demodulation, 147
Differential signal, 48
DMA, 66
Downstream, 37
DTE, 9

Electronic transmission, 1
Encoder, 28

Fiber optics, 7
File server, 184
Filter topology, 135
Frame, 19
Frame transmittal, 75

Gateway, 187

Header hub, 40
HHUB, 40
Hub, 32

Idle signal generator, 50
IEEE 802, 14
IHUB, 40
Intelligent repeaters, 186
Interface board, 24
Intermediate HUB, 40
ISO mode, 9

Jam pattern, 25

Line coupling, 134
Line integrity test, 183
Link segment, 129
LLC, 13
Loop back, 110

MAC, 12
Manchester format, 18
MAU, 93
Maximum configuration, 127
Medium dependent interface, 128
Modem, 147
Modulation, 6, 147
MOS level, 72
MPR, 124
Multiport, 124

Node board, 8
NRZ, 62, 107

Overrun, 80

Physical layer, 12
Pointer field, 87
Preamble field, 77
Presence signal, 23

Promiscuous mode, 77
Propagation delay, 42

Repeater board, 124
RJ-45, 129
RS-422, 49

Shared memory, 111
Sharing peripherals, 1
Single-ended signal, 48
Space division multiplexing, 2
Squelching, 56
Standard protocols, 2
Starlan, 31
Start topology, 5
Station, 9
Synchronous transmission, 155

Telephone wiring, 42
Terminator, 93
Thin wire, 98
Topologies, 4
Transceiver, 93
Twisted pair, 123

Upstream, 37
USART, 148

xTYPEy, 16

Yellow cable, 98

ABOUT THE AUTHOR

Nathan Gurewich is a senior electronics design engineer for a research laboratory, Tego Technology, Inc. He is involved in the design and implementation of advanced microprocessor-based circuits for use in data communications, telecommunications, and voice-processing applications. He holds a BSEE from Hofstra University and an MSEE from Columbia University.

Communications

How to design cost-effective, sophisticated LAN systems with off-the-shelf components

COMMUNICATION SYSTEMS

PRACTICAL GUIDE TO INTEL'S CONNECTIVITY DESIGNS

This comprehensive guide to communications systems design includes practical procedures for designing cost-effective, sophisticated LAN systems using off-the-shelf Intel components. Both hardware and software issues for all major types of communications systems are covered, from Starlan, Ethernet, and Cheapernet, to the new 10 BASE-T (Twisted Pair Ethernet).

Each design chapter includes sufficient information, drawings, and detailed implementation procedures for you to design systems using off-the-shelf components such as the Intel 82586, 82588, and others. You'll also find a wealth of practical LAN design know-how, including detailed step-by-step design illustrations for the implementation of node boards, workstations, repeaters, multiport repeaters, adapters, and more.

But you get more than practical professional working know-how—this invaluable guide also explains why things are done and suggests alternate solutions. You'll investigate the theory and operation principles of the popular 802.3 LAN standard, and gain a thorough understanding of the issues required to master this important technology.

About the Author

Nathan Gurewich is a senior electronics design engineer for a research laboratory, Tego Technology Inc. He is involved in the design and implementation of advanced microprocessor-based circuits for use in data communications, telecommunications, and voice processing applications. He holds a BSEE from Hofstra University and an MSEE from Columbia University.

Cover Design: Lerner Communications, Inc.
Cover Photo: © Bill Santos 1989

ISBN 0-07-025234-3

90000

9 780070 252349

McGraw-Hill, Inc.
Serving the Need for Knowledge
1221 Avenue of the Americas
New York, NY 10020