INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of "sectioning" the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.

University Microfilms International
300 N. Zeeb Road
Ann Arbor, MI 48106

Amirfaiz, Farhad

DESIGN OF A COMMUNICATION PORT AND IMPLEMENTATION OF A RECONFIGURABLE MEDIA TRANSLATION GATEWAY

The University of Arizona M.S. 1986

DESIGN OF A COMMUNICATION PORT AND IMPLEMENTATION OF A RECONFIGURABLE MEDIA TRANSLATION GATEWAY

by

Farhad Amirfaiz

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE
WITH A MAJOR IN ELECTRICAL ENGINEERING

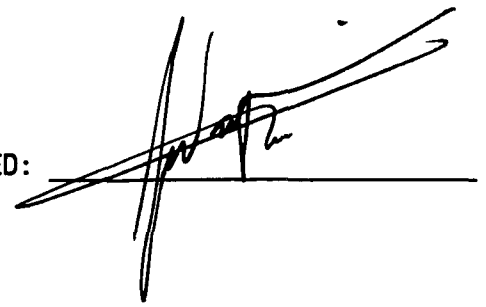
In the Graduate College

1 9 8 6

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of the requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the copyright holder.

SIGNED: 

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Ralph Martinez
Professor of Electrical Engineering

6 May 86
Date

ACKNOWLEDGEMENTS

This work was made possible with the help of IBM Tucson and The University of Arizona.

Special thanks to Tom Williams of IBM, and Dr. Martinez of the Department of Electrical and Computer Engineering of the University for their help, support, and guidance.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT

The objective of this work is to take a step towards developing a universal link between different computer systems and computer networks. "A Communication Port" is an information pipeline for data transfer between incompatible equipments and networks. The three card design discussed here is capable of interconnecting two incompatible systems, two networks, or a system and a network. The port is a generic, high performance, and reconfigurable low level link. One of the many configurations of the design, Implementation of a media translation gateway connecting Ethernet and IEEE488 networks, is given. The design is flexible enough to accept two available (IBM PC) networking cards.

CHAPTER 1

INTRODUCTION

Included in the introduction is discussion of the links leading to the choices that should be made in selecting a link and why a universal link would be desirable. This section ends with a few words about the low level link developed. The remainder of the paper covers the general design of the link and the details of a gateway developed based on this design.

Communication Links.

Communication links satisfy the need for sharing resources like programs, data, and processing power. Computer systems and networks are build around a communication link - a bus or a network. A link may also be a bridge or a gateway.

Buses. In a single board processor, the system is built around a bus. Buses can also serve as links between a processor and its peripherals in a single work station, or between similar processors in different work stations. Buses are generally short in length. Buses may be serial or parallel, unidirectional or bidirectional. Examples are Unibus, Multibus, STD bus, and VME bus. Buses are specific to the needs of the workstation and the users.

Networks. Networks are links between two or more processors or intelligent peripherals in the same work station, or between workstations located in the same or in different locations. There is a variety of networks in the government and the industry. Networks may be base band or broad band, local or long haul. Still, the exact design and implementation of a network is unique to a supplier. Examples are System Network Architecture (SNA) of IBM, Ethernet of Intel Digital and Xerox, General Purpose Interface Bus (GPIB) of Hewlett Packard, and Defense data network ARPANET. Networks are from a few meters to few kilometers with different throughputs, protocols, and topologies to serve the cost/performance needs of a specific section of the market. A discussion of different networks is given in Computer Networks (Tanenbaum 1981). Also see overview of Ethernet and IEEE488 as examples. Different implementation of the same network is often incompatible. For example 3COM implementation of Ethernet is not exactly interchangeable with the Digital implementation. The need for interconnecting systems and networks is satisfied by bridges, network interfaces, and gateways. These components are described here.

Bridges. Bridges are links between two system buses, or two networks of the same type. The purpose of the bridge is to allow heterogeneous processors to share their resources, send and receive messages or to synchronize their operation. Shared memory is one way of providing a processor (system) with access to the memory of another for program, data or peripheral sharing, and for sending and receiving status or commands. This shared memory may be physically in one or the other system or it may be part of the bridge. The processors may have different

architectures, word sizes, and bus protocols. This translation is the responsibility of the bridge.

Bridges between two system buses are not very common. They are most effective when the systems are very closely coupled and where a network can not provide the high throughput desired. Very few bridges were ever marketed to a larger customer base. Most products serve internal needs and are highly specialized. In local area networking, bridges allow connection between similar networks. This may be desirable in extending the local networks coverage in terms of number of users or the geographic span.

Network Interface Modules (NIM). In some literature called a Bus Interface Unit (BIU) for broadband networks, or Interface Message Processor (IMP) for ARPANET. A NIM is a link between a processor and a network and it consists of the hardware and software that enables a data device (system), or a workstation to exist on a network. NIMs provide the functions of the lower three, up to all seven, layers of the OSI model (International Organization for Standardization, ISO, Reference Model of Open System Interconnection, June, 1979), Figure 1.
The services of the lower layers are provided in hardware, while the upper layers are supported in software. The NIMs available in the market are specifically designed for a unique processor and network, and the software of the higher levels is normally supplied by the user. Different implementations of NIMs range from a single card residing in a system slot to a dedicated front end processors that interacts with the system through shared memory or system bus. The software may reside in the main processor, in both the main and the NIM processors, or in both.

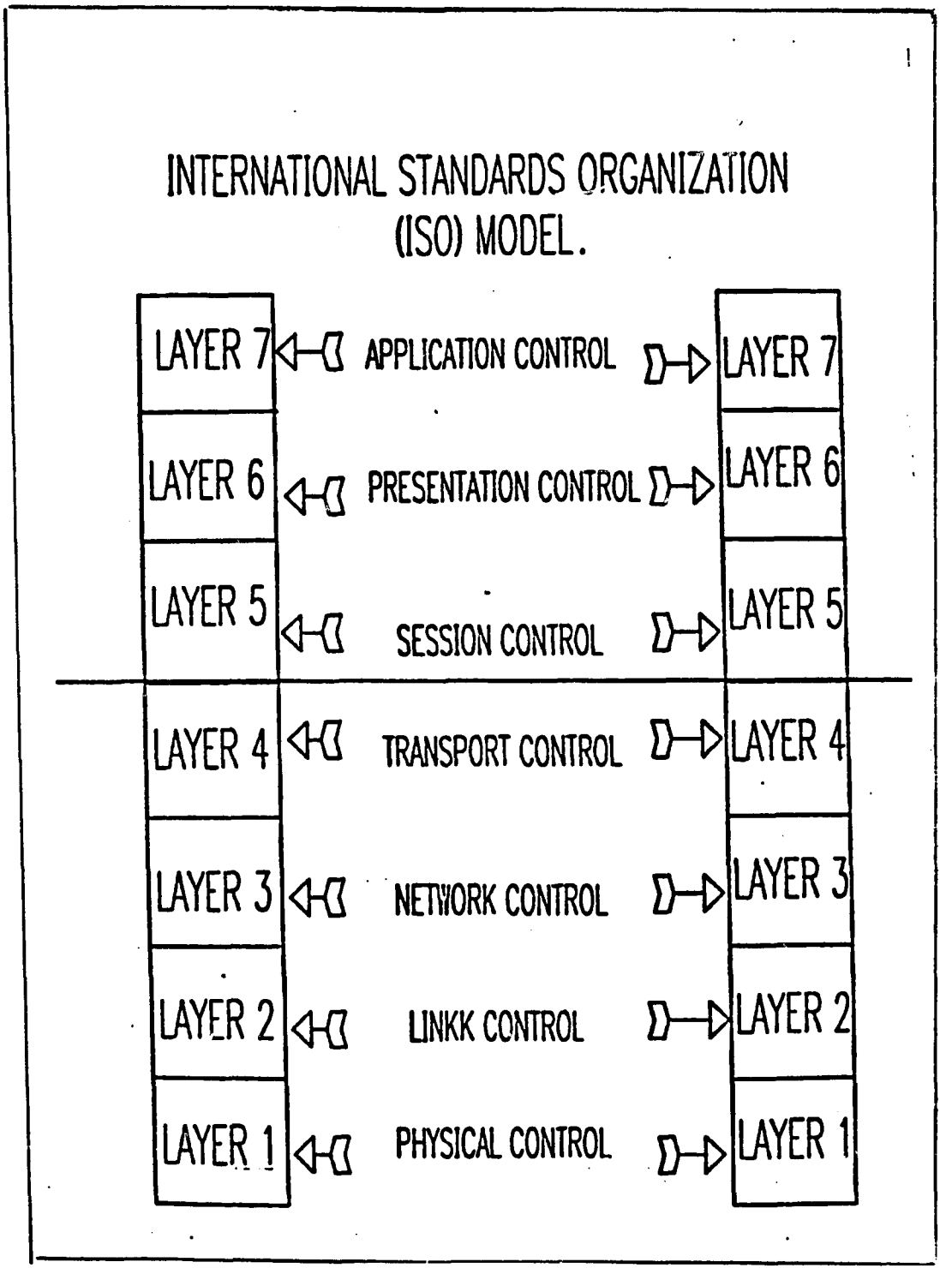


Figure 1. Open System Model of Networks.

Gateways. Gateways are communication links between two networks. To each of the two network the gateway appears as a node. In the sense of the ISO model, media translation gateways implement the lower two layers of each network (physical and data link layers) and the network layer of the gateway. Protocol translation gateways implement the lower three layers of the two networks plus the transport layer. Much has been written on the concept of gateways. Sapronov 1984, discusses the basic concept of gateways, the specifics of some example gateways, and the problems associated with interconnecting incompatible networks and concludes that "Gateways should be adaptable, endurable and compliant with national and international standards". The gateways reported in the literature are specific to the two networks they interconnect and their design and implementation is closely tied to the services that an environment required. Gateway research in the Computer Engineering Research Laboratory of University of Arizona is summarized by (Martinez, 1986).

The Choice of Communication Links

From this brief review of the communication links, it is clear that some complicated choices must be made in selecting one or more of the links that collectively meet a user's requirements. In other than trivial cases, this requires careful planning and a thorough study of the current and future needs. There is still a risk, that due to unforeseen needs, at

least a part of the link will have to be modified or abandoned sometime in the future. The aging of the technology is another factor to consider.

The choices are further complicated by a vendor's lack of support of facilities from the competition. Equipment, systems, and networks are significantly different in their characteristics. Often this is intended to protect market share and sales from the competitors. The choice of a link, once made, will usually constrain future options. This may later mean, either accepting limited choices, or scrapping the initial investment which may be business wise or otherwise impossible.

One Step Toward a Solution

Take a case where a large data base is to be made available to a number of stations from different vendors. The data base may completely reside on a host, or must be loaded up to a server host from the stations or an external source, or the data may be fragmented across a number of hosts and stations. Each work station may be have in turn different hardware architectures and capabilities. Some of the stations may reside on an existing local net or have the potential for networking. Others may have no networking facilities. Equipments in a single station may be from a single source or the station may be built around equipment from multiple sources. There is simply no single source of support for all stations and all equipment in such an environment. Many special purpose links must be purchased or developed to accommodate all stations and their needs or some hard choices must be made in re structuring and replacing some stations.

Aside from the development and/or purchase cost of the special hardware, there are other contractual, legal and maintenance problems. Generally, suppliers will not contract a link to another supplier's products. Third parties can often deliver a good technical solution, but maintenance of the hardware is still an issue, so is the ramifications of foreign components to the maintenance contract. As the products are replaced by newer offerings, the composition of the stations changes. The workloads of a station may also grow significantly different from the plan. This forces additional changes in the configurations of the work stations and the links required and the same problems are re-visited.

In looking for a single solution to interconnecting different equipment, workstations, and networks A Communication Port (ACP)was designed. The concept is to develop a generic physical link that can support virtually every network and system available. One that is competitive in cost to special purpose gateways or bridges but can be completely reconfigured at a fraction of the original investment to meet unforeseen needs. This solution opens some interesting possibilities:

1. The physical incompatibilities associated with mixing different equipment and services is overcome. The differences in the signals, protocols and hardware of the systems and stations can be ignored and a basic dialogue is possible throughout the environment regardless of the specifics of each station. This also allows for more freedom in selecting new equipment that best serves the changing needs of the business.

2. Changes are easier and cheaper to implement. Links can be reconfigured, removed, replaced or shared at a lower cost. Capabilities and loads of a station can be increased, decreased or off loaded to another station at will. Even if all links should be reconfigured, the initial investment is is protected.

3. Uniformity of the links increases productivity and reduces problems since only a single component is supported and maintained.

This does not mean, however, that the problem is completely solved. The fact that two parties speak the same language, does not mean that their philosophical differences is overcome. Even though data is transferred from a host, station, or network to another, the question of what this data means once it is received is not completely addressed here. ACP provides a frame work for standardizing higher level functions of incompatible systems by introducing uniformity and equality in the lower layers.

Design Objectives

ACP was designed primarily to provide a very high performance physical link between networks or systems. The port adapts to the specifics of each environment at a competitive price and is reconfigured to a new link at a low cost. Transfer of data is possible between two systems, a system and a network or two networks.

Incremental hardware capability and ease of programming are two important factors in the design of ACP. The cost of the port is mainly a function of the performance required. In cases where investment has already been made in some hardware, the port should also take advantage of the the existing network interface cards. The port must be easy to trouble shoot and maintain, in house, and in the field. Throughputs up to 300 Mega bits per second is desirable.

These objectives are delivered through a flexible Data Flow and Control (DFC) card and a set of Network Interface (NI) cards.

CHAPTER 2

DESIGN OF A COMMUNICATION PORT

The objective of the design is to provide a flexible communication link for interconnecting two systems or networks. Performance and reconfigurability of the design are of primary importance. ACP, Figure 2, is such a link.

The port is composed of a Data Flow and Control (DFC) card, with two programmable ports, which interconnects two systems directly or two networks through the two Network Interface (NI) cards. The DFC card is a programmable bridge. In the case of a bus extender the micro code of the two ports cards are identical. The combination of DFC and two NI cards is a gateway. In the case of a repeater the two NI cards are identical. In general, any combination can be transformed to another by replacing one or both of the network interfaces and reprogramming the ports. These combinations are described below:

A Gateway. A gateway configuration uses three cards. One card in this example is the DFC card. The second is a NIM card for network "A". The third is a NIM for network "B". Two sets of Programmable Read Only Memory (PROM) specialize the ports of DFC card to the two network interfaces. The address of each NIM on networks is selected through the switches on the cards. At system power up, the gateway performs a self test. Suppose that a packet from a node on "A" is to be passed on to a node on "B" through the gateway. The network interface card for "A"

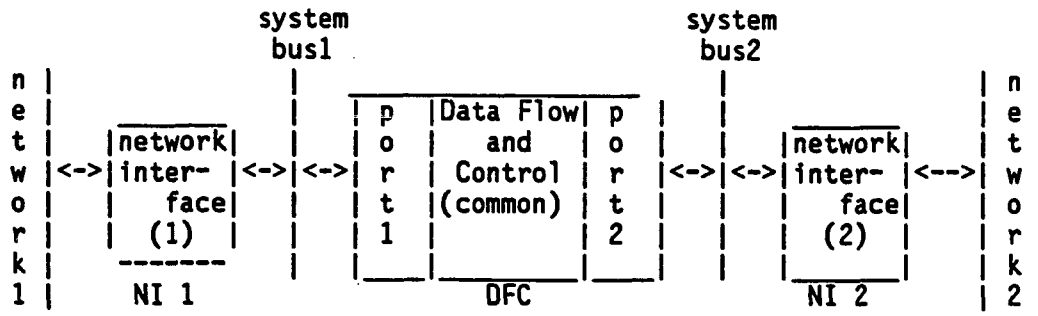


Figure 2. Model of A Communication Port (ACP)

receives the packet from network "A". The control fields of the packet are separated from the data, and passed by one port to the other. The data is transferred through the buffer section. Port2 sends the appropriate control information to the NIM for network "B" and drives the network interface card to assemble and transmit a packet on "B" as soon as the first word of data is in the buffer. The task under work is continuously displayed on the Light Emitting Diodes (LED).

A Bridge. The single DFC card provides a bridge between two system buses. The buffer of the DFC card functions as a shared memory on the two system buses and the two ports supervise DMA (Direct operations Memory Access) to and from the two hosts. The two sets of PROMs are replaced on the DFC card to specialize the ports to their tasks. The base address of the resident buffer is selected through the switches on the DFC card (device address of the ACP on the two channels). At system power up, the bridge performs a self test. Suppose that a file was to be transferred from the buffer of host "A" to the buffer of host "B". Host "A" enables the DFC card. This is done by simply writing to the device

address. One port starts the transfer of data to the buffer of DFC, while the other port initiates a DMA to the memory of the second host to move data at high speed to host "B"s buffer.

Two identical NI cards are used where the bridge connects two similar networks. This combination is a special case of a gateway described above, where network "A" and "B" are the same network.

A NIM. In order to configure a the DFC card is complimented with an NI card. In this configuration one port supervises DMA operations to the DFC card as a device on the bus, while the other port drives the NI card to send and receive from the network. The address of the node on network and the base address of "A"s buffer are selected through the switches on the cards.

Suppose that a packet from host "A" was to be sent on the network. The host enables the DFC card to transfer the buffer from its memory. Port1 supervised the DMA operation, while the second port drives the network interface to transmit the packet on network "B". As soon as the first word of data is available in the buffer, the NIM begins assembling and transmitting.

Regardless of the configuration of the link, in all cases a desirable link is one that:

1. Fits well within an environment and works well with the existing equipment.

2. Meets the technical requirements of the job competitively.

3. Is easy to customize, use, upgrade and modify.

4. Is simple to maintain in the field.

These features were also considered in the design.

Hardware Design Overview

The objective is to deliver a competitive and linear cost/performance curve in an easy to configure and simple to program system. Throughput of the system is a function of the word size and the clock rate and ranges from few hundred bits per second to about 320 Mega bits per second. Cost of the system is a function of the buffer depth, the throughput desired and the code development time. The number of configurations of the port is large since the DFC card can connect to a number of systems and accept a number of network interface cards. Other parameters that improve the design are modularity, testability, manufacturability , and maintainability. All components are visible from the connectors and a power on self test is standard. Operational errors are detected and displayed. Race conditions and critical timings are avoided, and availability and cost of the parts were considered. A tool is also discussed to expedite microcode development. Following is a list of the design's characteristics.

1. The DFC card features separate data and control paths, decentralized control, programmable interfaces with field replaceable customizing PROM, and switch selectable clock frequency and addressing. The buffer of the card is capable of 12 million words of continuous, full duplex, throughput and the size is expandable up to to 4 Mega bits. The ports are capable of 8 million instructions per second and function asynchronously. The buffer is designed to accept lower cost, slower parts where cost has priority over performance.

2. The network interface cards are simpler cards in design. They are composed of two to four Very Large Scale Integration (VLSI) parts and are fixed in function and relatively lower in cost. The major objective in the design of the network interface cards is the implementation of all physical and data layer functions of the network at a low cost.

Data Flow and Control Card (DFC)

This is the mother card, Figure 3, with two interfaces for NI type cards or direct connection to a system bus. DFC card is common to all configurations and must control a wide variety of network interface cards and interface to existing systems. Conceptually, the card is composed of two general purpose and reprogrammable ports that specialize to the network interfaces, and a shared buffer section.

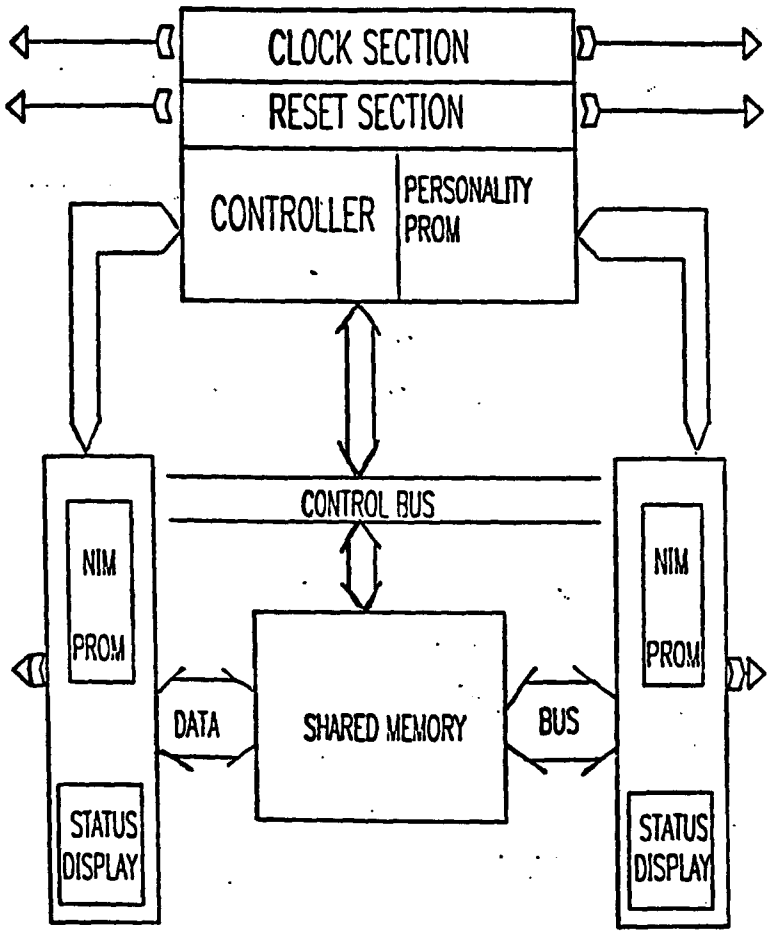


Figure 3. Functional block of the Data Flow and Control card.

A few design options were considered and these are summarized here. Microprocessors, bit slice processors, and finite state machines were considered for the design of the DFC card. The alternatives were measured against cost, performance, and flexibility. A sequencer is the best overall choice for the design at hand. Gate level logic, despite its speed, does not lend the flexibility required for a multipurpose design. Eight and sixteen bit microprocessors are the best choice for programmability. Many development systems are commercially available for these microprocessors that help verify the design quickly and program the application software efficiently. However they are not an attractive choice unless the throughput requirements of the design or cost objectives are relaxed.

The data and the control paths are separate in order to minimize speed dependencies between the two sides and to allow each to flow asynchronously. The control section was further decentralized into two interacting control units (ports). This allows each port to specialize to a channel interface card independently. The two units communicate through a local control bus.

The final design of the DFC card is composed of five functional elements and two local buses. The functional blocks are : two ports, a clock circuit, a reset circuit, and a buffer section. The two buses are data and control buses. The ports interact with the control portion of the signals from the system or network cards attached. They supervise the flow of the data from one channel to the other. Error handling and driving of the channel cards are the two other functions of the ports. The clock circuitry locally provides the clock pulses of the card and

clocks the network interfaces through the interfaces. The reset line is also available at the interface. The buffer section stores and forwards data in both directions. It can serve the two channels asynchronously and simultaneously.

Ports of the DFC. There are two ports on the DFC card as shown in Figure 4. Each port consists of a micro sequencer and micro program control store, a channel driver, a timer, a condition multiplexer and some latches, registers and LEDs. The micro code executed by the two may be different, since each port personalizes to the needs of one of the network interface cards. The function of the port are:

1. Interacting with a system bus or a network interface in its own language. This consist of physical and logical interfacing. The physical interface and the control signals are redefinable. The control sequences required interact with the system bus or network card is reprogrammable also. The signals are provided by the sequencer through a set of I/O lines that can be redefined. The control sequences are provided by the channel driver. This scheme is most flexible for accepting available network interface cards.

2. Control the data flow of the buffer. The read and write cycles of the buffer is supervised by the ports and the buffer conditions are monitored. The goal is to make the buffer appear as a wire connection between the two channels and as transparent as possible. Data markers are inserted and removed in the data by the ports to help in error detection and data blocking.

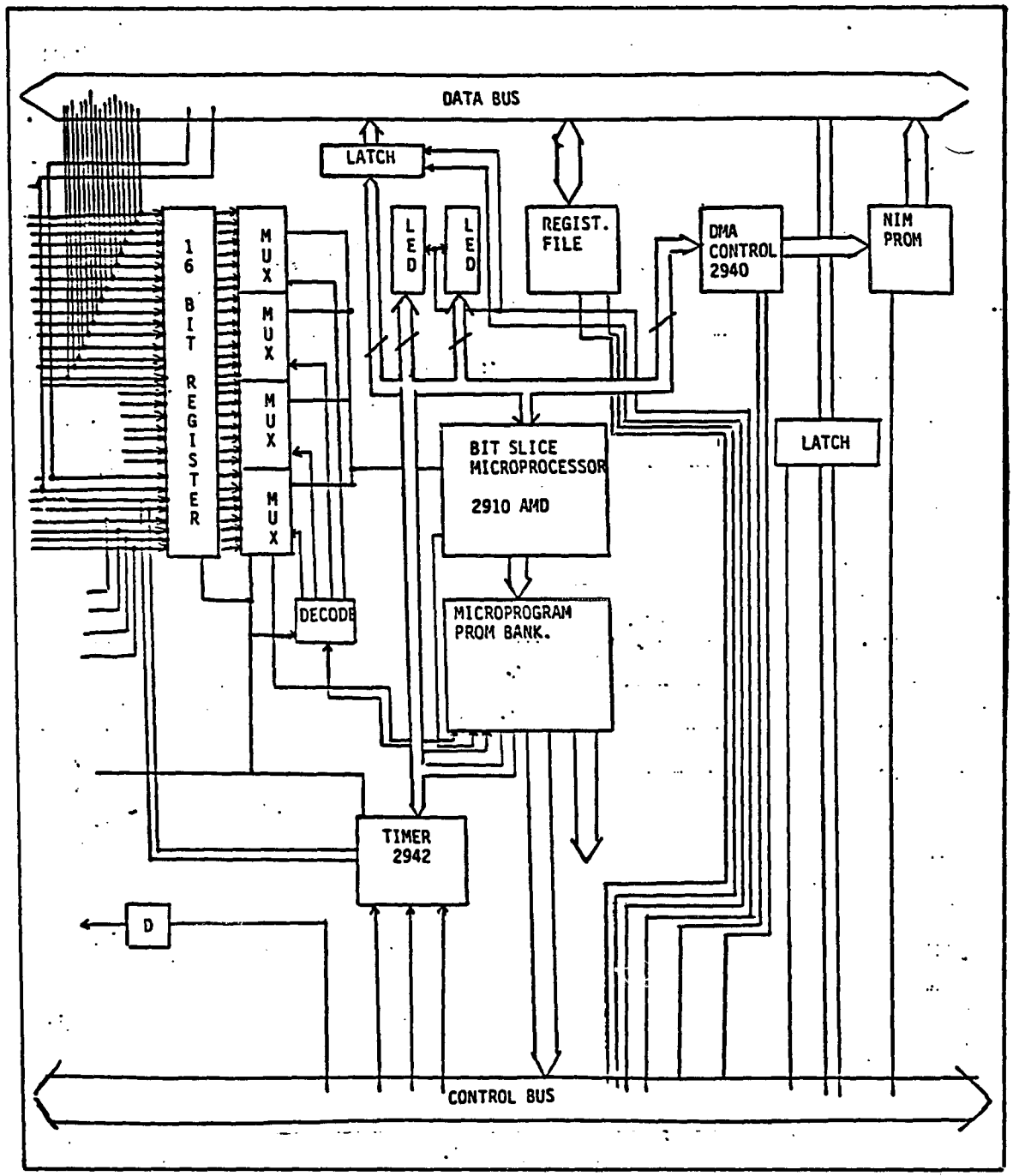


Figure 4. Design of the Ports of the DFC card.

3. Provide status to the host or net, and to the opposite port. Operation of the two ports is harmonized through the local control bus. Status is updated periodically for visual display and the port responds to queries from the host or the net.

4. Error handling. The port uses the timer and the interrupt lines to detect errors. The decision as to how an error should be handled is made by the port. This depends on the error, the source and which side owns the problem. The different options available are to take action, notify the opposite section and display and/or send status. Each port can reset the network interface and the DFC. The port can adapt to error detection and recovery needs of each case. The choice of recovery and the course of action is not fixed.

The main characteristics of the DFC card is described below:

Sequencer of the Port. The heart of a port is a bit slice sequencer that drives a micro program store. For a discussion on bit slice designs see (Donnamaie and White, 1981) and (Mick and Brick and Brick, 1980). In each cycle a control word is accessed from the control store and routed to different functional blocks of the port. This word is composed of commands, data and addresses. The sequencer generates an address every instruction cycle. The source of the address is the internal registers, internal counter, internal stack, or external inputs of the sequencer. This is the address of a control word in the control store. This scheme allow programming of the functions of the port, by simply replacing the control store. Conditional Branching is done through

a conditional multiplexer (MUX), which selects conditional input of interest as input to the sequencer. Some conditions are local others are external. This line is used to decide what the source of the next address should be. Part of the control word specifies the Instruction to the MUX, the type of branch, and the branch address. Other bits of the control word are commands to the timer, channel driver, and signals to the interface.

The Interface. This is the physical connection of the port to a channel interface card or a system bus. The interface is composed of the control signals from the port, the conditional inputs to the port, the data bus to the buffer and clock, reset and power lines. These lines can be redefined in firmware or jumpered to a different pin so that the interface can be specified as required for each application.

The timer. This device is a safe guard against the problems that the port may encounter. When a critical command is issued locally by the port, the timer is activated. If the command is not carried out in the time allowed, the port is notified. The timer is checked each time the system is powered and then used in normal operation to identify hardware problems and system hangs.

The channel driver. Provides, device specific, control sequences to the network interfaces or systems. This will off load the sequencer from the burden of both controlling the interface and executing driver routines. Like the sequencer, the channel driver has a micro control store and a counter that sweeps through the range of valid addresses. These control word are routines for the network interface cards or status

and responses that a host expects from an intelligent device existing on its system bus.

Shared Buffer of DFC. What is desirable is a buffer throughput that matches the requirements of an application. For this reason the buffer section accepts slower but pin compatible memory elements and the clock section is programmable. The two networks or systems may have significantly different data rates. The buffer can accept 8, 16 and 32 bit memory chips. Most available chips are pin compatible and the sockets allow the buffer to be up or down graded in width. Memory devices often have an extra bit for parity. For example 8 bits plus parity, a total of 9 bits. These bits can be used both for error correction and as data markers. The design given here is an 18 bit buffer. The two bits are used as data markers. The two network, systems, and the buffer of the DFC may have different word sizes. A funnel of registers is added between the buffer buffer and the interface to transform the word sizes. Among the options explored for the design of the buffer section, dual port RAMs (Random Access Memory) and FIFOs were considered. (First-In-First-Out memory). Dual port RAMs, are fast but expensive. They are not as dense and increase the physical size of the card. There are other disadvantages to dual port RAMs. They require address generation and control circuitry. If a large buffer is needed for an application, the card is over populated. Problems arise in managing the address control functions of a large buffer of this kind and the overhead becomes critical. Reliability and testability of the design may also suffer. FIFOs are the better choice. There is no random access capabilities in a FIFO architecture, however, the buffer can be sequential here. Therefore FIFOs work well.

FIFOs are cheaper and denser, there is no address generation overhead, and they are easily expandable. The control of such a buffer is considerably simpler. There are two main cases to monitor, full and empty. In comparison even though, FIFOs are two to three times slower than what could be achieved by RAMs, they are sufficient for the design. As the technology advances to make denser memory available that are pin and power compatible, the higher density elements can replace the existing ones. No alteration is needed to the design to accommodate this. The Buffer is shown in Figure 5.

Clock and Timing Section This section provides the clock pulses used throughout the system. There are two identical circuitries in the clock section each providing 8 clock pulses. Two Crystal oscillators generate the basic frequencies whose rate and duty cycle are both switch selectable through the clock chip. Thus, the section is capable of providing 16 different pulses whose frequencies and edges vary. Figure 6 is one of the two clock sections.

Reset Section This section provides a strobe used both on the DFC card and available at the interface for use by the network interface cards. A pulse is generated using an RC network and a gate at power on, at the request of a port, or manually from the system cover. This pulse is used through out the design for initialization. The section is composed of a timing element and some logic. The width of the pulse is adjustable through the capacitor and resistor combination.

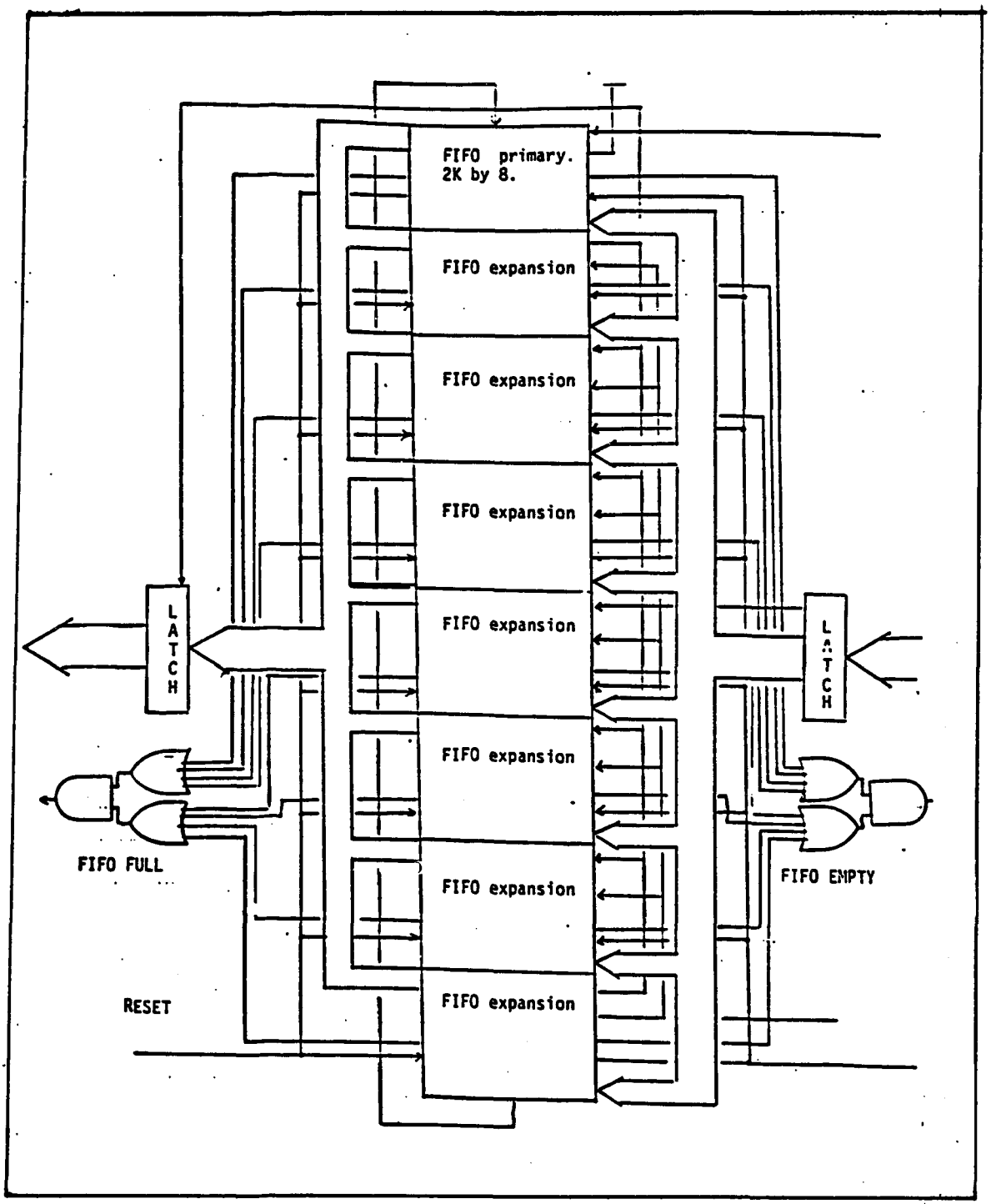


Figure 5. Design of the Buffer Section using FIFOs.

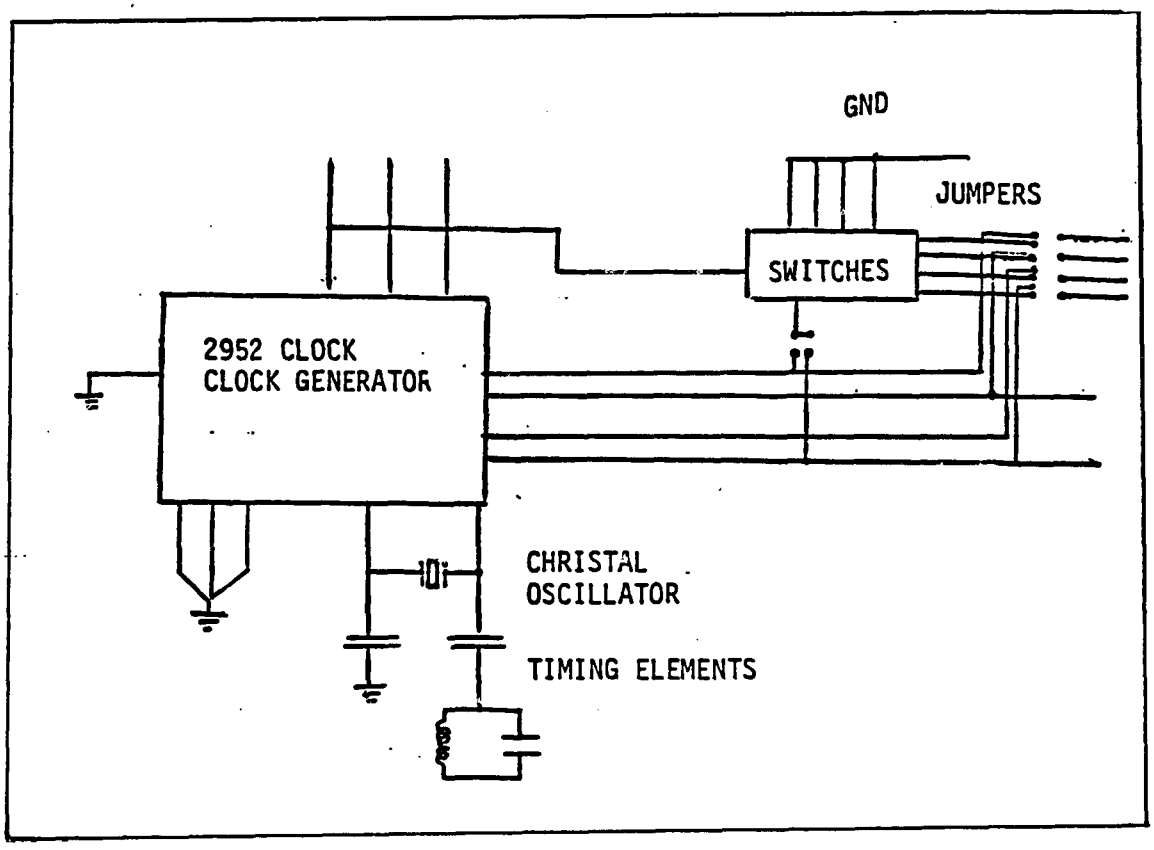


Figure 6. Clock Section of DFC.

Network Interface Card (NI).

Each slot of the DFC card can accept one of a number of different NI cards. These cards implement the physical and the data link layer functions of the two networks. These functions are data encoding and decoding, channel access, error control and packet assemble, disassemble, receive and transmit.

Just about every major component manufacturer has a set of VLSI parts for the popular networks which simplifies the design and development

task and many sources market network interface cards for most computer systems. A number of NI cards were designed, of which three are given in this section. The objective was to size the extend and the cost of the effort.

Firmware Design Overview

Design of the ACP allows for the system to be configured into a NIM, a bridge or a gateway by accepting different network interface cards including readily available hardware. The DFC card accepts a set of PROMs that contain application specific code. This code specializes each of the two ports. The hardware is fast and flexible and the ports are essentially two general purpose processors. The code defines the self test, the application, the type of services provided and the capability of accepting a NI card or existing on a system bus.

The purpose of the code is to: define the interface, driver routines, self test, error detection and recovery, flow control and the higher level functions of the port. The system or interface card expects a certain signalling scheme used in passing control and data words. The interface of the port is programmed to provide the proper handshaking. Initialize, send, receive and status commands are basic to all networking and DMA functions, but the command words and the driving sequences are different for every case.

Reprogramming. Only one of the ports may need to be reprogrammed. In cases where one of the networks or systems is changed, only the code

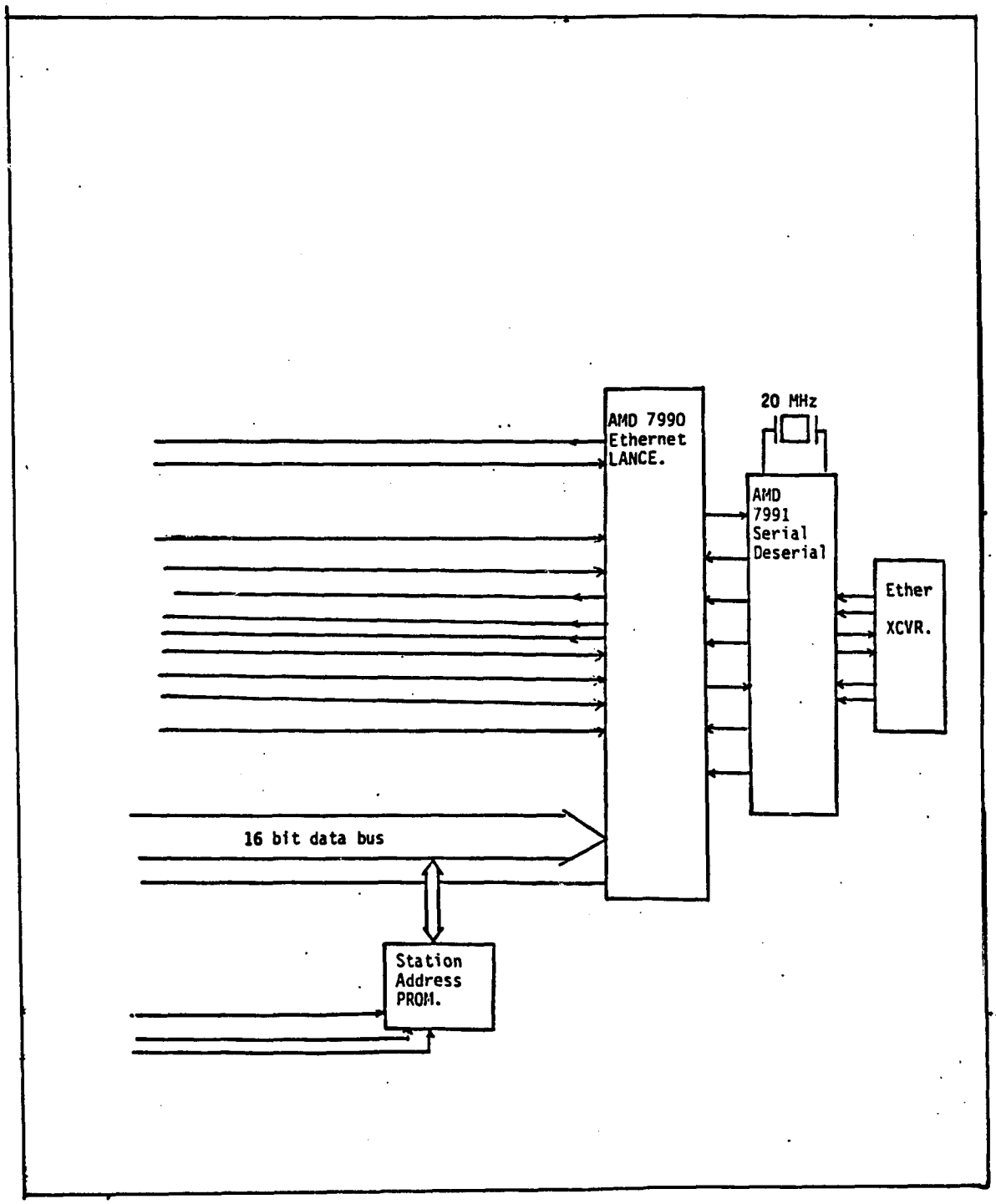


Figure 7. Ethernet NIM Design using AMD parts.

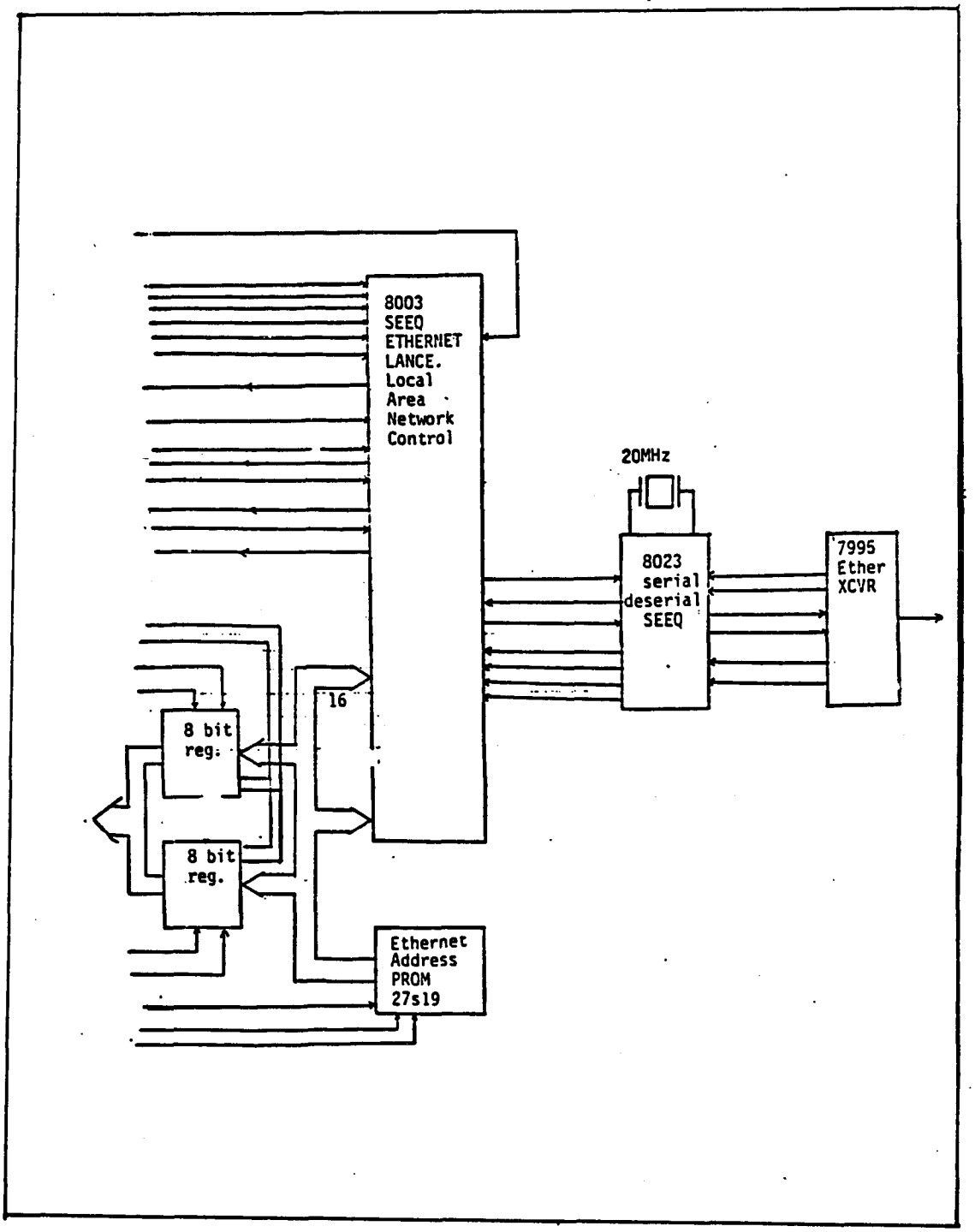


Figure 8. Ethernet NIM Design using SEEQ parts.

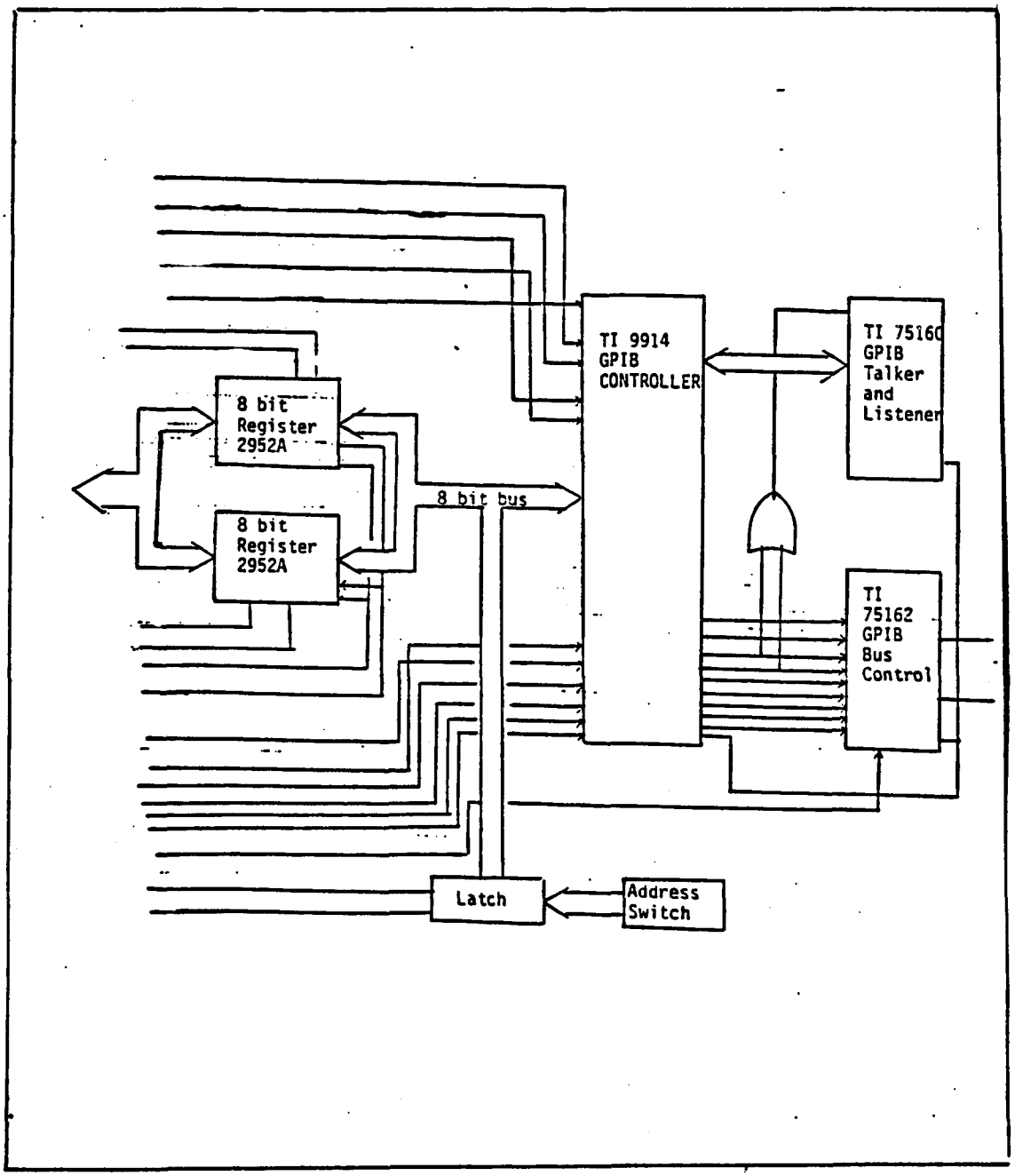


Figure 9. IEEE488 NIM Design using TI parts.

executed by one port needs to be replaced. Most popular systems can host a number of different network interface cards. For these cards, the physical interface (system slot) is the same but the driver routines are different. To take advantage of this, the code was logically divided into segments and physically separated in each port. The segment of code, responsible for self test, error detection and recovery and higher level functions of the port resides in the sequencer's control store. The segment that defines the details of the interaction of the port with the interface card or the bus resides in the channel driver's storage. This arrangement reduces the time and cost of developing code in some cases. In the gateway discussed here the two networking cards both plug into the same interface and they expect the same basic send, receive and status commands but the details of each command differ.

Still, another scheme is proposed to further simplify coding. If an available NI card is observed in its natural environment, the words and the timing gathered provides valuable insight into the channel driver's coding. This technique is simple and powerful for this purpose as well as its original application to comparison testing, (Penney 1979). Since code development time is a critical factor in the success of the design, a discussion of application of this technique included here under Micro Code Development.

Programming the Sequencer

It is assumed that the reader is familiar with bit slice processing. For a discussion on bit slice processing please see (Donnamaie and White, 1981) or (Mick and Brick, 1980). A typical format for the sequencer is given in Figure 10 and it is composed of three main records:

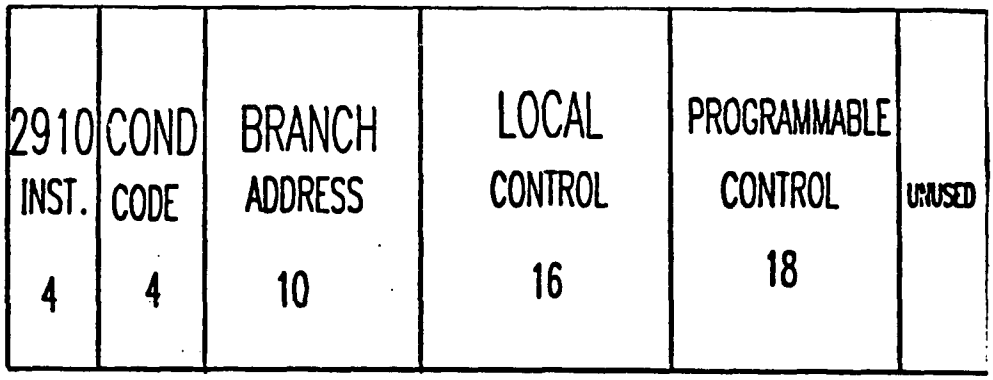


Figure 10. Micro command word format of the ports.

1. Those associated with the functions of the micro sequencer:

   a. Conditional MUX Select field which selects one of the conditional inputs to the port as input to the sequencer. 32 bits are available, 16 bits are predefined, fixed, and are internal to the DFC, while 16 bits are available for interrupt handling from the interface. The first group comes from the internal conditions, like timer 1 and 2 time out, opposite port failure, and buffer

full and empty. The second group from the interface to the NI card, to monitor send, receive, and error interrupts.

b. Micro Sequencer Command field which specifies the instruction of the sequencer.

c. Branch address field If the condition of a branch is true, the next instruction is accesses from this address.

2. Those associated with the different sections of the port - Local Control. This group includes:

a. Timer control. These lines program, start and stop the two timers.

b. Channel driver control. These program, enable and disable the channel controller.

c. Buffer control. These are the read and write lines of the buffer.

d. Data flow control bits are end of segment and error markers added in the buffer.

e. Status control. That drive the status LEDs, make the status available for to the NIM to transmit, and inform the other port of failure or success.

f. Internal file control lines for read and write to and from the internal register file. This file has status of the port and / or the control fields of the packets.

g. Recovery control. Reset request and failure / success acknowledge are in this group.

3. Those associated with the interface. These signals define the control portion of the interfaces to the NI cards.

Driver routines for the bus or network interface card reside in the channel driver's store. The routines are sequences of command words like initialize, send, receive and status read. As an example, the following command sequence may initialize a NI card :

Cycle	Command	Hex
1	Address the command register	31 Hex
2	Send command word	01 Hex
3	Address Device register	31 Hex
4	Send initialization command	FF Hex

Figure 11. Initialization of a NI card.

This set of commands is stored as a routine in the channel driver's store. Other routines are send, receive, and status.

Configurations of the Port

Given 10 NI cards and the code for interfacing to these 10 NI cards as well as to 10 system buses, the port can take 210 different configurations. 10 bus extenders, 10 network repeaters, 100 network interfaces, 45 bridges and 45 gateways. The DFC card is developed only once and the existing NI cards have no hardware development costs. The full 210 different configurations are supported with only 20 sets of PROMs.

CHAPTER 3

IMPLEMENTATION OF A MEDIA TRANSLATION GATEWAY

Of the different possible configurations of ACP, gateways are the more interesting class. This section covers the implementation of an Ethernet to IEEE488 media translation gateway, developed based on ACP's design.

This section summarizes some concepts in network interconnection and how the design would handle these issues. The summary is followed by a brief look at the Ethernet and the IEEE488 networks and the specifics of how the design may be utilized in a typical application.

There are a number of networks in the government and the industry. Examples are SNA, Ethernet, GPIB, ARPANET and DECnet. Not only the characteristics of these networks are different, also the exact design and implementation of a network is unique to a supplier. The ISO model in Figure 1 provides a general tool for the study and comparison of different networks. Networks differ in the layers implemented, the services provided in each layer of the model, and how they are implemented in hardware and software.

Gateways are special purpose processors that act as communication links between two networks. The gateways available in the market are specific to the two networks they interconnect, while the design given here is adaptable.

There are at least two types of gateways. Media Translation and Protocol Translation gateways, (Estrin 1982). Figure 12 and Figure 13 The media translation gateway implements the physical and data link layers of the two networks and the network layer functions of the gateway. The protocol translation gateway implements the lower three layers of the two networks as well as the protocol layer functions of the gateway. A media translation gateway is close to the ACP design.

Internetworking Philosophies

There are three main schools of thought in network interconnection. Translation, encapsulation and internet protocols. The features of each approach are presented here.

Translation. In this approach the packet is received from a net, disassembled, re-assembled into a packet of the second network and sent out. Ong, 1982, discusses this approach in length and gives a mathematical definition of protocol translation including a gateway based on translation. One of the advantage of translation is that bandwidth is not wasted on multiple packet headers as in encapsulation and the implementation of the two networks is not modified as in internet protocols. However, the disadvantages are the processing required at the gateway and the tables that must be maintained to help the translation algorithm. This approach is used in the Ethernet to 488 gateway since the assumption was made that the source and the destination nodes are the only

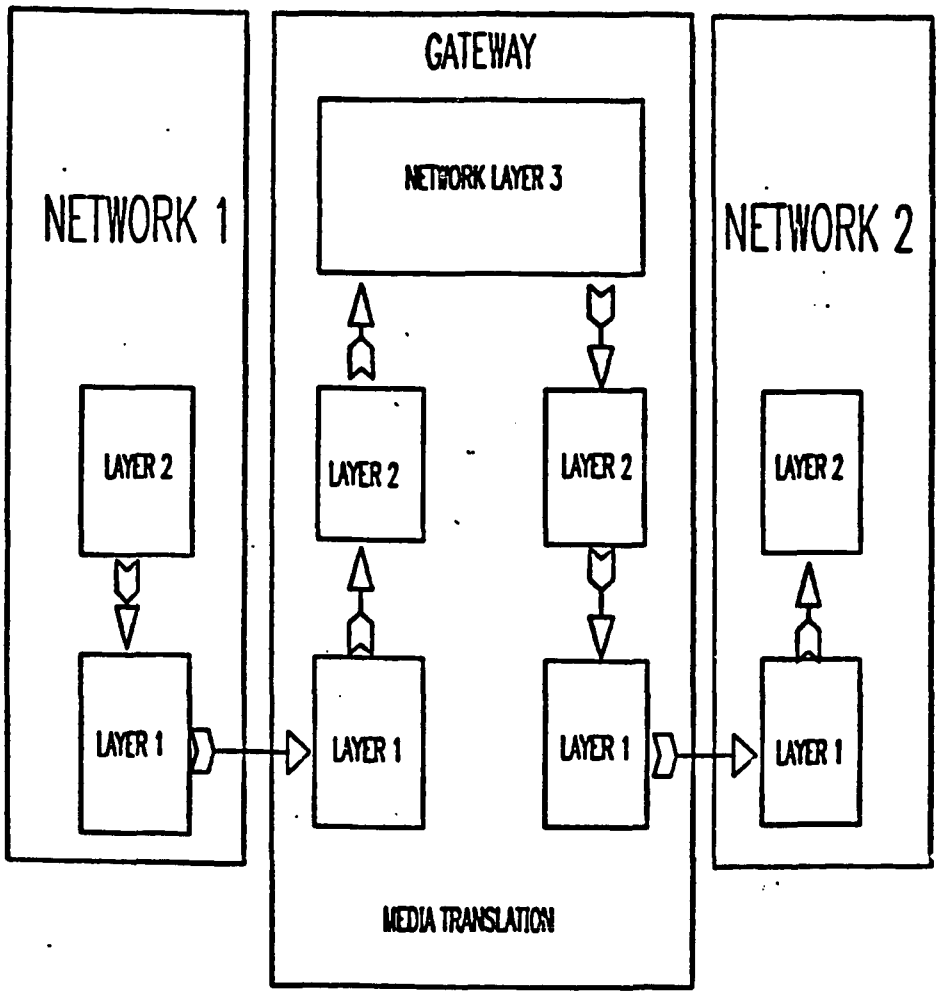


Figure 12. Media translation gateway.

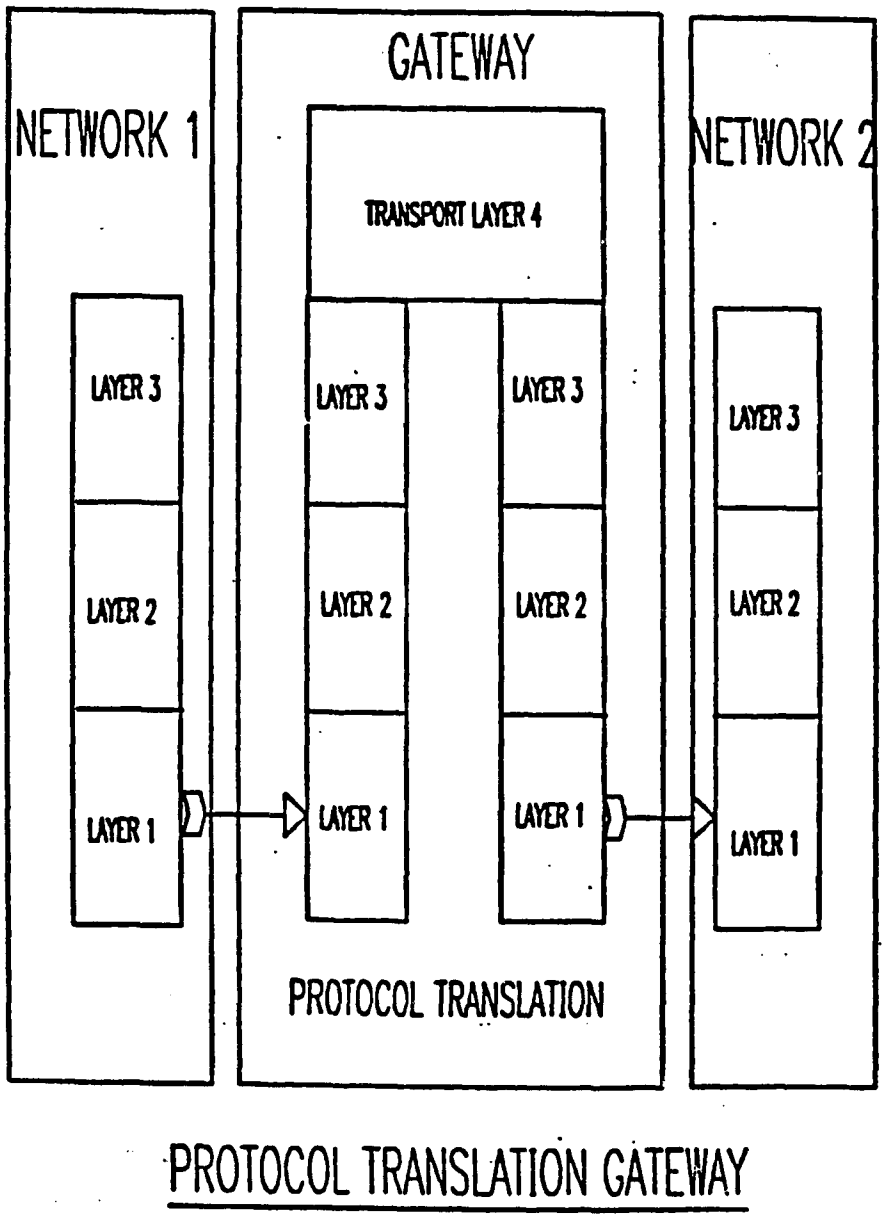


Figure 13. Protocol translation gateways.

nodes on each network and the requirement of transferring data at the highest possible rate.

Encapsulation. In this approach the data is encapsulated in the header fields of the first net, then this packet is encapsulated in the header fields of the second net, and so on. At the gateway the packet is received and the the fields associated with the first network is stripped. The data left is a packet ready to be sent on the second network. Postel discusses this approach in length. One of the advantage of encapsulation is that little processing is needed at the gateway versus protocol translation and the implementation of the two networks is not modified as in internet protocols. However, the disadvantage is the bandwidth wasted and the fact that the packet may get too large to fit in a frame of one or more networks, hence the need for fragmentation and blocking of the packet. This approach is used here for addressing.

Internetworking protocol. In this approach a new internet protocol is introduced and one or more layers of the two networks is modified such that there is a single format for the packets. Postel discusses this approach with examples. IP and UDP internet protocols are introduced, discussed and the complete implementation is given for the ARPANET. This approach conserves the bandwidth and avoids the processing required at the gateway. Perhaps this is the best technical solution. The drawback, if any, is business wise or political and stems from the facts that the layers of the two network must be modified, the modification must be acceptable, and the new protocol supported by all parties. This approach is not used here since different interest groups support the two networks.

Some internetworking issues. Routing, addressing, flow control, error detection and recovery, fragmentation and blocking, and format translation are issues involved in media translation. Shoch (1978) defines a name as "what an object is in the internet", an address as "where it is", and a route as "how to get to it". His paper clarifies these ideas and expands on their relation. For the case of the gateway developed, only how ACP provides the tools to deal with these issues is discussed here. The design of ACP handles these issues as follows:

Routing. Is the decision of which path is taken in getting a packet from the source to the destination. The design of ACP provides a register file in each port that can hold a routing table. The gateway maintains the table and it can be accessed by addressing the gateway from either of the two networks through a "command" packet. This is simply a packet with some bits set that the gateway recognizes. The choice of the bits is made when the ports are programmed. ACP can provide status of its operation, the load of the buffer and other network management information using this same mechanism. In short the design, as presented, can satisfy the more complex applications in a decentralized routing scheme. In the implementation given, however, the route is fixed and the address is encapsulated in the packet.

Addressing. Is handling the information imbedded in the packet that routes the data from the source to the destination. A frame in a network has a source address and destination address. In interconnecting two networks the first is the address of the sender and the second is the address of the gateway. Hence the need for an additional field that specifies the final destination address. Of the different options that

have been proposed, the idea of encapsulating the internet address is best suited here. A field was added to the frame for this purpose. The first "n" bytes of data is interpreted as the destination address by the gateway and a frame may contain a number of nested addresses. "n" is a function of the protocol of a network. For the Ethernet to IEEE488 gateway, "n" is six.

Flow control. Is managing the finite buffer of the gateway. Since the buffer is an order of magnitude faster than either of the two networks, there is never a case where the gateway is incapable of keeping up with the source. However, when the two nets have different throughputs, flow must be controlled by the gateway. A number of options are available. Obviously, one can drop a packet, if the receiving network can not keep up, and rely on the sender to retransmit because of no acknowledgement from the receiver. A second alternative is to provide status upon request to the sender. A third is to program the gateway to send a status frame to the sender when the buffer is getting full. ACP is flexible to function in either of the three modes. Also the on board buffer is expandable to about 5000 Ethernet frames waiting for transmission on the 488. Another common technique is also exploited: Normally, as soon as the first word is available in the buffer, transmission on the slower network can begin. On the other hand when passing frames from a slow network to a faster net, the gateway may tie up the transmit side of the faster NIM, while the slower NIM fills up the buffer word by word. Packets may be lost in some NIMs since the transmit circuit has tied resources that the receiver can not access. Thus a frame destined for the gateway on the faster network is lost. In such cases one

port notifies the other when the complete packet has been received and is available in the buffer. The Ethernet side is "end of packet" driven, while the 488 side is "buffer not empty" driven. In general, it may be desirable to signify end of packet, packet error, data error or other markups in the buffer as well as through an interrupt. Both options are available. The buffer is actually 18 bits wide and one of the two extra bits is used to signal end of packet in the data. An interrupt is also generated by the receiving side of the gateway to notify the sending side of "receive started" and "packet completed" conditions.

Error detection and recovery. Is handling the problems associated with packet sending and transmitting. Detection is done at one or more layers of the model, depending on the requirements of the design. The design can handle this task at all layers. In the lower two layers there are two modes of detection. Controller can read the status of a NIM, and time the operation that is being performed. Different actions are taken for recovery based on the severity of the problem and the type of error. For recovery at the higher levels the gateway can notify the sender of a fatal or non-fatal condition. The gateway has the capability to Reset itself.

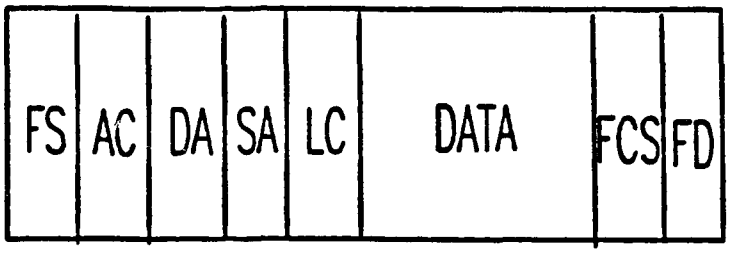
1. Fatal errors are displayed and in some cases followed by a system reset. Sender can also request a reset which is carried out immediately or after the completion of a task.

2. Non-fatal errors in transmission and receiving are displayed and the NI type card(s) or the gateway is reset. The port also perform a subset of the self test routines when idle.

At firmware design time, the type of errors that should be detect and the actions required was decided. The resulting code was added throughout the firmware to achieve the desired degree of error recovery. For example during operation, a time-out may occur on the receiving side. The transmitting node is then notified. If the the receiver does not pass initialization and test, the port is out of operation. The LEDs display the error code for the failing side and operation and field maintenance must follow. The prototype does not send a negative acknowledge, but this can easily be coded into the software at a later time.

Fragmentation and blocking. The two networks have different frame sizes and the gateway must fragment the larger into the smaller, or block the smaller packets into one larger frame. The buffer of ACP is perfect for the operation. Data and control fields are handled separately. Ports have a register file to hold the control fields and only data passes through the buffer. This approach allows the buffer to be one continuous pool of data. There is an end of packet flag in the data that can be set to point to the end of one data segment. These features help the gateway maintain a smooth buffer operation and eliminate the fragmentation and blocking problems.

Format translation. A typical Local Area Network (LAN) message frame, Figure 14, contains a data field and network specific control fields. The gateway disassembles the frame and re-assembles it into a new



FS = FRAME START DELIMITER
AC = MEDIA ACCESS CONTROL FIELD
DA = DESTINATION ADDRESS
SA = SOURCE ADDRESS
LC = LINK CONTROL FIELD
FCS = FRAME CHECKSUM FIELD
FD = FRAME DELIMITER

Figure 14. A typical message frame.

format. Figure 15, shows the format for an Ethernet and an IEEE488 frames - the frame formats may differ in different implementations. The destination address in both cases is the address of the gateway on the two networks. Following this field is a six byte internet address which is added (encapsulated) to the data field of the frames by the sender. This will become the destination address field for the out-going frame. The

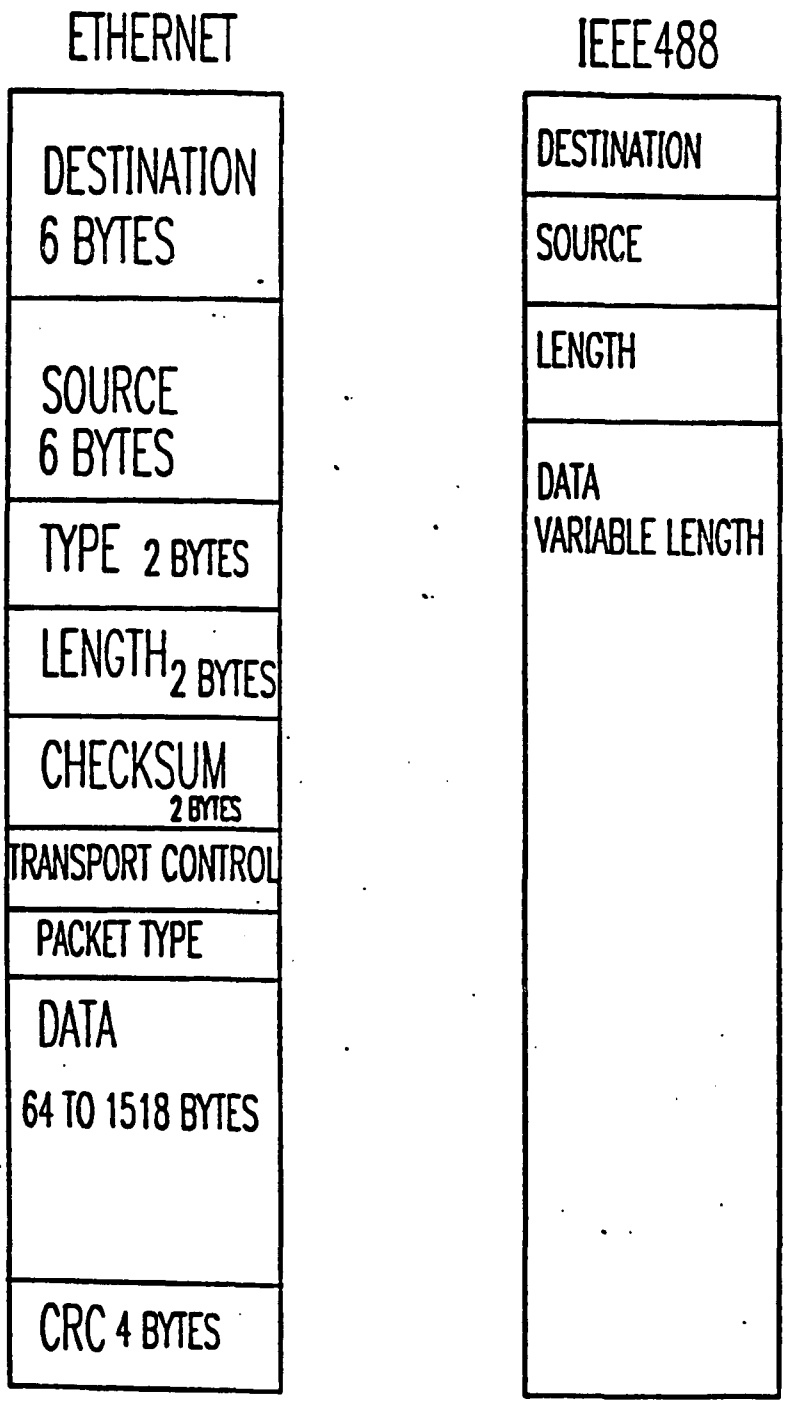


Figure 15. Ethernet and IEEE488 frames.

source address of the 488 packet can be padded by the gateway and the source address of the Ethernet is mapped to the 488. The "length" field is generated by counting the number of bytes on the receiving side and passing it to the sending port. The remaining fields from the Ethernet, except for the data field, are ignored by the controller.

Co-existing with available hardware. There are many network interface cards available in the market. The challenge for the design is accepting two off the shelf NIM cards and configuring a gateway that links the two networks. As a prototype a gateway was developed, based on ACP, with two available cards. Since IBM PC is commonly used in a variety of applications and many networking cards are available for the PC, two IBM PC NIM cards were selected. The implementation is not different in principle if any other card(s) was selected - Digital Ethernet card and National Semi's 488 cards can be adopted, as easily, to configure a similar gateway.

Scenario of How the Gateway Works

Once self test has been completed successfully, the Ethernet and the 488 cards are initialized. The two ports then keep track of the status of the buffer, the NIMs and the opposite port. Suppose that a packet from IEEE488 is to be passed on to a node on Ethernet through the gateway. The network interface card for IEEE488 (TECMAR card) Detects its address on the net. The corresponding port (Port1) detects the condition and branches into the receive routine, it activates the timer and checks for

data available in the buffer of the NIM. It will then read a byte from the NIM and latch it into the funnel. Packet header bytes are written to the port's internal register file, data to the buffer. Having restarted the timer, it waits for the second byte of the word. While receiving the header, a decision is made whether this packet is a command to the gateway. The word is written into the FIFO buffer if this is not the case. FIFO-Empty will then go false indicating that data is available to Port2. The receive loop is ended when Port1 detects an End Of Packet condition, if an error status is read from the NIM or a time out has occurred. The last word written to the buffer has a flag. Port1 then signals the opposite side (Port2) that the packet has been completely received. The control registers are passed to the Ethernet NIM and the data is funneled into 8 bit bytes and written to its buffer. A transmit command is then issued for the packet to be sent on the Ethernet. If the header indicates a command, the data is the type of the command to be carried out. For example a status may have been requested. The task under work and errors are displayed on the LEDs by both ports for every function. As stated, error detection is based on the information in the NIM registers, the timer and the interrupts passed from one port to the other. The gateway can notify a sender of a problem or respond to a status request.

Receiving a packet from Ethernet and transmitting on the 488 is much the same, except, Port1 starts assembling and transmitting when the very first word is available in the buffer from the Ethernet side. In the face of heavy traffic from both sides, the buffer section will buffer data in both directions until one side is near full. An interrupt is generated

to the opposite port to transmit a status packet. However, if the sender does not honor the request and the buffer is filled, packets may be lost.

The motivation for developing the prototype was to link incompatible test equipment and stations such that a basic dialogue is possible for transferring test data at very high rates. Ethernet and IEEE488 was chosen as the target network.

The prototype is an Ethernet to IEEE488 gateway for feasibility purposes. Any other configuration of the system is principally the same except for the NI cards and associated micro code.

Overview of Ethernet

Briefly, Ethernet, developed mutually by Intel, DEC and Xerox, is a base band network with a data rate of 10 Mega bits per second. Up to 1024 stations can share the shielded cable with a maximum length of 2.8 Kilometers. The topology is that of a branching, non-rooted tree. The access protocol is Carrier Sense, Multiple Access, Collision Detection (CSMA/CD) with random back-off. The signal on the coaxial cable is Manchester coded binary. Each node listens for clear coax, and transmits when the medium is available. It will then listen for collisions and "jam" the network if two or more nodes attempt to transmit at about the same time. Each station will wait a random amount of time before retransmission. The packet is of variable length, from 64 to 1518 bytes of data. The physical and data link layers of Ethernet are given in (DEC/Intel/Xerox 1982).

The network is in wide use. It is also used as the basis for other commercial networks, like GRnet, with differences in the details of the layers and their implementation.

Overview of IEEE488

IEEE488 is a network used primarily in instrumentation. This network is base band, with a maximum data rate of one Mega bits per second. Up to 15 primary stations can share the 24 wire, shielded cable with a maximum length of 20 meters. Each station can be a talker or a listener. There is always one talker, but there can be a number of listeners. One of the stations is a system controller at power-up. The controller can then assigns a station to be a talker. The packet is of variable length, one or more listeners are selected by the talker and a block of data is transferred in 8 bit bytes. See IEEE standard 488-1978. This network is in wide use and is used as the basis for other networks.

Physical Specifications

The Gateways is 8x12x20 inches in size as shown in Figure 16. The box houses the DFC card, two network interface cards, power supply and four 5 inch fans. The box splits in two 4x12x20 inch cavities one holding

the power supply and the other the electronics of the gateway. Each half is cooled with two fans so that normal operating temperature remains about 10 degrees above the surrounding's temperature even while engineering work is being carried out on the opened system. Power and reset switches and cable connectors are mounted on the electronics side. The gateway weighs approximately 25 pounds as described. The system requires one 110 Volt, standard plug and draws a maximum of 250 Watts.

Hardware Design (IBM PC Version)

The design of the ACP is taken with no modification to the hardware of the DFC card. The buffer section is 18 bits wide for 16 bits of data plus two control flags. The flags are used as end of packet and packet error markers in the data. The clock circuit is simpler and only half of the circuitry is implemented. Power supply is a Boschert model XL125-3601. It supplies plus/minus 5 and 12 volts at maximum 130 watts. This is very close to an IBM PC supply.

Gateway DFC card. This is the mother card with two slots that accept an IBM PC Ethernet network card by 3COM and an IBM PC IEEE488 card by TECMAR. The design of the card is identical to ACP's DFC card described earlier. Only changes and refinements to that design are given here.

Control Sections of DFC. There are two ports on the DFC card as shown in Figure 4. Each port consists of a micro sequencer and micro

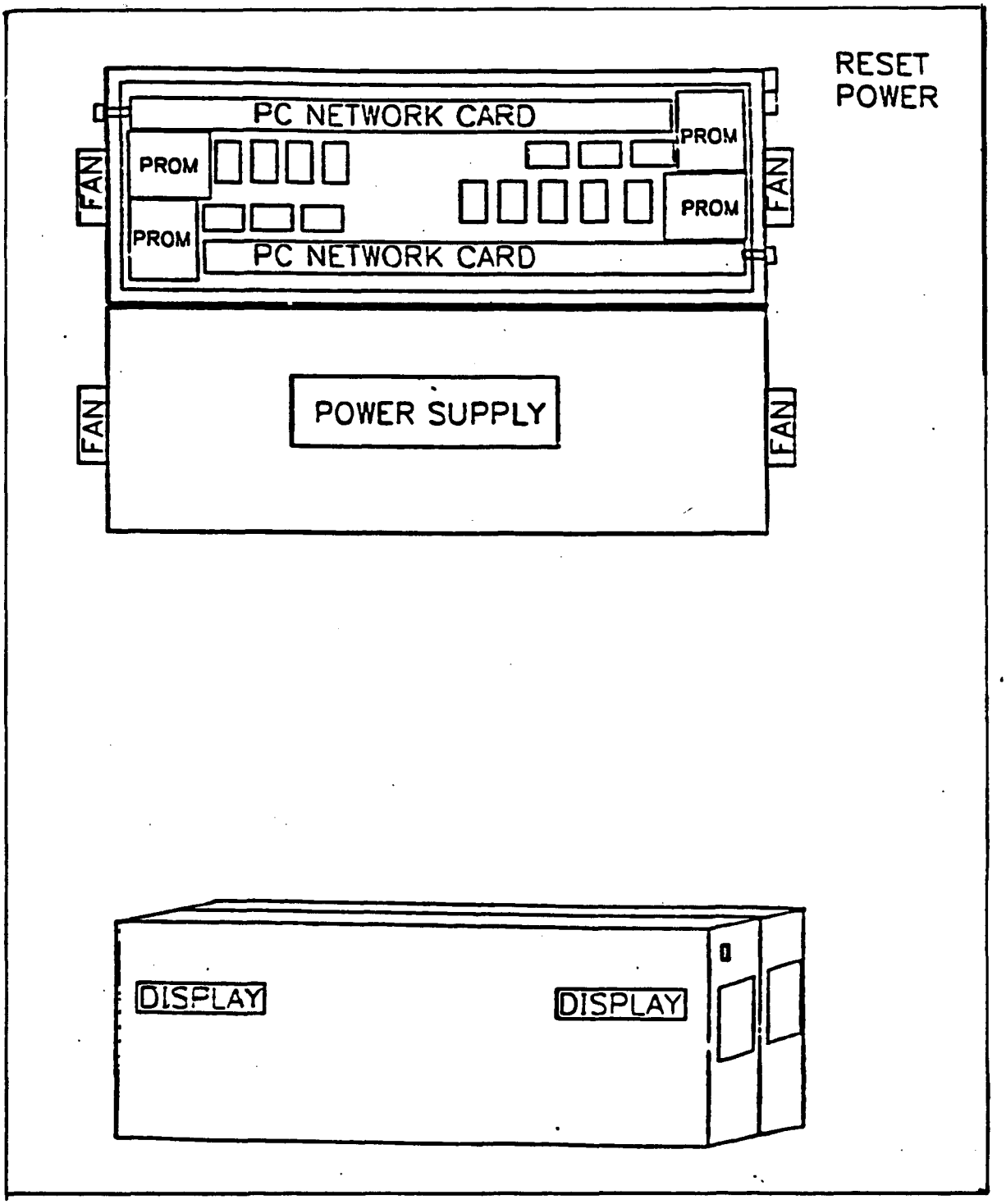


Figure 16. View of the prototype gateway.

program control store, a channel driver, a timer, a condition multiplexer and some latches, registers and LEDs. The functions of the ports are:

1. Interaction with the NIM card. The control signals necessary for the IBM PC Interface of the NIMs are the same for both NI cards. Therefore, control stores of the two ports contain identical code. Only the micro control sequences required to initialize and drive each network interface card is different. Thus, the channel drivers' store has different bodies of code. The entry points are similar.

2. Control the data flow of the buffer. The read and write signals of the FIFO buffer is supplied by the ports and the buffer full and empty conditions are monitored.

3. Exchange status with the opposite port. Two fixed conditional lines report the condition of the opposite port. One is a normal end of task, the other is a fatal error.

4. Visual display the task under work. During power up self test all failures are reported for each of the two port. In normal operations the displays are periodically updated. These registers are open to queries. A special frame will cause the port to assemble and transmit this status.

5. Error handling. If a failure is detected at power up, both sections display codes. The failing port will display an error code, while the operational side points to a failure in the opposite port.

The Sequencer. Is an AMD 2910A of the bit slice family of devices the company offers that drives a 2716 PROM store of 128K bits. For a discussion on bit slice processing see (Donnamaie and white, 1981) and (Mick and Brick, 1980). Also refer to AMD manuals, (AMD Bipolar Microprocessor, Logic and Interface, 1982 Data Book) (AMD Bipolar/MOS memories, 1984 Data Book) for chip details. Every cycle, a control word is accessed from the 2716 PROM bank and latched into the pipeline register. Some bits of the control word are commands to the timer, channel driver, and signals to the interface. A 16 to 1 MUX selects a conditional input of interest as input to the sequencer.

IBM PC Interface. The interface is the same as a IBM PC slot on the IBM PC mother board. Figure 17.

The timer is an AMD2942. It is a safe guard against the problems that the port may encounter. When a critical command is issued by the port, the timer is activated. If the command is not carried out in the time allowed, the port is notified. The timer is checked each time the system is powered and then used in checking other sections of the port. An example of this case is an initialization command to the NIM. The counter is loaded and the command is issued. If the NIM does not respond, the timer will interrupt. Other usage of the counter is clock functions.

The NIM Driver. Is an AMD2940 DMA controller. This device provides specific control sequences to the network interfaces. This will

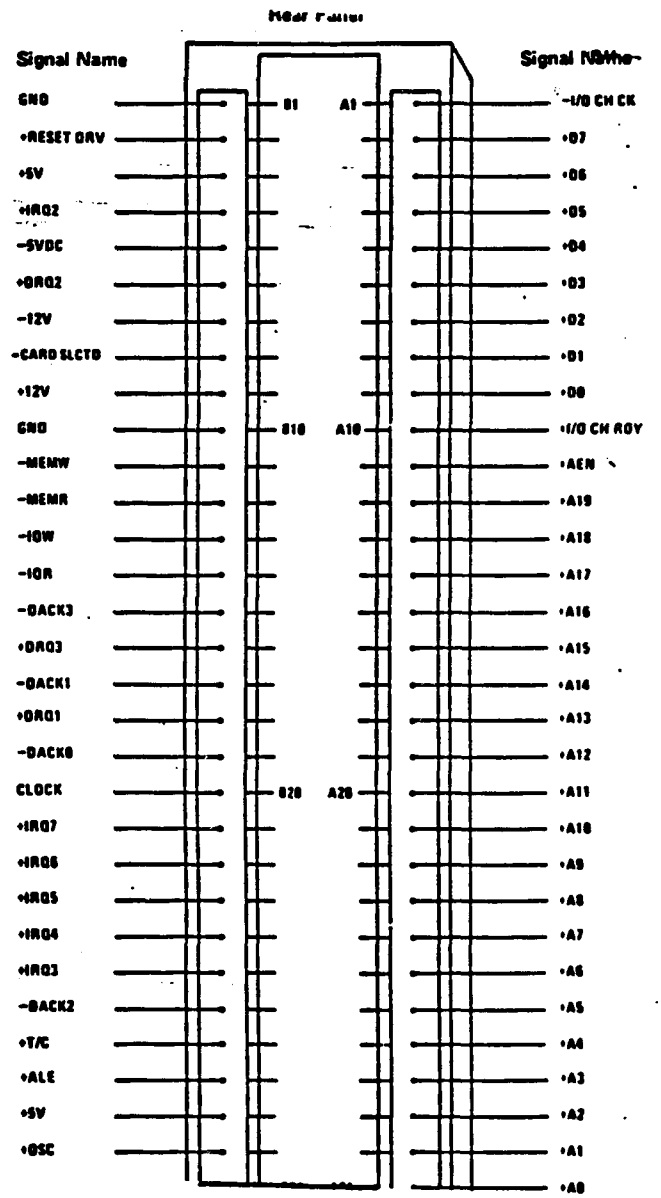


Figure 17. Definition of the IBM PC interface.

off load the sequencer from the burden of both controlling the interface and at the same time sending control and data through the interface. Like the sequencer, the channel driver has a micro control bank of 2716s. NIM specific commands is burned in predefined addresses in the PROMs. The sequencer will load the 2940 with the start address of the routine. This scheme has the added advantage that only the channel driver routines need to be developed and supplied when a different IBM PC network card is attached.

The DFC card was designed to for up to 12 Mega words per seconds. The buffer consists of two banks of Mostek 4501, 9 bit, FIFOs so that data can flow in full duplex. Each bank can be written by one port and read by the other. This operation is asynchronous and can be simultaneous.

The clock section has a 2925, an 8-dip switch, a single pole double throw momentary switch and a 14.31818 Mega hertz crystal. Oscillator frequency is fed to the 2925 which supplies a 4.7777 Mega hertz, 50% duty cycle, to the Interfaces for use by the NI cards. The dip switches program the 2925 for 80 to 720 nano second pulses used on the DFC card, in increments of 80 nano seconds. This set up allows a different set of FIFOs or PROMs to replace the existing banks. The desired cycle is simply dialed. There is an added advantage also, the maximum frequency that the card can operate under is easily found by toggling the switches. The prototype uses 200 nano second PROMs and 80 nano second FIFOs. The board is operational up to 280 nano seconds with these components.

The reset section is implemented using a TI24LS14, OR gates, an RC circuit for 50 nano second pulse and the diode - capacitor circuit handles manual request, diode the local. The diode passes the current to the 470

pF capacitor at low resistance, but provides a 2.2 Mega ohm resistor for pulse generation so that the short signal of the port is stretched. The logic can be part of a Programmable Logic Array (PLA) that implements all gate level functions of the system in a later version.

Board layout is shown in Figure 18. The clock and reset sections are laid in the center of the board and the two ports occupy the top and bottom halves of the card. The board is laid out symmetrically. The terminal connector supplies the power. The 2925 and the dip switch, in the center, mark the clock circuit. The 62 pin edge connectors at extreme top and bottom are the IBM PC connectors - PC1 and PC2. The networking cards use these two.

The top half. Of the card is occupied by one of the two ports. The first row contains three of the pipe line register chips (AMD 29821, 8 bits each) with two more on row 3. Some of the bits from the control store are latched locally in the conditional section, or by the 2910. Four bits specify one of 16 conditions and the fifth selects one of the two 74LS273s (16 bit MUX) in the center right. In row 2 is an 8 bit register, AMD29845. The 2716 PROMs are in row 3 and 4, which implement the main and the driver control stores, along with the AMD2940, address generator of the channel driver section of the port. Below the PROMs and to the right are the AMD2910 sequencer and the AMD2924 status registers. The AMD2942, timer/counter in row five has the FIFOs (Mostek 4501s) to the left. TI 74LS14, LS04 and LS32 are logic gates, in middle left. PC3 and PC4 are used for software development, debug, emulation and field maintenance. PC5 and PC8 are the respective connectors for the channel drivers control store.

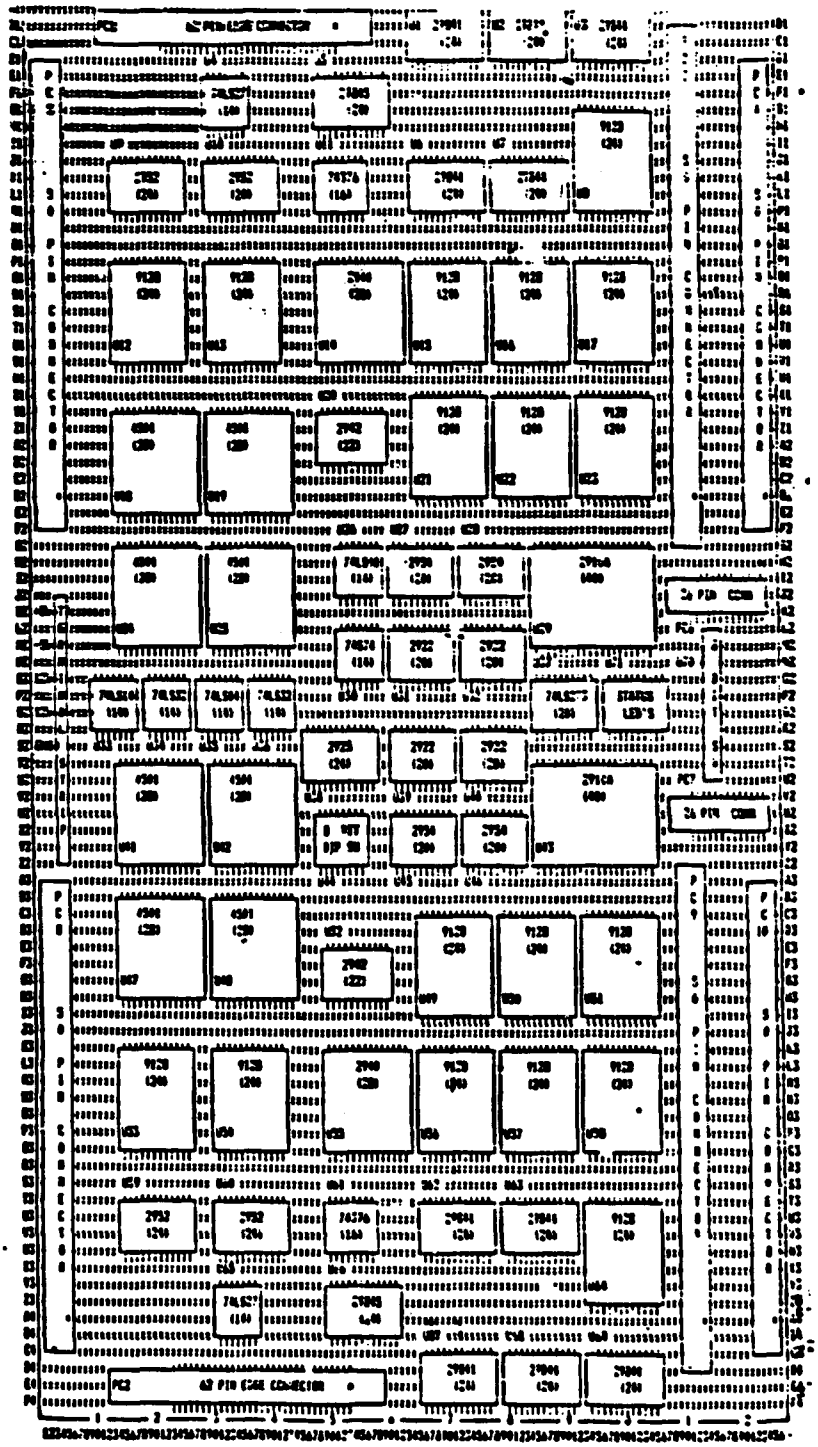


Figure 18. Layout of the DFC card.

The Bottom half. Of the card is occupied by the second port. The layout is an image of the top section. The personality of the two ports are switched by swapping the PROMs and the NI cards.

PC Network Interface Cards

The code for the 3COM card was developed in "C" using the example software and the documentations shipped with the card. No other documentation or consultation was solicited or necessary. The code for the TECMAR card was developed using a timing analysis system. See Micro Code Development section for details.

3COM Ethernet Version. The 3COM card is based on a proprietary chip. A glance at the card reveals that there is an 8001 and an 8023 (Seeq) on board with some address decoding logic and an Ethernet transceiver. The manuals, (3COM Etherseries Internals Documentation, 1984) and (3COM IE Ethernet Controller/Transceiver External Reference, 1984) explain the internal and external specs of the card.

TECMAR IEEE488 Version. The TECMAR card is based on 8291, 8292 and 8293 chips from Intel. (Intel Micro System Components, 1985 Data Book). This is a standard configuration using Intel chips for other IEEE488 and GPIB cards. The address of the card was set at 310 hex, for memory mapped operation and the card installed. This address was monitored for I/O while the supplied object code running on the PC initialized the card. The code was read off the timing analyzer and hand input to the AMD development system. All other routines were developed by

following the same basic steps. References were made to the documentation on the 8291 and the 8292 chips for the details of error bits and status registers of the card.

Micro Code Development

There are two basic approaches to code development: manual and the development tool. Both was carried out for understanding of the time and effort required. Although the code for the 3COM card took only 7 days, this is an optimistic estimate of the effort required. The card is an example of a well engineered and documented product. A more reliable estimate is a month. The code for the TECMAR card was developed using a timing analysis system. The logical address of this card, as most networking card, is switch selectable. If the input output activity of the card at this address is monitored with an analyzer, useful data and timing streams are gathered that represent how the card interacts with the outside world.

Suppose the goal is to develop the code for initializing the card. In step one the card is placed in a PC slot and the software is loaded according to the card's installation manual. In the second step, the analyzer is connected to the slot pins and armed to monitor the logical address of the card. Third, a command is issued to initialize the card. At this point the analyzer has captured the sequence of commands that the card received to initialize. The technique works since the function of a card, the program and the capture technique are all repeatable. This means

that ANY routine, including the original program, which stimulates the SAME card in the SAME manner, will effectively initialize that card. The code is about 1k by 52 bits for either of the two ports, self test and error detection included. The control word format of the port is shown in Figure 19.

Network interface cards are driven by the channel driver of the port. The code is a collection of subroutines with predefined entry names and entry addresses. These routines are called by activating the counter (address generator) under the control of the main control store. The entry and return addresses of a routine are passed to the generator and the generator is enabled. The sequence of control words are sweeped by the generator and passed to the network card through the interface.

Prototype Testing

Testing covers two phases: development and operational. In the first phase one is concerned with the hardware and software of the prototype in meeting the design objectives and specifications. In the second phase, the prototype is operational and the focus is on reliability and maintainability of the port.

1. Hardware and microcode of the port were tested based on the customary "Bottom Up" philosophy. In hardware, each section was tested for connectivity of the lines. Then components were added one at a time, followed by a test. In coding, the basic routines were traced first,

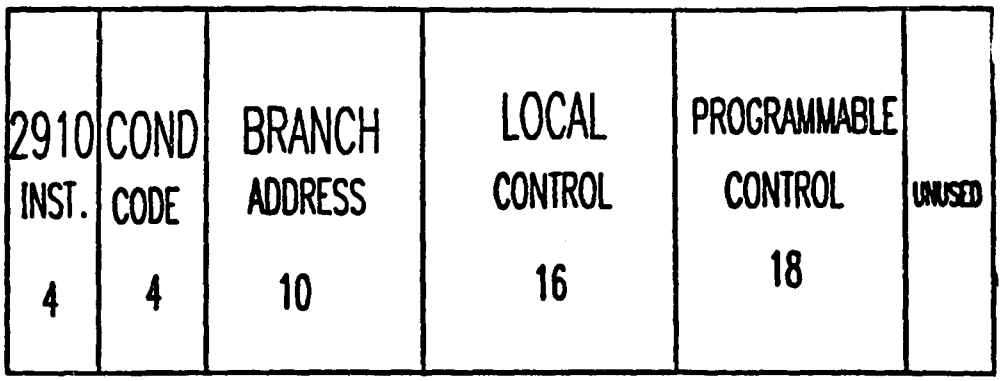


Figure 19. Control Word Format of the port.

then break points were set for the major segments with the aid of the development tools and each segment was debugged.

2. In the operational phase, there are two basic modes of addressing errors and failures. Self testing and error detection. The first addresses faults that are internal to the port, while the second reports and handles external problems.

   a. Self test has two principle goals: to check the operation of different sections of the DFC card, and to check the integrity of the NI cards. The first is a set of fixed routines that must start at address zero. Following this segment is the code that initializes the network interface and checks for the correct status after initialization. These routines can be complemented with a optional loop back test that circulates a dummy packet through the two network interfaces and checks the integrity of the data. This would exercise the send and receive code of the channel driver and the main functions of the network interface.

A successful test will display "F2"s on both LEDs. This means that the port is up and operational and all internal sections have been initialized and are ready. An unsuccessful test will display an error code. The code refers to a section and a function. For example a "06" points to the buffer section's failure to respond to a bufferful condition. An "A0" will mean that the opposite section failed to complete its test. Note that each port tests the well being of the opposite port during self test.

b. Errors encountered in the normal operation are triggered by a status or of an interrupt from the network interface that indicates an error, or caused by a time out from the timer. Both the degree of resolution of errors and the decision to recover are under the control of the code. The choice of requesting a soft reset and/or transmitting a status packet is also part of the code's logic. Many problems may be encountered while receiving, buffering, controlling or transmitting a packet. Whatever the cause, the port displays a code for the failing operation and takes action to recover. Some tests may be deleted or added depending on the choice of networking cards, the needs of the user, and the requirements of the two networks.

3. The gateway has four connectors on board. These are used in the development phase by the emulator. In manufacturing and in the field, they are used by a tester or field maintenance device. Normally, the

sequencer is in control and the control store contains the code that is executed. The connectors provide a means to bypass the clock, sequencer, control store or all. In essence the complete control of the system is passed to the tester. Testing in these phases need only be concerned with the proper operation of the clock, sequencer and the store, functions of all other sections are covered in self test.

CHAPTER 4

FOLLOW ON WORK ON A PROTOCOL TRANSLATION GATEWAY

The ACP is a powerful and flexible design. The media translation gateway that has been developed is fast, reconfigurable and easy to maintain. Media translation gateways, however, do not in themselves answer network connection needs of many customers since the higher level layers of the networks are not covered. Following are recommendations for follow on work in improving the functionality and performance of the gateway.

1. Protocol translation gateway. The primary function of the transport layer is to provide a virtual connection between the session layers of the two network such that packets are received in order and without any loss, errors or duplication. These are accomplished through buffering, sequencing, error checking and acknowledgement of the frames before a block is passed to the higher levels. Note that the layer is implemented, predominantly, in software. The transport layer functions can be added to the gateway.

2. System interconnection. The bridge configuration of the ACP is capable of interconnecting two dedicated machines. The DFC card can be programmed to act as shared memory between the two, integrating the two nodes into a closely coupled system.

Figure 20 illustrates the concept of using ACP as a link between two dedicated machines.

3. Supporting higher level software of the networks in a gateway configuration. The two systems can be nodes on two networks providing a peer protocol translation for layer 4. The transport layer of node-1 (system-1) buffers the packets and provides a virtual circuit to the transport layer of node-2 (system-2), in the same manner as it se·es the session layer, the connection established through the ACP.

4. A gateway between a datagram service and a virtual connection. In a case where one of the two networks provides only a datagram service, the NIM configuration of ACP is sufficient for the network providing a datagram service only. This is illustrated in Figure 21. In general ACP integrates two network nodes, in hardware, into a gateway which greatly reduces the size of the problem and the software development time.

5. Increasing throughput. Where desirable the ports may be upgraded to throughputs above 300 Mega bits per second. There are pin compatible components available for the sequencer section of the ACP, for the control store PROMs and the FIFO section. The clock circuitry can readily generate the higher clock rates for these components and no major design changes are needed. This higher throughput in the bridge configuration will enhance data base and distributed operating system

applications. At least two vendors have these higher speed and density components already on the market.

6. Ease of integration. Two device drivers have been developed for the hosts integrating ACP as a gateway, an Ethernet driver and an IEEE488 driver. A UNIX environment can simply use the device existing drivers since they respond to the generic read, write, control, open, and close commands of the operating system. The programs are also functional in the IBM PC environments under DOS (Disk Operating System), as remote command, message and file transfer services. The program is transparent unless a command line in DOS is proceeded by by a token (! for remote command, * for remote message, and @ for remote copy). However, there is no naming, addressing, or multiplexing implemented. Network commands are simply broadcasted, allowing single client / server implementation only. The receiver (server) only acknowledges to the network. No facilities currently implemented to distinguish between clients or to check the authenticity of the requester. All remote command are carried out, legitimate or not. These techniques are supported under many systems and the code given can be integrated into an existing environment to provide better services to the network.

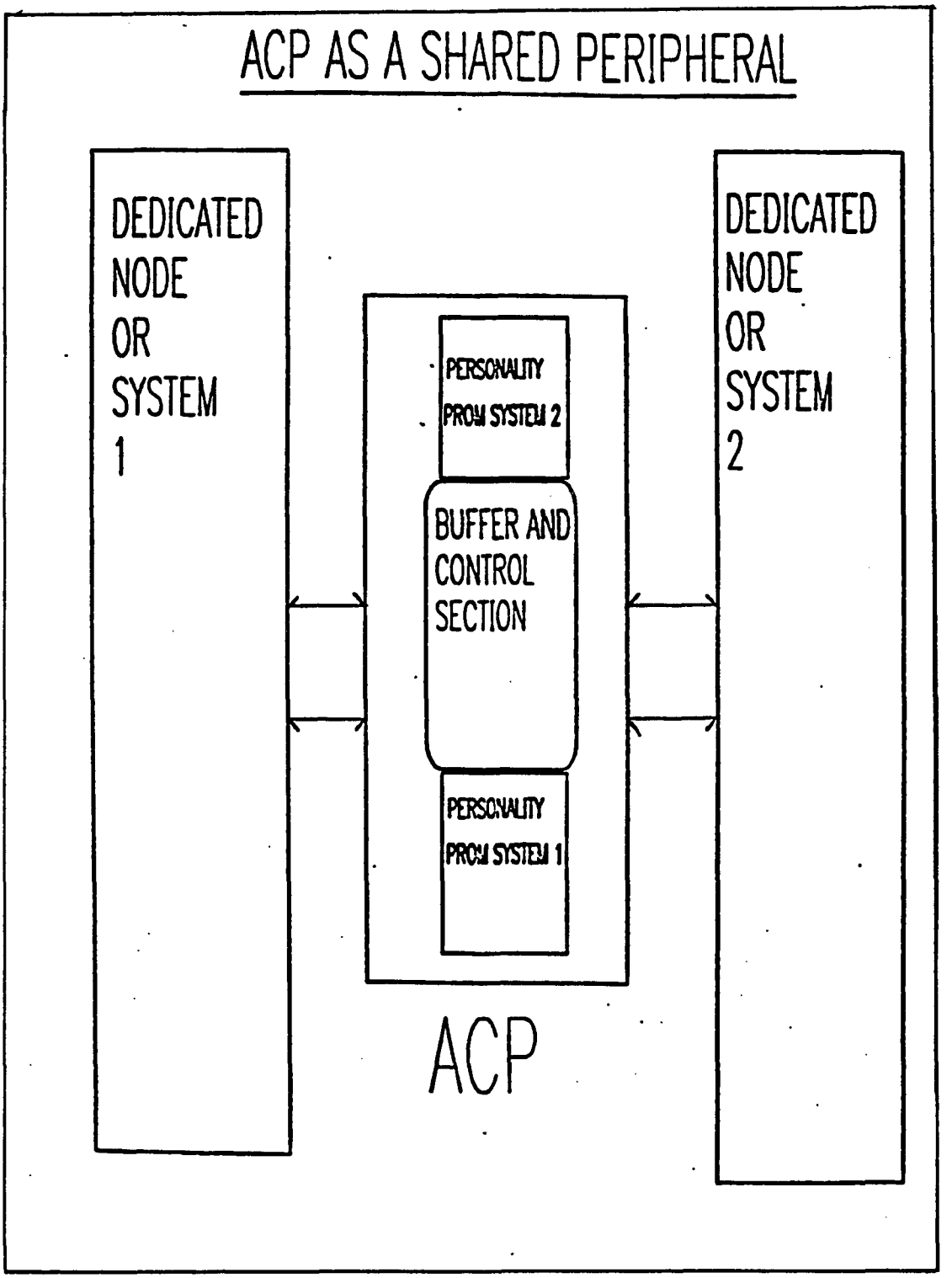


Figure 20. ACP as a link between two dedicated nodes.

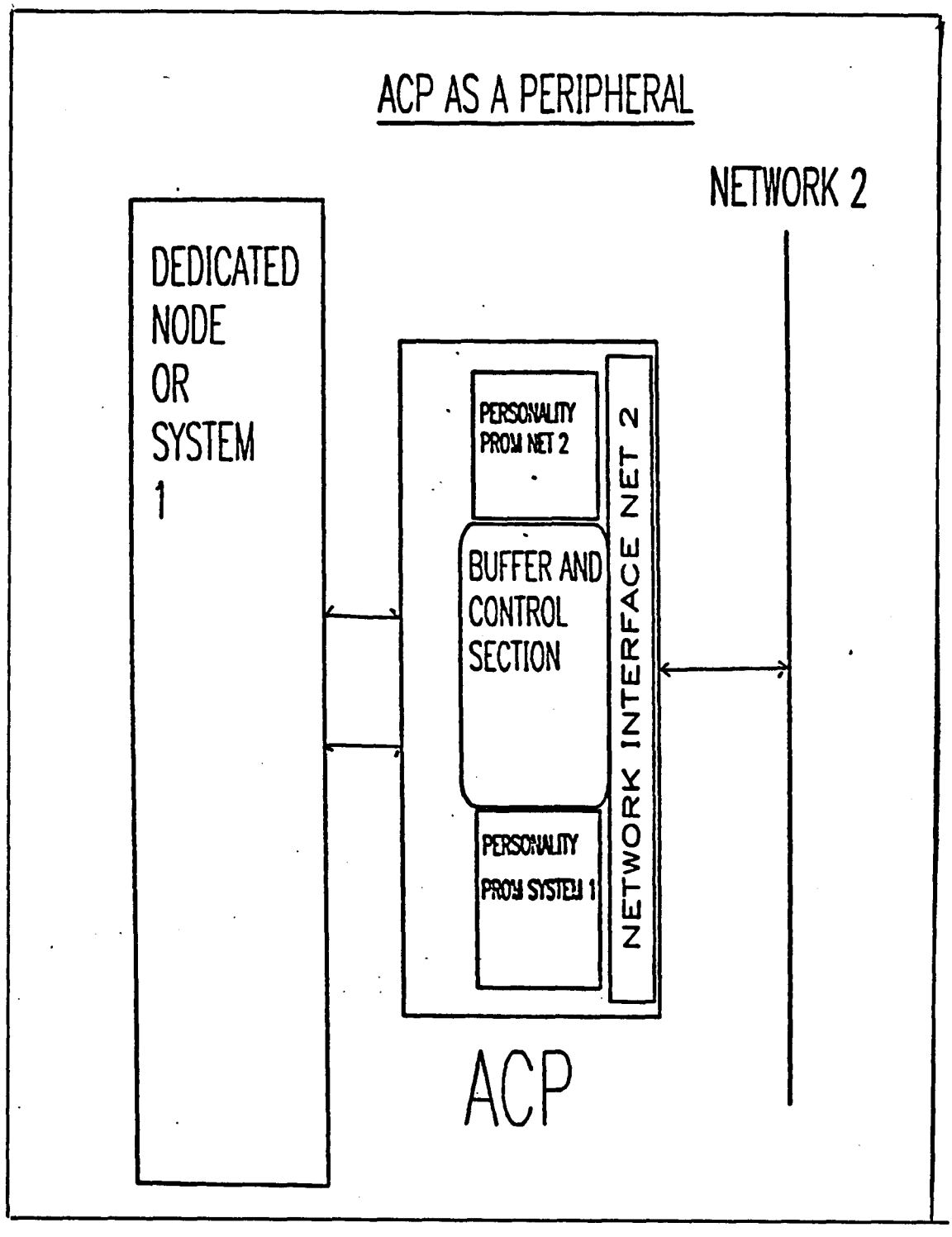


Figure 21. ACP as a link between a dedicated nodes and a NIM.

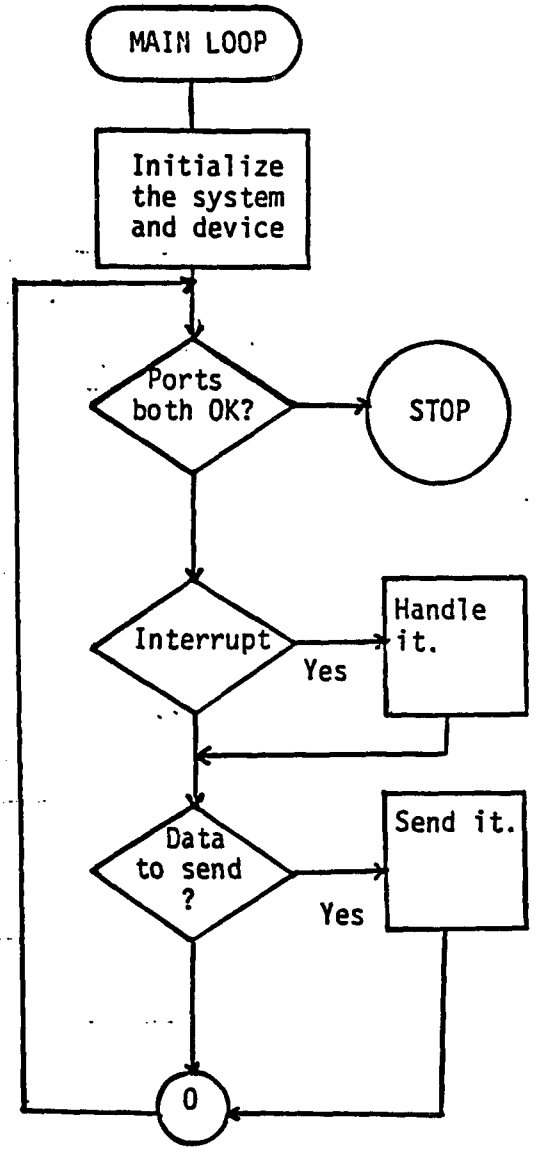
CHAPTER 5

SUMMARY

Design of a data flow and control card for network interconnection has been given. The multipurpose design is fast and flexible. The card has two high performance and programmable ports and a fast full duplex buffer. The single card can interface a number of incompatible buses, allowing the two systems to share resources. The media translation gateway developed interconnects Ethernet and IEEE488 networks using available IBM PC networking cards. The gateway can accept any two IBM PC networking cards by reprogramming the ports only. The interfaces of the ports can also be reconfigured to accept another set of network cards, PC AT or DEC for example. The card can also serve as a base for integrating networks in higher levels, layer 4 and above of ISO reference model. Low level device drivers have been developed to ease application of the ACP to an existing system environment.

APPENDIX: FLOWCHARTS OF CONTROL SECTIONS

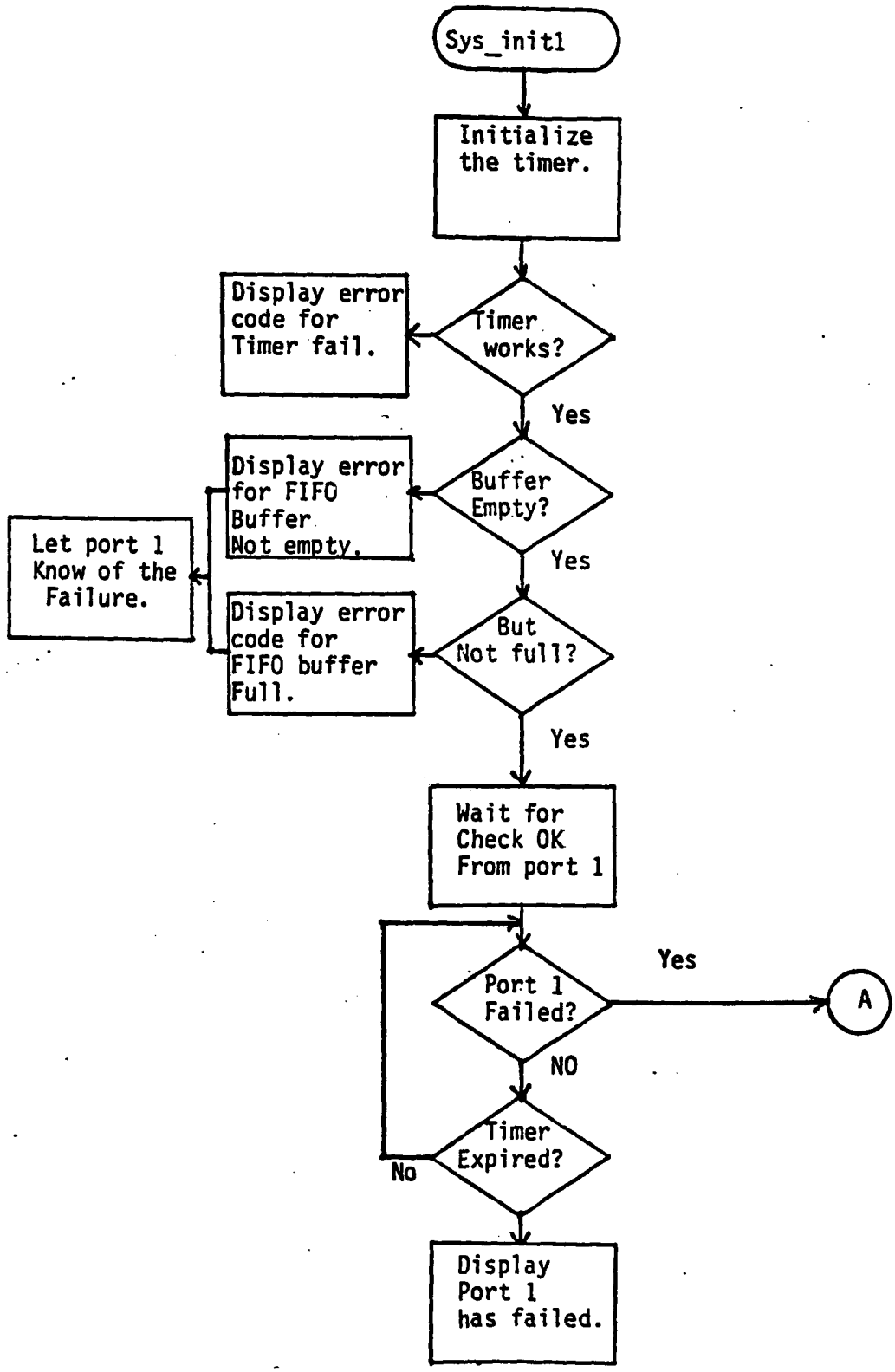
DEVICE INDEPENDENT CODE. ETHERNET AND IEEE488.



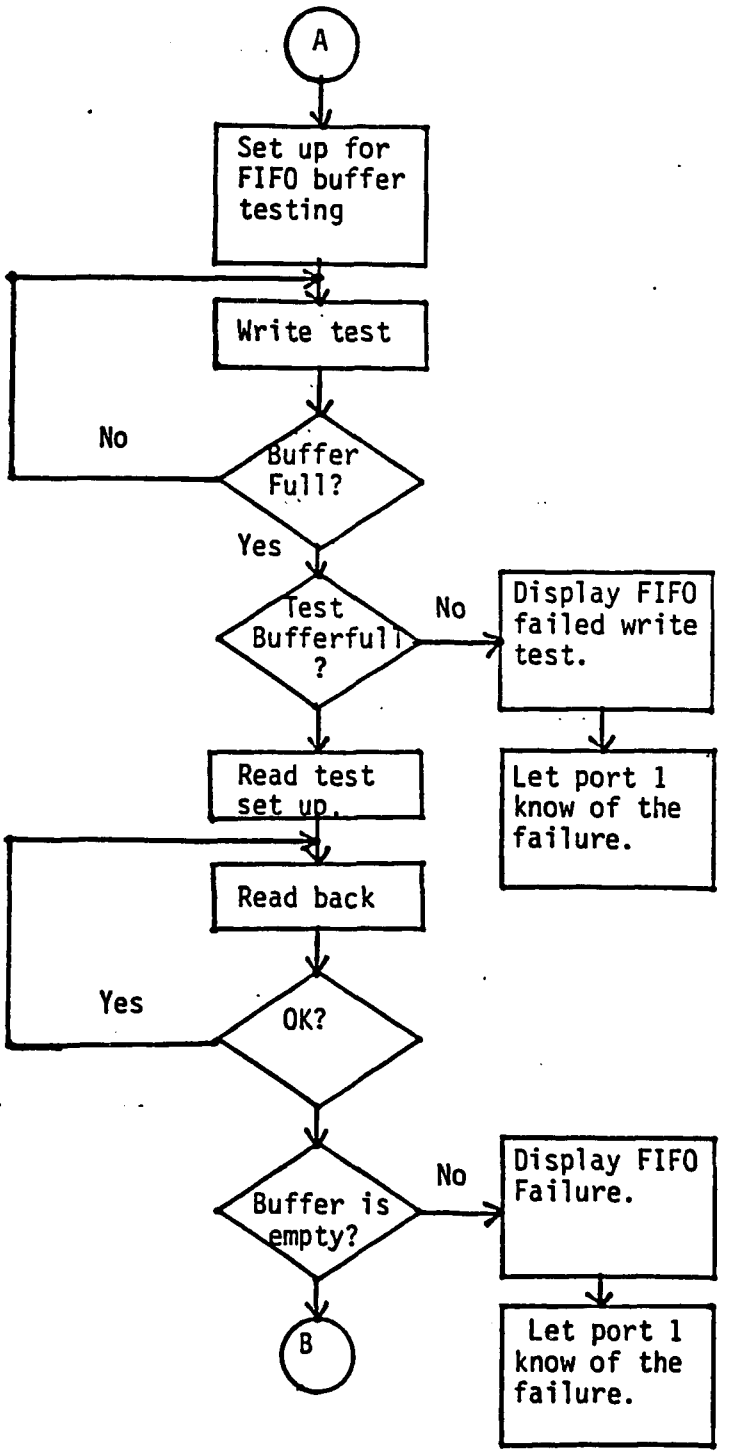
The basic overall flow is to check the hardware periodically, and send and receive.

An interrupt is normally a received packet. Data to send means opposite side received a packet that must be sent out.

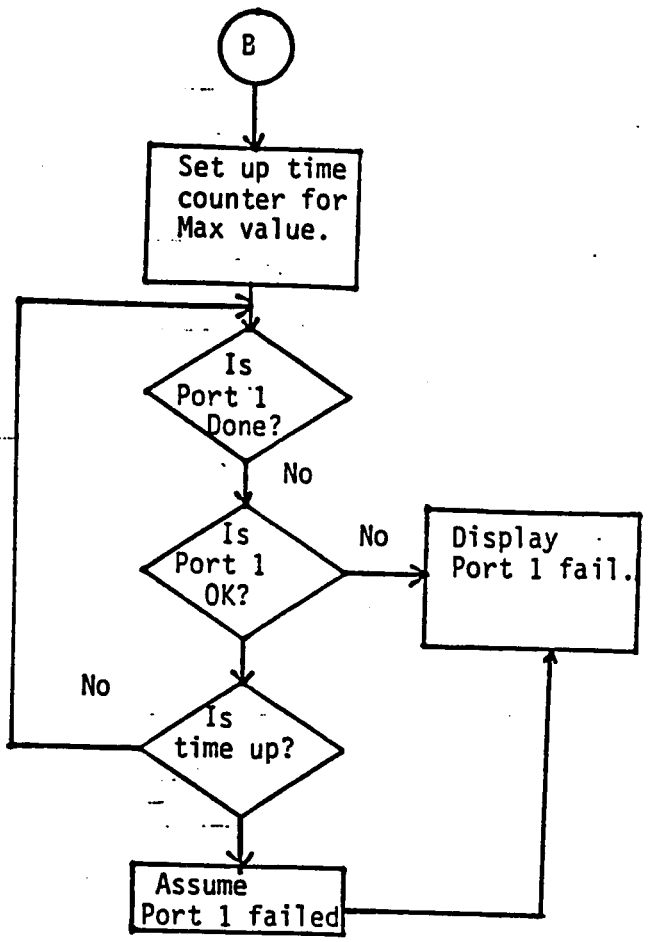
DEVICE INDEPENDENT CODE. ETHERNET AND IEEE488.



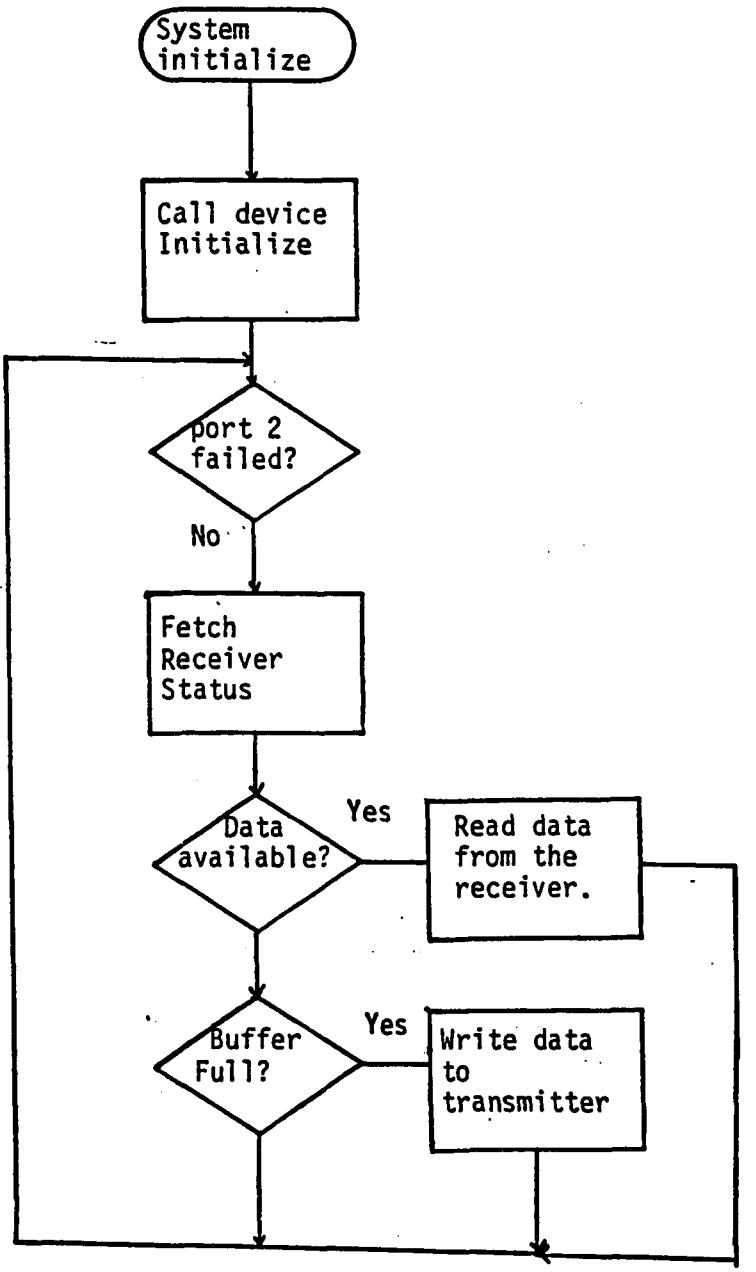
DEVICE INDEPENDENT CODE. ETHERNET AND IEEE488.



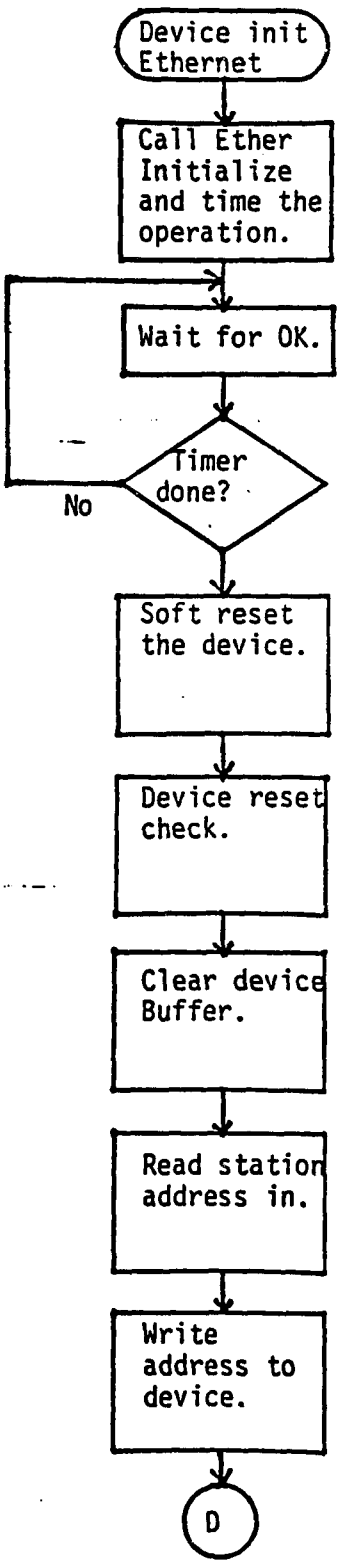
DEVICE INDEPENDENT CODE. ETHERNET AND IEEE488.



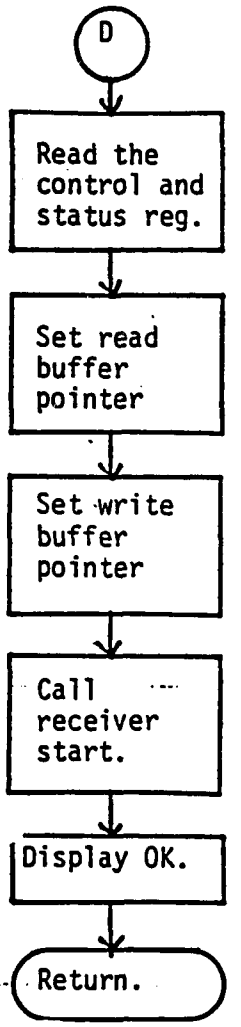
DEVICE DEPENDENT CODE. ETHERNET ONLY.



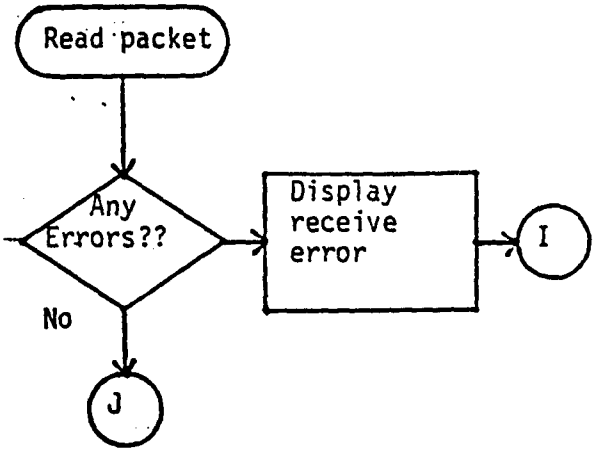
DEVICE DEPENDENT CODE. ETHERNET ONLY.



DEVICE DEPENDENT CODE. ETHERNET ONLY.

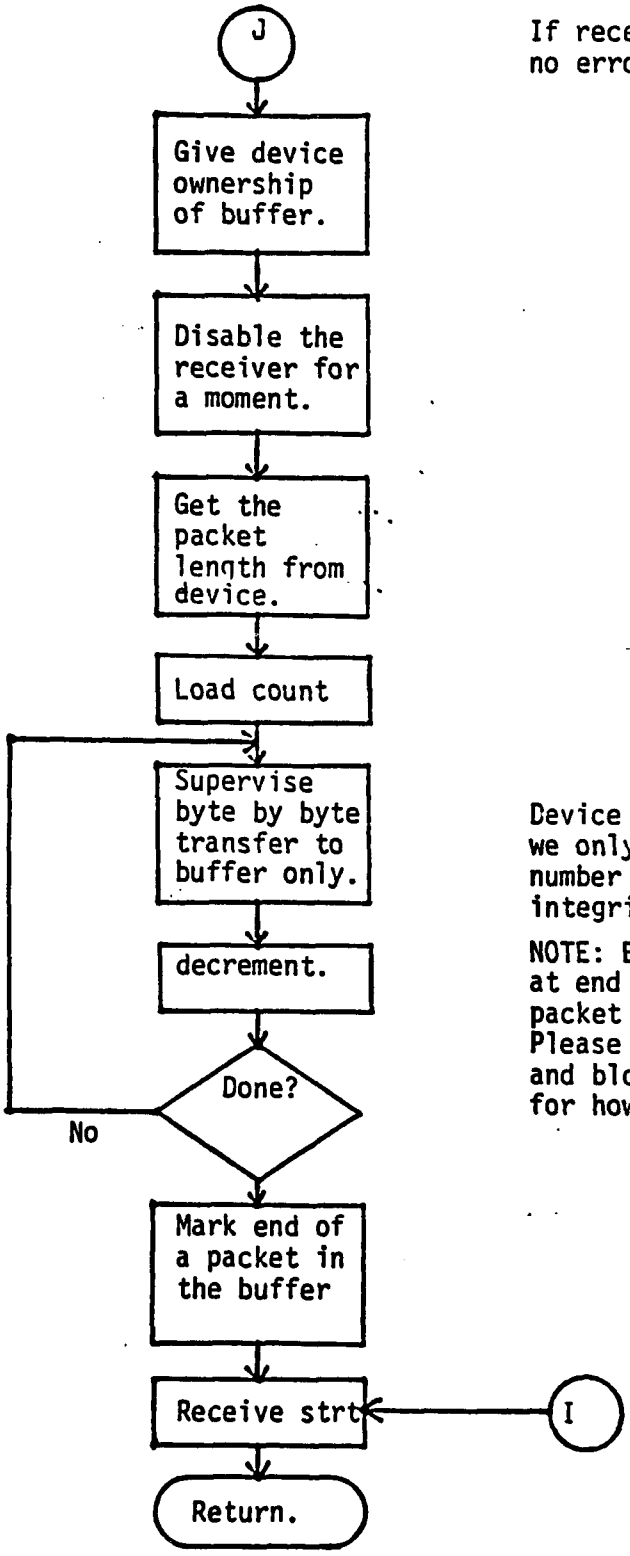


DEVICE DEPENDENT CODE. ETHERNET ONLY.



If there are any errors, report them. That is all that is implemented now. In a more sophisticated version (I) will send negative acknowlegement to the sender also.
For now (I) only restarts the receiver, relying on the sender to time out and retransmit.

DEVICE DEPENDENT CODE. ETHERNET ONLY.

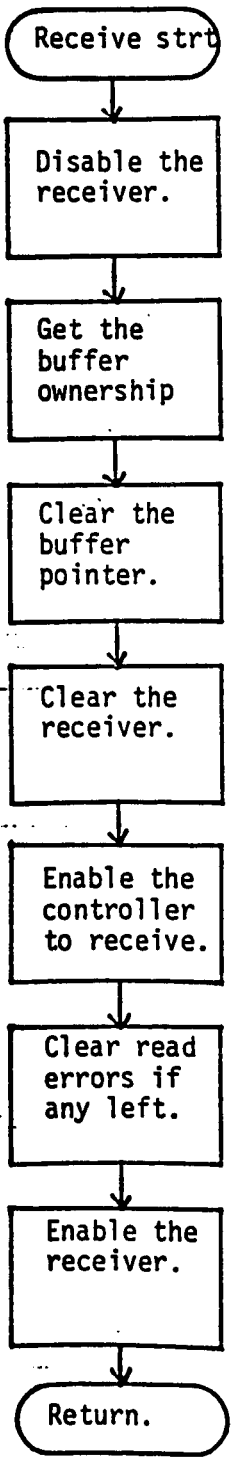


If receive and no errors.

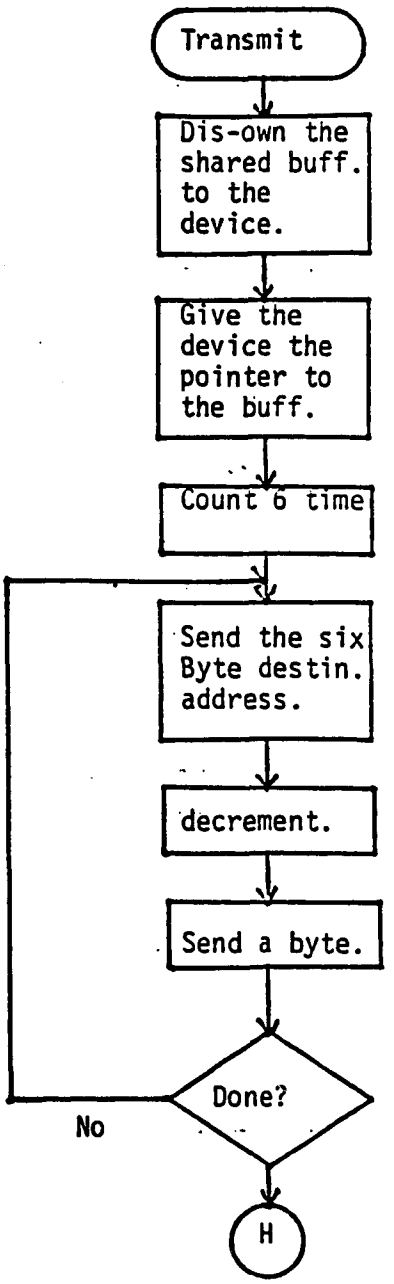
Device does the transfer, we only watch for the number of bytes and integrity of handshake.

NOTE: Buffer is marked at end of each full packet recieved. Please see Fragmentation and blocking discussion for how buffer flow is handled.

DEVICE DEPENDENT CODE. ETHERNET ONLY.



DEVICE DEPENDENT CODE. ETHERNET ONLY.

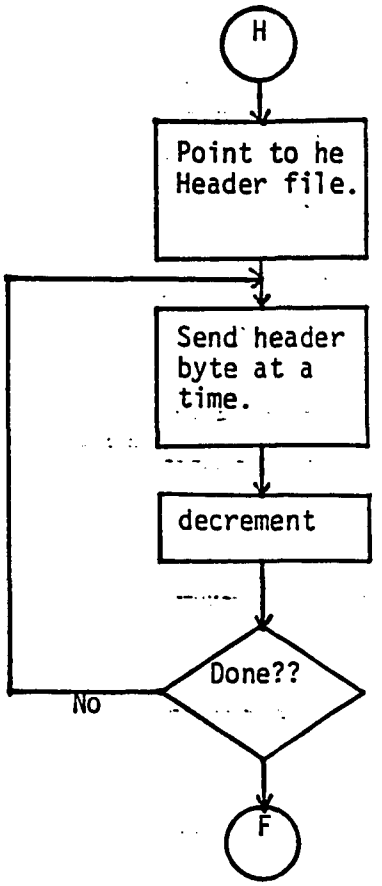


Ownership of the shared buffer is exchanged between the controller and the network interface card.

Destination address is handled separate from the raw data.

Go to send data.

DEVICE DEPENDENT CODE. ETHERNET ONLY.

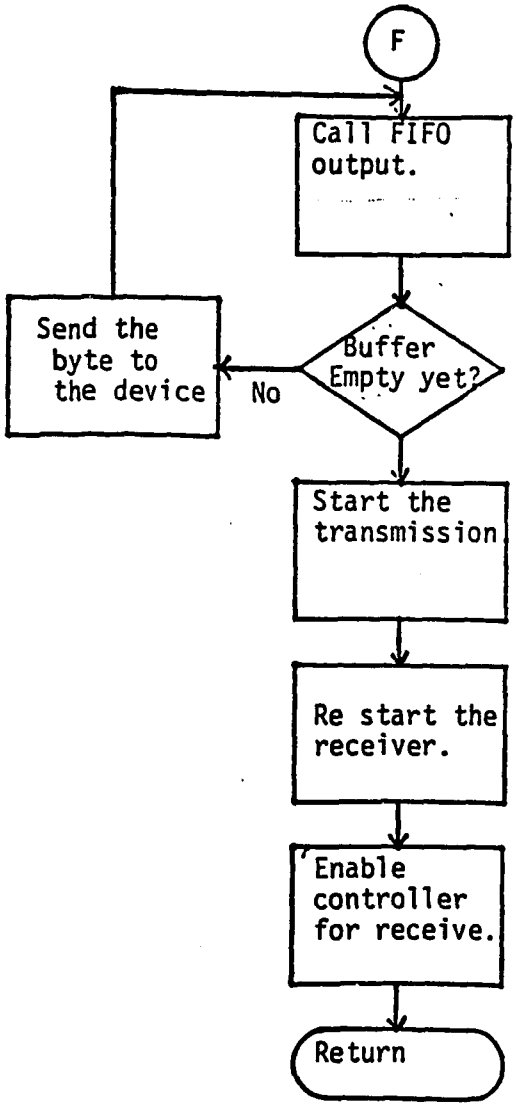


Send the data now.
BUT first the header.

Header (packet) is also handled separately.

Get set for DMA.

DEVICE DEPENDENT CODE. ETHERNET ONLY.

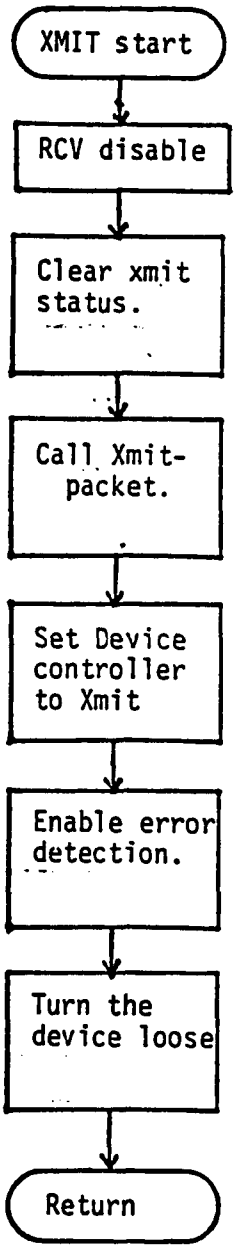


As long as there is data in the buffer send bytes to the device.

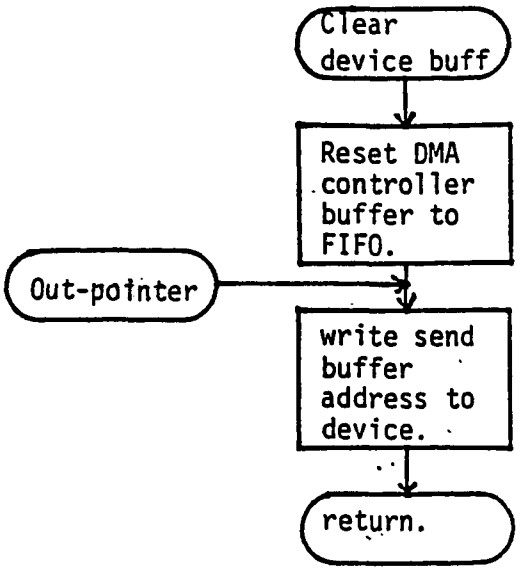
Send the packet out.

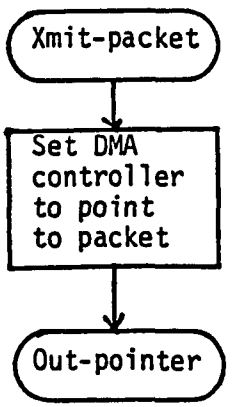
Get back to receiving, DONOT miss a packet.

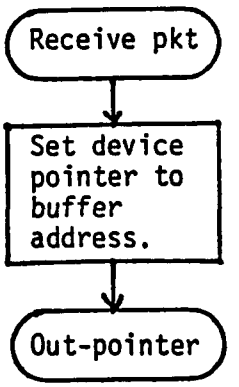
DEVICE DEPENDENT CODE. ETHERNET ONLY.



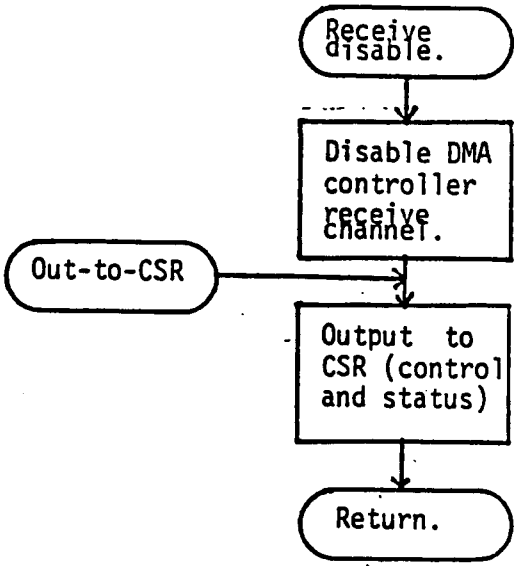
DEVICE DEPENDENT CODE. ETHERNET ONLY.

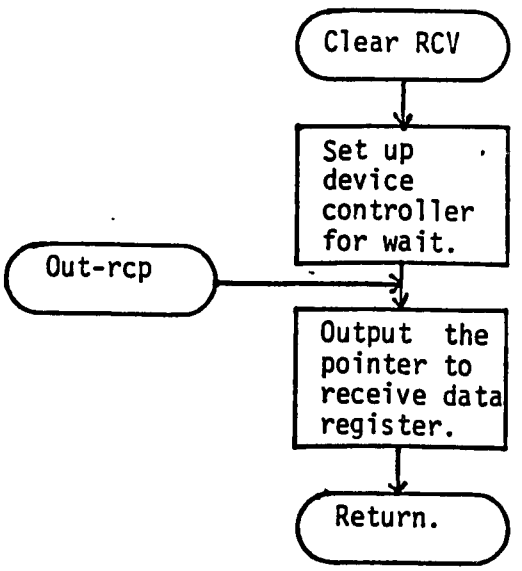




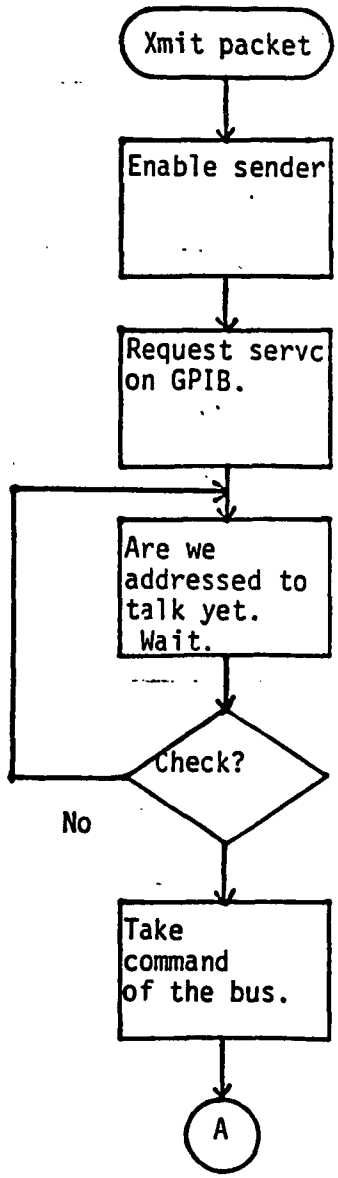


DEVICE DEPENDENT CODE. ETHERNET ONLY.





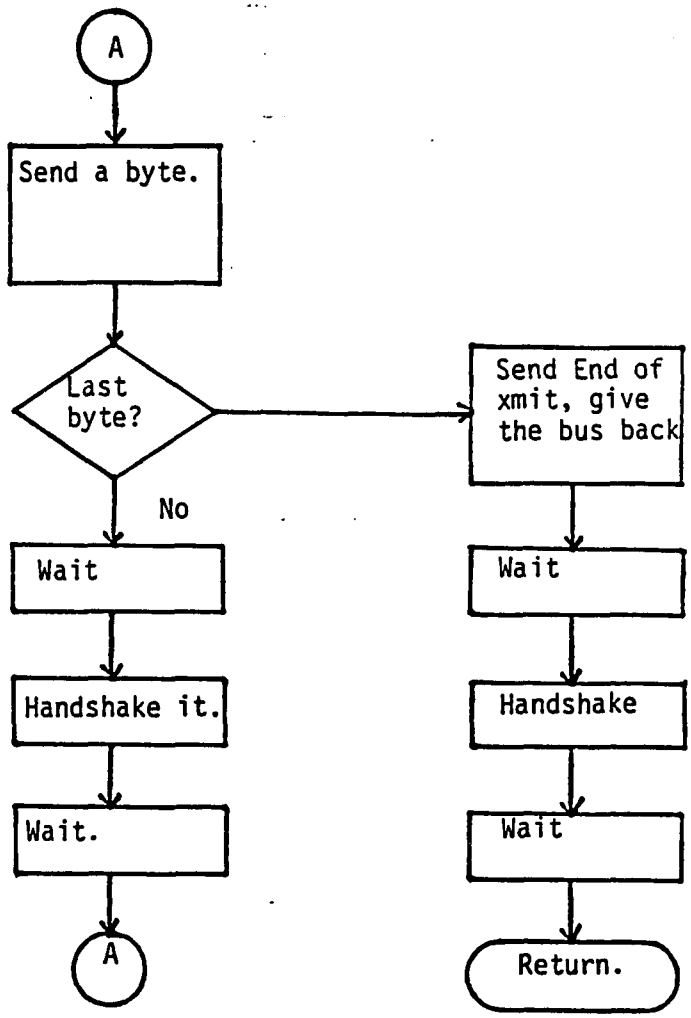
DEVICE DEPENDENT CODE. IEEE488 ONLY.



This flow is for a slave on the bus.

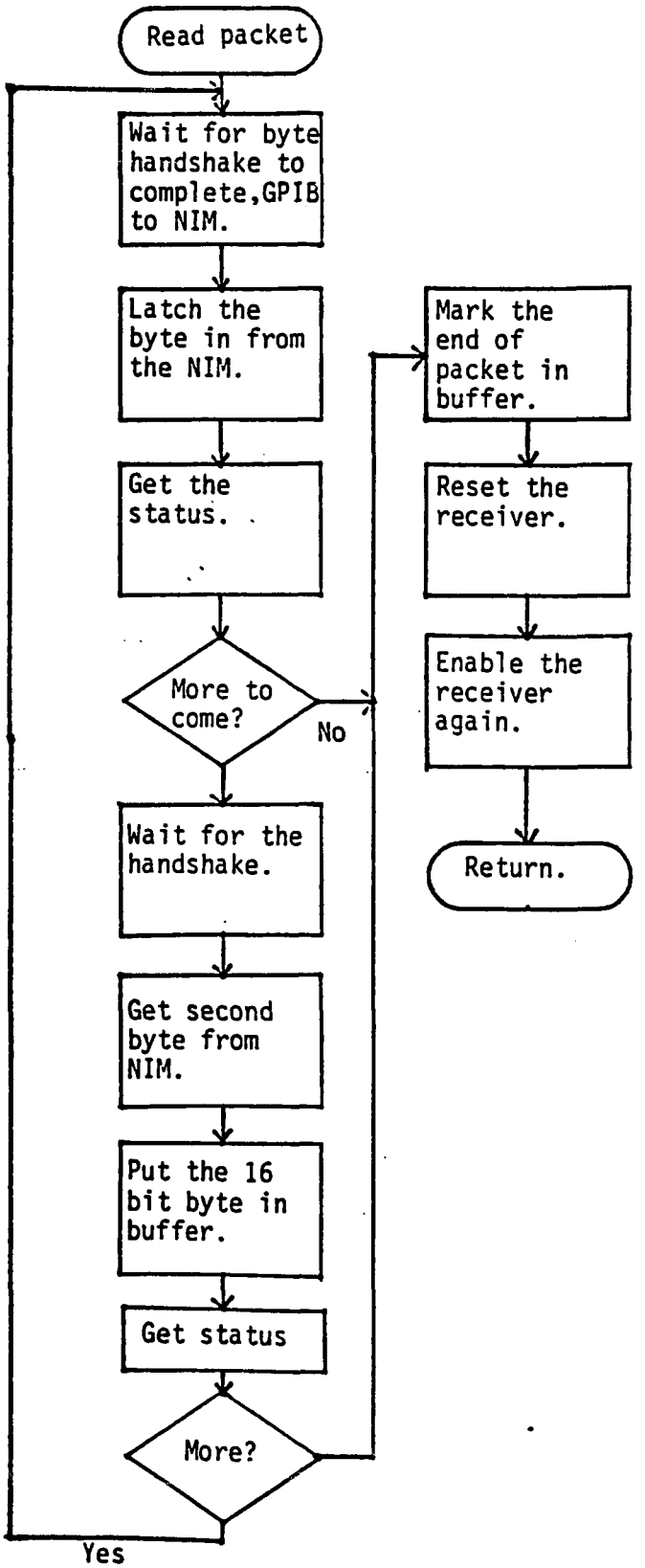
We must be allowed to take command of the bus.

DEVICE DEPENDENT CODE. IEEE488 ONLY.



Waits are there since we run much faster internally, than the GPIB bus can handle.

DEVICE DEPENDENT CODE. IEEE488 ONLY.



LIST OF REFERENCES

1. " 3COM Etherseries Internals Documentation", 3COM Corporation, March 17,1984.

2. " 3COM IE Ethernet Controller/Transceiver External Reference Specifications", 3COM Corporation, March 17,1984.

3. AMD 7990 Family, 1984 Data Book. 901 Thompson Place, P.O.Box 3453, Sunnyvale, California 94088.

4. AMD Bipolar Microprocessor, Logic and Interface, 1982 Data Book.

5. AMD Bipolar/MOS memories, 1984 Data Book.

6. DEC/Intel/Xerox "The Ethernet: A Local Area Network, Data Link Layer, and Physical Link Layer Specification", V 2, November 1982.

7. Donnamaie and White, "Bit Slice Design: Controllers and ALUs" Advanced Micro Devices Inc. 1981.

8. Estrin and Carrico "Gateways Promise to Link Local Area into Hybrid Systems", Electronics, September 1982.

9. Intel Memory Components, 1985 Data Book. 3065 Bowers Avenue, Santa Clara, California 95051.

10. Intel Micro Controller, 1985 Data Book.

11. Intel Micro System Components, 1985 Data Book.

12. International Organization for Standardization (ISO), Reference Model of Open System Interconnection", Document no. ISO/TC97/SC16 N227, June 1979.

13. Martinez R., "Internet Gateway Design for Defense Data Network Access", CERL Report No. 15, University of Arizona, 1986.

14. Mick J. and Brick J. "Bit Slice Processor Design" McGraw-Hill, 1980.

15. Ong M., "Protocol Translation and Translators for Heterogeneous Networks", Lawrence Livermore Laboratory, March, 1982.

16. F. J. Penney, "Automatic Generation of Functional Test Patterns", IBM Technical Disclosure Bulletin, P. 1245, September, 1979.

17. Postel, et al. "The ARPA Internet Protocol", Computer Networks, V5, pages 261-271, 1981.

18. Sapronov, "Gateways Link Long-Haul and Local Networks", Data Communications, July 1984.

19. Shoch, J. "Inter-Network Naming, Addressing and Routing", IEEE Proceedings of COMPCON, pages 72-79, Fall, 1978.

20. A. S. Tanenbaum, "Computer networks", Prentice-Hall, 1981.

21. TI TTL Data Book, 2nd edition.

22. Zimmerman, "OSI Reference Model - The ISO Model of Architecture for Open System Interconnection", IEEE Transactions on Communication, COM-28 4, April 1980.