AN ABSTRACT OF THE THESIS OF

Venkata Ram Atluri for the degree of Master of Science in Electrical and Computer Engineering presented on June 27 1988. Title: A Multiprocessor Node for Communication and Control.

Abstract approved: Redacted for Privacy

James H. Herzog

This paper describes the design and implementation of "A multiprocessor node for communication and control" (MPCC). MPCC is a high speed, high performance local area network capable of performing real time control tasks. This is achieved by the concurrent operation of two 8051 microcontrollers, a dual port RAM and the intelligent and high performance serial interface unit on-board the INTEL 8344.

To achieve the desired high performance in the MPCC, "the division of labor" principle was applied. A dedicated "application processor" is employed to carry out all application tasks and a "communications processor" to take care of all communication needs in the MPCC. These needs include: host to node, node to host, and node to node. The INTEL 8051 microcontroller is used as the application processor and the INTEL 8344 RUPI is used as the communications processor. To increase the frame size of the

transmitted and received frames, the INTEL 8344 microcontroller is operated in the expanded node.

MPCC employs the Token passing method for bus access and uses the RS-485 multidrop implementation for the bus. Data rates as high as 300 kbs have been achieved.

A monitor, debugger program, MDP-44 was developed for the 8044. TASKMASTER [HERZ 86] was modified to receive the command packet from the dual port RAM. Software was developed to operate the SIU in expanded mode for communications.

A Multiprocessor Node for Communication and Control

by

Venkata Ram Atluri

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed June 27, 1988

Commencement June 1989

APPROV Redacted for Privacy

Associ[illegible]uter Engineering in charge of major

Redacted for Privacy

Head of Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented June 27, 1988

Typed by Miss Shu-Ing Ju for Venkata Ram Atluri

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank the following individuals for their support and encouragement throughout my study at Oregon State University.

I would like to give my special thanks to my major professor, Prof. James H. Herzog for his constant support and encouragement throughout my study at OSU. I wish to thank Prof. Bella Bose, my minor professor for his guidance. Special thanks to Prof. John M. Murray and Prof. John Peterson for acting as my committee members and giving me useful advice. I am grateful to Oregon State University for providing me with an assistantship for the last two terms of my study at OSU.

I would like to take this opportunity to thank my parents and family members for their financial support and encouragement without which I would not have come to the USA for my MS. Lastly I would like to specially thank my friend Miss Shu-Ing Ju for helping me prepare this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation:

The need for a high performance, high speed and low cost Local Area Network capable of carrying out distributed tasks in real time prompted the design and implementation of the MPCC. This is achieved by operating two Intel 8051 microcontrollers and the high performance, intelligent serial interface unit (SIU) on-board the Intel 8344 microcontroller concurrently.

1.2 Background:

In the beginning man created the microprocessor. With the evolution of the microprocessor, we saw the advent of personal computers, engineering workstations, microcomputers and robots. All of the above have the capability of performing a variety of tasks all by themselves. The concept of distributed processing emerged with the advent of these small but powerful computing elements. The need to connect these computers emerged to satisfy the users requirements such as:

1. Sharing and exchanging information and data.
2. Sharing resources.
3. Provide distributed processing.
4. Parallel processing.

As data communication technology advanced progressively, the distinction between parallel and distributed processing became smaller and smaller. In an extended sense, distributed processing may be viewed as a form of parallel processing. [KAI 84]. Thus the importance and usefulness of distributed processing and computer networks grew steadily.

The goal of this thesis was to design and implement a node that can execute both communication and application tasks concurrently, thus maximizing the efficiency of each node. Moreover, the communication speeds have to be improved if we are to take advantage of the concurrent operation of the application and communication processors.

In a network the hosts are connected by a communication network often referred to as a subnet [TANE 81]. Subnet is divided into switching elements and transmission lines. In the MPCC a SIBEC II, single board computer is used as the switching element. These switching elements commonly called IMPs (Interface message processors) take care of establishing the communication with other IMPs, passing messages to them, and providing error control.

The transmission lines, also called the communication channel, is the medium through which all communications take place. Commonly used media are coaxial cables, fiber-optic cables, twisted pairs and radio transmission. The MPCC employs a twisted pair for its transmission line.

To standardize the network protocols, the International Standards Organization (ISO) introduced the Open System Interconnect (OSI) model allowing computers of different types to be connected. The IEEE Local Network Standards Committee (Project 802) developed LAN access standards and LAN protocols [FRED 86] similar to the ISO OSI reference model. These standards are called the IEEE 802 standards. A more detailed discussion on LANs and their standard is presented in chapter 2.

1.2 Overview of the MPCC:

During the past few years, considerable effort has been expended to develop a control oriented local area network at Oregon State University. This effort resulted in the development of COLANs I - IV

COLAN I, developed by Y.P. Zheng [ZHEN 86], was a modification of the daisy chain structure of TASKMASTER [HERZ 85]. It used a hybrid medium access method of token passing and CSMA/CD. It was not fully implemented. S.K. KAO [KAO 87] improved upon this and developed COLAN II. He used token passing method of bus access. D.H. EUM [EUM 87] developed COLAN III with CSMA/CD for bus access. YONG THYE [THYE 88] developed COLAN IV. This version was very mature and included a powerful user interface.

The MPCC consists of two powerful microcontrollers operating concurrently. The Intel 8051 microcontroller is used exclusively for running all application tasks. The Intel 8344 which consists of an Intel 8051 core and a high

performance and intelligent serial interface unit (SIU) is used as a communication processor, the two processors, ie. the Intel 8051 and 8344, are interfaced through a dual port RAM, allowing interprocessor communication.

Each MPCC node consists of two SIBEC II single board computers. Each MPCC is connected to an RS-485 EIA standard bus. Also each MPCC is connected to its host, an IBM PC through an RS-232C standard bus interface. Since the network is control oriented, a token passing scheme was employed for bus access.

The MPCC employs a bit synchronous mode of communication in contrast to the asynchronous mode used by COLANs I - IV. This is possible because of the SIU in the 8344. The SIU runs concurrently with the 8051 cpu and implements the SDLC/HDLC format. It can be said that the communication software has been implemented in silicon. The SIU thus reduces a great deal of overhead on the CPU. The user software is also greatly reduced. The SIU is capable of communication speeds of up to 2.4 Mbps external clocked and 375 Kbps self clocked. (calculations based on a 12 MHz crystal).

Besides the above features, the full 8051 features are available to the user. The SIU and the CPU are interfaced through an on-chip dual port RAM. A few special functions registers are available to program the SIU. A total of 192 bytes in the dual port RAM has been reserved for transmit and/or receive buffer. Thus the SIU can receive or transmit

a 192 byte frame without any CPU intervention. However by manipulating the SIUST (SIU state register) the 8344 can be used to receive and transmit using an external data buffer.

This chapter has served to introduce the MPCC. A brief explanation of its most important features and its structure has been presented. Chapter 2 gives an introduction to LAN standards and terminology. Chapter 3 presents the hardware structure of the MPCC. Chapter 4 discusses the software developed for use on the MPCC. Chapter 5 presents the implementation of the MDP-44, a monitor debugger program developed for use with the 8344. The final chapter of this report identifies some areas for future development.

CHAPTER 2

Local Area Networks - A review

2.1 Introduction:

Often it is necessary to transfer data from one computer to another. It was pointed out in chapter 1 that work stations, microcomputers and personal computers are computers in their own right. However they are often connected to other computers and the required connection is achieved through a communication network. Networks that span a small geographic area, with distances not exceeding a few kilometers are called Local Area Networks (LANs).

To provide error-free and maximally convenient information transfers, the network operation is regulated by a set of rules and conventions called network protocol [LESZ 86]. The protocol defines connectors, cables, signals, data formats, and error-checking techniques as well as algorithms for network interface. A brief introduction to the ISO OSI references will be given next followed by a discussion of the IEEE standards for LANs [IEEE.82], [IEEE.83] on medium access control.

2.2 ISO OSI reference model:

The need to allow computers produced by different manufacturers to communicate with each other on a network led the International Standard Organization (ISO) to provide a network architecture model referred to as open systems

interconnect (OSI) model. The goal of ISO OSI model was to provide a framework for network standards acceptable to all manufacturers, yet allow their unique products to communicate with each other. The OSI model is general and applies both to wide area networks and LANs. In the ISO OSI reference model all the networking functions are partitioned into "layers". Seven layers have be defined by the ISO OSI model (Figure 2.1). One of the benefits of this layered architecture results from the fact that a higher layers can use services provided by lower layers and do not have to deal with details of the lower layer's operation. The function of the 7 layers is summarized below.

The physical layer provides a physical path for electric signals representing bits of transmitted data. It also defines the characteristics of these signals, such as voltage and current levels, frequencies, and timing. It also specifies the mechanical properties of network cables and connectors.

The data link layer defines rules of sharing the use of the physical layer among network nodes. Information is transferred in frames. The format of these frames and the method by which information is transmitted or accepted is defined. Addressing, error-detection and error-correction are used to ensure accurate and error free transmission.

The network layer is responsible for buffering and routing of data packets. It can be viewed as routing data packets through a virtual data link. It is basically a

software layer. Routing makes sense in wide area network but does not have much meaning in a bus topology LAN.

The transport layer partitions the data from the session layer into smaller units. It isolates the session layer from the inevitable changes in hardware technology. On the receiving side, the transport layer reassembles packets and ensures that they are not misplaced. So the transport layer can be viewed as a boundary below which a data packet is a unit of information handled by a network. Above the transportation layer, messages are regarded as information units.

The session layer is responsible for providing a communication session between two user's processes running on two separate nodes. It is the user's interface to the network. A session is created on the request of a user process submitted through the application and presentation layer. The session layer is responsible for maintaining and terminating a session.

The presentation layer presents the information given by the user to the lower layers. It encodes data to achieve security. Above the presentation layer data fields are meaningful messages; below it, packets are treated as meaningless cargo.

The application layer serves as a boundary between the OSI network and the user process. In a distributed system, the application layer is responsible for direct communication with the elements of the distributed operating

system. It would also include many utilities such as mail and file transfer protocol facilities. It is the seventh and the topmost layer in the ISO OSI reference model.

2.3 802 Standards for LAN:

The IEEE Local Network Standards committee (project 802) developed LAN access standards and LAN protocols in a layered approach (Figure 2.2) similar to the ISO OSI reference model [FRED 86]. Six such standards have be been defined, they are summarized below. However only the CSMA/CD and Token-Passing bus access methods will be described in detail.

IEEE 802.1	Addressing, internetworking and network management.
IEEE 802.2	Logical link control (LLC) common to the various types of media implementation.
IEEE 802.3	Carrier Sense Multiple Access and Collision Detection (CSMA/CD) access method and physical layer specifications
IEEE 802.4	Token-Passing bus access method and physical layer specifications.
IEEE 802.5	Token-Passing Ring access method and physical layer specifications.
IEEE 802.6	Metropolitan Network access method and physical layer specifications.

2.4 IEEE 802.3 (CSMA/CD):

The Carrier Sense Multiple Access with Collision Detection (CSMA/CD) is a contentious scheme. Here network nodes contend to capture and use the transmission medium. It essentially uses a bus topology for the network and communication in among peers.

A state diagram for the CSMA/CD algorithm is given in Figure 2.3. If a node wants to transmit a message it checks for any activity (carrier sense) on the bus. If the bus is free the node transmits its message. If several nodes transmit at the same time a collision occurs. In this case all transmitters abort their transmission, and back-off for some time before trying again. The back-off time is random and is calculated anew after each collision. A brief jamming signal is transmitted to let every station know that a collision has occurred.

Usually each node that is in the listen state receives a frame and checks for its ID. If the received ID matches its ID (MID), then the node continues to receive else it aborts reception. Upon a request from the user it may enter into a transmit mode.

2.5 IEEE 802.4 (Token-Passing bus access):

Token Bus is a more complex bus access method than CSMA/CD. It is a noncontentious method in which the right to use the network medium is passed from node to node in an organized way. In this technique a logical ring is formed by the station on the bus. Each station knows the identity

of the stations proceeding and following it. The physical ordering of the stations is not important. All stations on the bus need not be in the logical ring. A station not on the bus can not transmit but can receive a frame. A typical Token Bus structure and its logical ring is shown in Figure 2.4. This is achieved through a special frame called "token", the state diagram for the normal operation is shown in Figure 2.3.

The IEEE 802.4 token passing bus access protocol defines the following for its proper function.

1. Ring initialization
2. Addition of a node
3. Detection of a node
4. Fault management by token holder
5. Classes of service

2.5.1 Ring initialization:

After the system power up, or a logical ring break down, a ring initialization is required. Initialization is a special case of addition of new nodes which will be discussed next.

2.5.2 Addition of a node:

A controlled contention process called "response window" is used to accomplish the addition of a node. Each node periodically grants an opportunity for new nodes to enter the logical ring. The token holder issues a solicit-success frame, inviting nodes with an address between itself and the next node in the logical ring to enter the ring. If

a node wishes to enter it will issue a "set-successor" frame. The token holder designates it as its successor and transfers the token to it. This can similarly be extended to multiple nodes entering the ring.

2.5.3 Deletion of a node:

If a node wishes to drop out, it waits until it receives the token. Then it sends a "set successor" frame to its predecessor. The predecessor then sets its successor as the successor of the node wishing to drop out.

2.5.4 Fault management by the Token holder:

Faults in a Token Bus topology might occur due to one of the following reasons (a) multiple tokens, (b) lost token (c) unaccepted token and (d) failed station.

If a token holder detects the presence of another token, it removes its token from the bus immediately. Thus reducing the tokens to either 1 or 0. This results in a lost token case. In this case a logical ring initialization should be done. If the successor does not accept a token a case of unaccepted token arises. The token holder assumes that the successor node failed. To find the new successor the token holder issues a " who-follows" frame. If the token holder receives a "set-successor" frame it sets up its new successor. If no such response is received, the token holder issues a "solicit-successor" frame inviting every node to respond. If there is still no response, the token holder goes into a listening mode. At this stage a logical ring initialization is required.

2.5.6 Classes of service:

Four classes of services are defined by the IEEE 802.4 standard. They are (a) synchronous (b) asynchronous urgent (c) asynchronous normal and (d) asynchronous time-available. The purpose of these services is to allocate network bandwidth to the higher priority frames and send lower priority frames when there is sufficient bandwidth. A station may hold data in more then one of the above classes waiting to be sent.

2.6 MPCC and IEEE 802.4 Token bus:

The INTEL 8344's SIU supports synchronous communication only. Since synchronous communication involves transfer of frames, collision detection can not be done in the true sense. Whereas in asynchronous communication collision detection is possible, as it is byte oriented and the transmitter can monitor the transmitted data to test for its integrity. So a media access protocol that does not involve collision detection had to be chosen. This prompted for the selection of the Token passing as the bus access method. Moreover in a distributed computing environment the occurrence of a collision may prevent the desired operation at the desired time. This can be avoided in a token bus by "cleverly scheduling" all the distributed tasks, since the token availability at each node is deterministic. Thus the Token Bus media access scheme has been selected.

The MPCC was designed with a view to mainly perform control tasks. Thus it uses a simple token passing scheme.

The IEEE 802.4 is a complex scheme and is very difficult to implement as a low cost solution. However the principles of token passing are the same as described in IEEE 802.4. Since the stress in this project is to develop a "node" that can be effectively used for communication and control the entire IEEE 802.4 standard was not implemented. Only the Physical layer and Medium access control layer are implemented. The logical link layer may be implemented in future development.

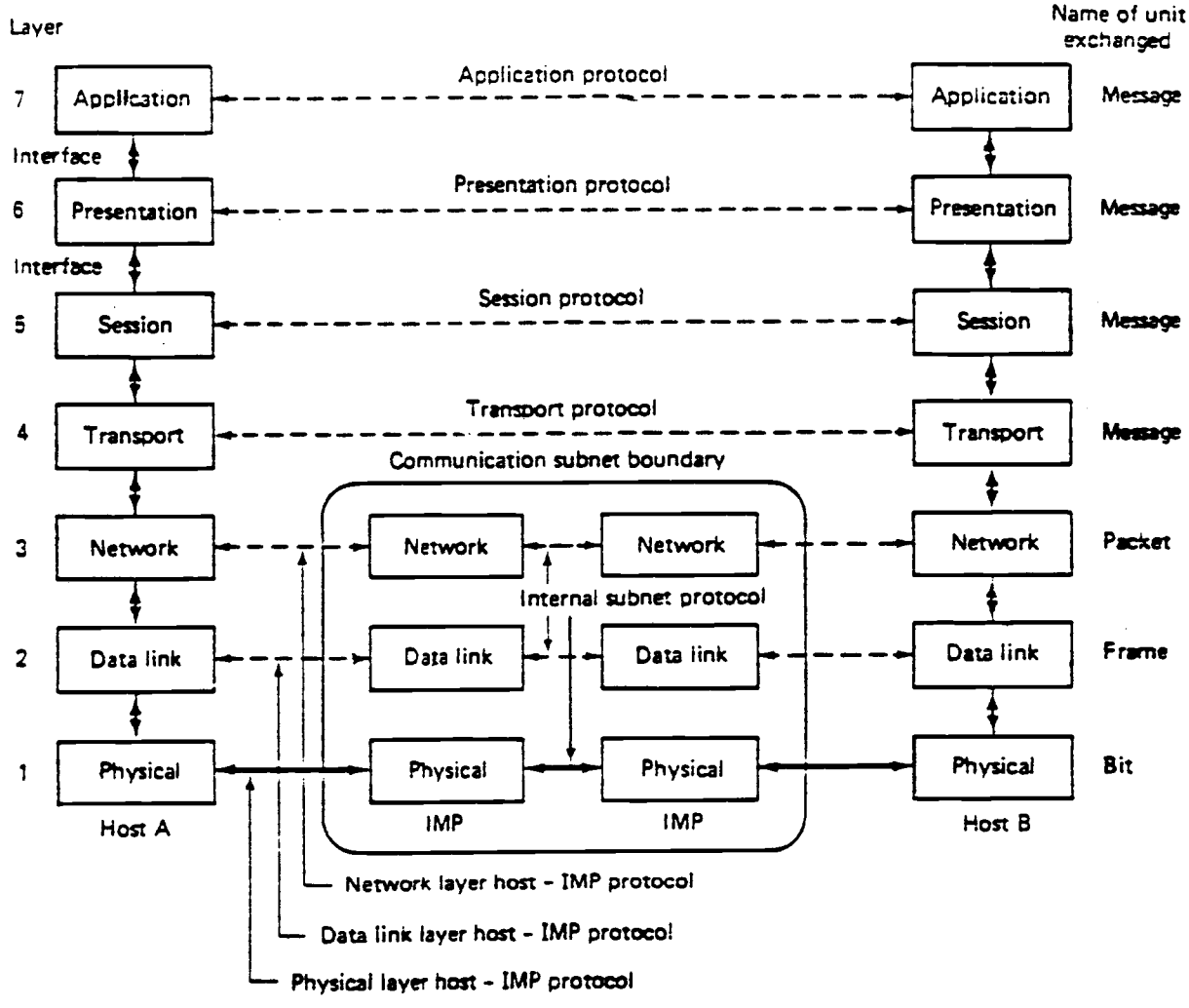


Figure 2.1 The ISO OSI reference model [TANE 81].

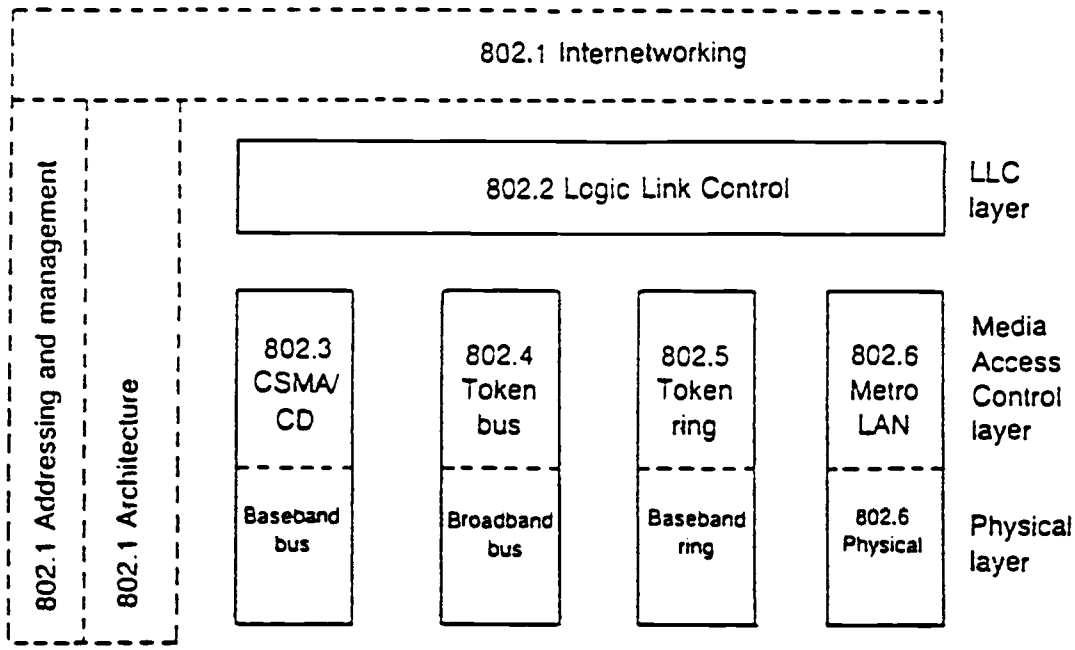


Figure 2.2 IEEE 802 Standards [FRED 86].

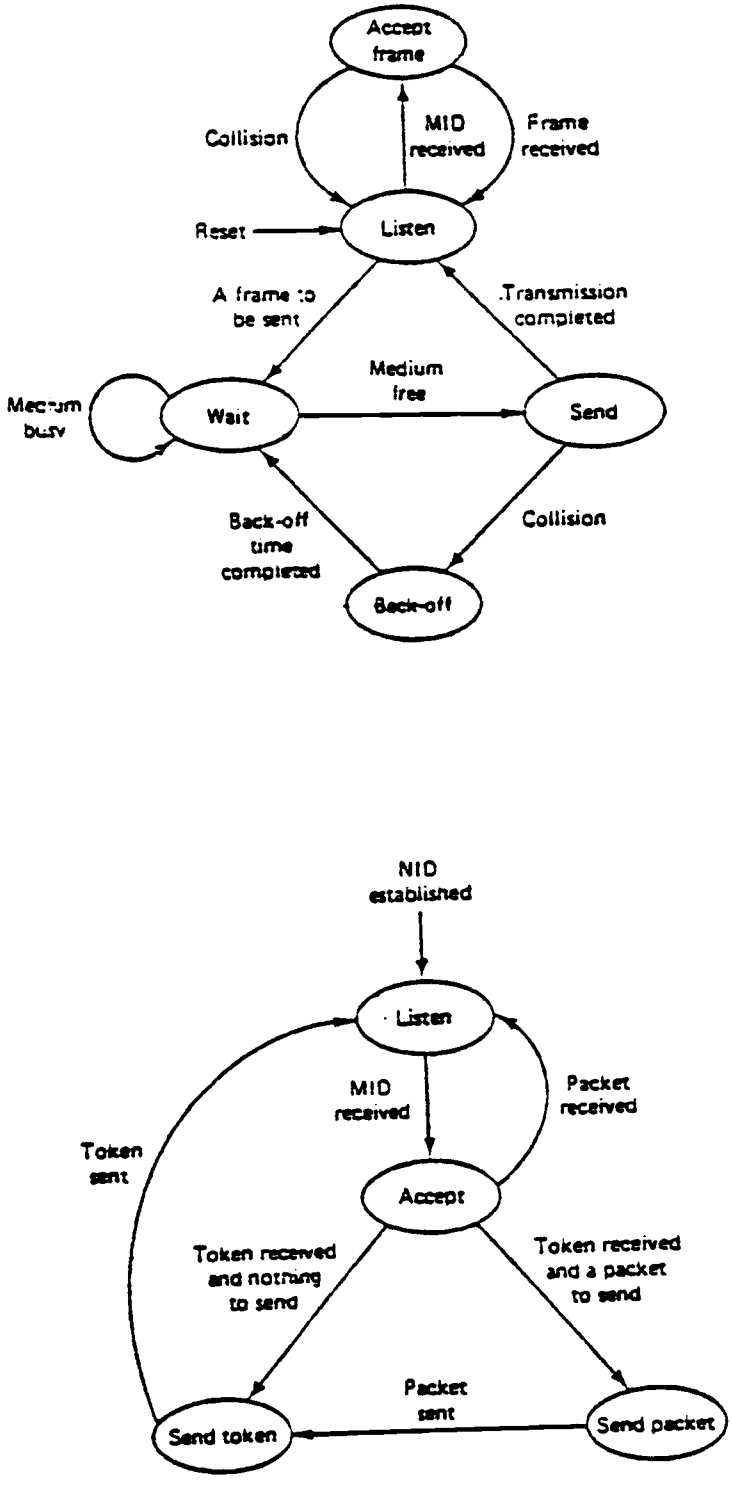


Figure 2.3 CSMA/CD and Token-Bus medium access scheme.

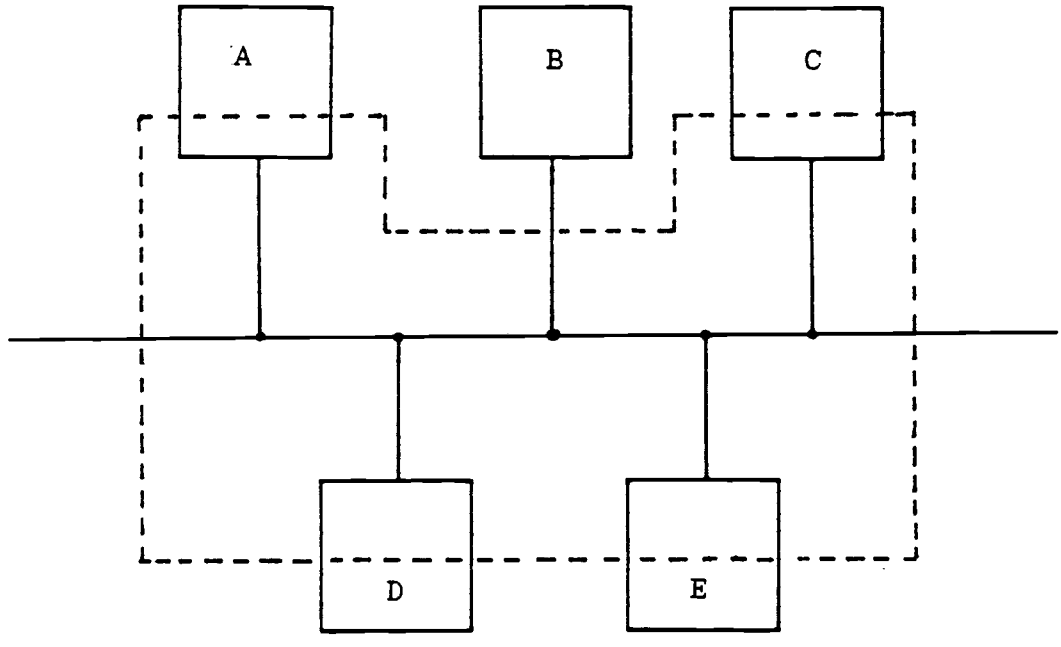


Figure 2.4 Logical Token Ring.
The dotted line indicates the logical ring.
Node B is not a part of the logical ring.

CHAPTER 3

MPCC Structure and Design

This chapter deals with the MPCC system structure. The next chapter deals with the software developed for the MPCC.

3.1 TASKMASTER - An overview:

The TASKMASTER system [HERZ 87] formed the basis on which a great amount of research was carried out in distributed processing at Oregon State University. MPCC is a part of this ongoing research.

The TASKMASTER is a master-slave configured system consisting of one host computer and one or more microcontroller based boards called TASKMASTERs. The system is organized in a daisy-chain structure and is shown in Figure 3.1. Each of the slaves is capable of performing real time control tasks.

The Host can communicate with a TASKMASTER through its serial port. The Host communicates a "task number" to the TASKMASTER. Each TASKMASTER has a task library which resides in its program memory. These task libraries can be modified by the user. Each TASKMASTER has an unique "device number" which is used by the host computer to access a specific TASKMASTER.

The Host computer communicates with the TASKMASTER through command packets. The command packet [HERZ 86] consists of five data fields enclosed in braces, "{ }" which

serve as the start and end delimiter of the packet. The form of a command packet is : { NN P TT Q DD / }.

The individual elements of the command packets are interpreted as follows:

"NN" represents the destination device address. 00 is reserved as an universal address.

"P" represents a pre-execution control character. This single character indicates the method to be used in executing the specified task number. Three pre-execution characters are permissible.

: the task is placed at the end of the task queue. It is executed at conclusion of the execution of all tasks proceeding it on the task queue.

? is a synchronized task and will not start until a synchronizing signal from the host computer.

! the task runs immediately and temporarily suspends an existing task which is currently running.

"TT" is a two character Hexadecimal code which specifies the task to be performed.

"Q" represents post-execution control character. This character indicates the action associated with task termination. 2 possibilities exist:

. the task is discarded after termination.

+ the task is requeued after termination, allowing some or all tasks to be continuously looped for repetitive action.

"DD" is an optional data field. This data field may contain up to five pairs of hexadecimal characters which allow the host to pass data, arguments, or encoded instructions to the TASKMASTER.

"/" is an optional echo request field used for verifying correct reception of the commands.

Besides the above tasks, there are three single character universal commands. These commands are immediately recognized by all TASKMASTERs. Three such commands are possible:

"%" Reset. All units immediately perform a system reset.

"&" Abort. The queued task currently under execution is aborted.

"$" Synchronize. This command serves as a synchronizing command for all synchronized tasks (they have a "?" as their pre-execution character). Upon receipt of "$" these tasks are executed.

The operating system of the TASKMASTER consists of the two major parts, the main routine and the interrupt service routine. The flow chart is shown in Figure 3-2.

3.1.1 Main Routine:

In this routine, the TASKMASTER is initialized. An interrupt window for the serial port interrupt is opened. Execution of the tasks in the task queue commences. If no tasks are present, the TASKMASTER enters an "idle loop". It remains in this condition until called to action by activity on the system bus. In the idle loop, the TASKMASTER continuously examines the task queue for the presence of a task to perform.

3.1.2 Serial Interrupt Service Routine:

The purpose of this part is to receive a character from the host, process the command packet upon receiving a complete packet, put the modified packet on the task queue if a queued or synchronized task, or execute it if an immediate task.

3.2 System Structure:

The daisy-chain structure of TASKMASTER is not desirable in a distributed environment. It is a master-slave configured system and hence is not suitable for LAN operation, since communication among peers is desirable in any LAN. The data communication speeds are very low. In order to overcome these deficiencies COLANs I-IV were developed. MPCC is a part of this ongoing effort.

3.2.1 Physical Structure:

The physical structure of MPCC is shown in Figure 3.3. It consists of two SIBEC II single board computers interfaced through a dual port RAM. One of the SIBEC II board serves all the communication needs of the MPCC. The other serves to execute all the application tasks. The later node runs a modified version of the TASKMASTER. This configuration achieves high performance due to the concurrent operation of the above processors. Besides this, each MPCC consists of an RS-232C serial link with the host computer. An IBM PC was used as the host computer. An RS-485 bus is available for connecting two or more MPCCs.

With the above structure, the following direct communications are possible. Host (IBM PC) can communicate with the communication processor board using the RS-232C link. The communication processor can communicate with the applications processor through the dual port RAM. The prototype area of the SIBEC board contains an INTEL 82530 serial communication controller (SCC) along with an RS-485/RS-232C converter circuit.

3.2.2 System Bus:

The MPCC provides an RS-485 standard bus interface. This bus is used as an interconnection between two MPCCs. The conventional RS-232C standard is used for communication between the SIBEC II and IBM PC. A twisted pair is used as the bus transmission media. The following are some of the advantages with a twisted pair transmission line.

1. Cancellation of noise due to the alternating polarity of the magnetic circuits provided by adjacent twists of the line.
2. Voltage differences between parts of the system appear as common-mode signals and can be rejected by the receiver.
3. Characteristic impedance is fairly uniform, making the line easy to terminate, usually with a 100Ω resistor at the receiver.
4. They are low cost, have long life, and are mechanically very rugged.

The key features of the RS-485 as suggested by Electronic Industries Association (EIA) are summarized below:

1. Driver common-mode output voltage -0.25V to +6V.
2. Receiver common-mode input voltage -7V to +7V.
3. Receiver input impedance is equal to 12 KΩ minimum.
4. Allows up to 32 transmitter and 32 receivers at the same time.
5. Built in protection to prevent failure if two transmitters are turned on at the same time.
6. Supports cable lengths of up to 4000 feet. The maximum data rate is 10 Mbs in a 40 ft length cable. The data rates decrease to 100 Kbs in a 4000 feet cable.

Because of the above advantages, RS-485 is chosen as the system bus. The chip set, SN 75174 and SN 75175 is used to convert TTL level signal to RS-485 and vice versa. Please see Figure 3.4.

3.3 An overview of INTEL's 8344 Architecture:

The 8344 is a serial communication microcontroller known as RUPI (Remote Universal Peripheral Interface) [INTE 86B]. It merges the popular 8051 8-bit microcontroller with an intelligent, high performance, HDLC/SDLC serial communication controller called the serial interface unit (SIU) (Figure 3.5). Thus the chip provides all features of the microcontroller and supports the Synchronous Data Link Control (SDLC) communication protocol. The modes of operation of the RUPI are described in Chapter 4. This dual controller architecture allows complex control and high speed data communication. The 8344 has 256 directly addressable bits and 192-bytes of on-chip data RAM.

3.3.1 The Serial Interface Unit (SIU):

The Serial Interface Unit (SIU) of the RUPI, shown in Figure 3.6, is divided functionally into a Bit Processor (BIP) and a Byte Processor (BYP), each sharing some common timing and control logic.

3.3.2 The Bit Processor (BIP) :

The Bit Processor interfaces the SIU bus with the serial port. The Bit processor is responsible for all the functions needed to Transmit/ Receive a byte of data.

These functions include Shifting, NRZ1 (Non-return to zero inverter) coding, Zero insertion/deletion and extraction of the clock from the data stream using a Digital Phase Locked Loop (DPLL) technique in the self clocked mode. The bit processor also detects flags. GA's (Go Ahead used in loop mode) and aborts (same as GA's). The shut-off detector monitors the receive data stream for a sequence of eight zeros which is a shut-off command for loop mode transmission. The Zero insert/delete circuitry (ZID) inserts zero's after every five consecutive ones.

The DPLL is a free-running four-bit counter running off the 16 X clock. When a transition is detected in the receive data stream, a count is dropped (by suppressing the carry-in) if the current count value is greater than 8. A count is added (by injecting a carry into the second stage rather than the first) if the count is less than 8. No adjustment is made if the transition occurs at the count of 8. In this manner the counter locks in on the point at which transitions in the data stream occur at the count of 8, and a clock pulse is generated when the count over-flows to 0.

3.3.3 The Byte Processor (BYP) :

The Byte Processor (BYP) is controlled by a Finite-State machine. The state of the FSM is kept in an 8-bit register called SIUST (SIU State Counter). This register may be used to manipulate the behavior of the BYP. The BYP also contains the special function register used to

configure the SIU. These registers can be read or written by the CPU over the 8344's internal bus (IB). Simultaneous access to these SFRs by the CPU and SIU is prevented by timing. In particular RAM access is restricted to alternate internal processor cycles for the CPU and SIU, in such a way that collisions do not occur. The transmit and receive states will be explained in Chapter 4.

3.4 SIBEC II:

SIBEC II is a MCS-51 family single board microcontroller made by Binary Technology, Inc., The board was designed for the INTEL 8031 microcontroller. In order to use the 8344 RUPI in place of the 8031 some modifications were made.

The on-chip serial port uses synchronous communication and supports very high data rates (375 Kbps self clocked) and hence cannot be used with the IBM's serial port. Thus an additional serial port in the form of INTEL 82530 (SCC) (Figure 3.7) was provided in the prototype area. The on-chip serial port is used to communicate with the RS-485 bus. An on-board DIP switch is used to set the address of each board. The address setting on the DIP switch can be read through the INTEL 8255 (Programmable peripheral interface).

The lack of on-chip ROM on the 8344 prompted the use of an external program memory. SIBEC II is designed to support a maximum of 48K external memory divided into 5 blocks. Since the READ strobe for these memory blocks is obtained by logically ORing the program store enable (PSEN) and the data

memory strobe (RD), all external memory blocks can support ROMs as well as RAMs.

In the MPCC, the first 8K is reserved as program memory. Memory from 3000H to 7FFFH is reserved for use by the incoming and outgoing messages. The INTEL 82530 SCC has been mapped into 0C000H. Locations 0E000H through 0E3FFH have been reserved for use by a dual port RAM which serves as interface with another SIBEC II board.

Of the five interrupts supported by the 8344, three have been used. External interrupt 1 is used for inter processor communication. Timer0 interrupt is used for detecting an incoming frame. The serial interrupt generated by the serial interface unit (SIU) is used to set up Timer0.

3.5 Inter Processor Communication:

Inter Processor communication (IPC) is used to communicate commands and data between the communication processor and the application processor. The physical connections are shown in Figure 3.8. IPC is achieved through a dual port RAM.

3.5.1 Dual Port RAM:

An AM 2130, 1024x8 Dual-Port static RAM is used for the purpose of IPC. The AM 2130 is a 48 PIN DIP [AMD 87]. It has two independent ports called left and right ports. Each port consists of an 8-bit bidirectional data bus and 10-bit address input bus and the necessary control signals.

The AM 2130 contains on-chip facilities for supporting semaphores. Address 3FEH and 3FFH serve as interrupt

generators. If any data is written at the address 3FFH from the left port, an interrupt signal is asserted on the right port. The interrupt signal is cleared by reading from the right port at the same address. The address 3FFH is similarly used by the right port to assert the interrupt signal for the left port. Each port of the AM 2130 provides a chip enable, an output enable, a port busy and read/write enable control signals.

3.5.2 Operation:

The Dual-Port RAM has been mapped at 0E000H on the SIBEC II memory map. The communication processor, upon receipt of a command, loads the command packet starting at location 0E000H and generates an interrupt (INT1) to the application processor by writing at location 0E3FEH and 0E3FFH. In the interrupt subroutine of the application processor, the application processor decodes the command and clears the interrupt by reading the locations 0E3FEH and 0E3FFH.

Meanwhile, if the communication processor receives another command packet, it monitors its P1.1 (which is connected to the interrupt of the communication processor) to see if the interrupt in progress is cleared or not. If the interrupt is cleared the above process is repeated. No communication between the application processor and the communication processor was implemented and is suggested for future development.

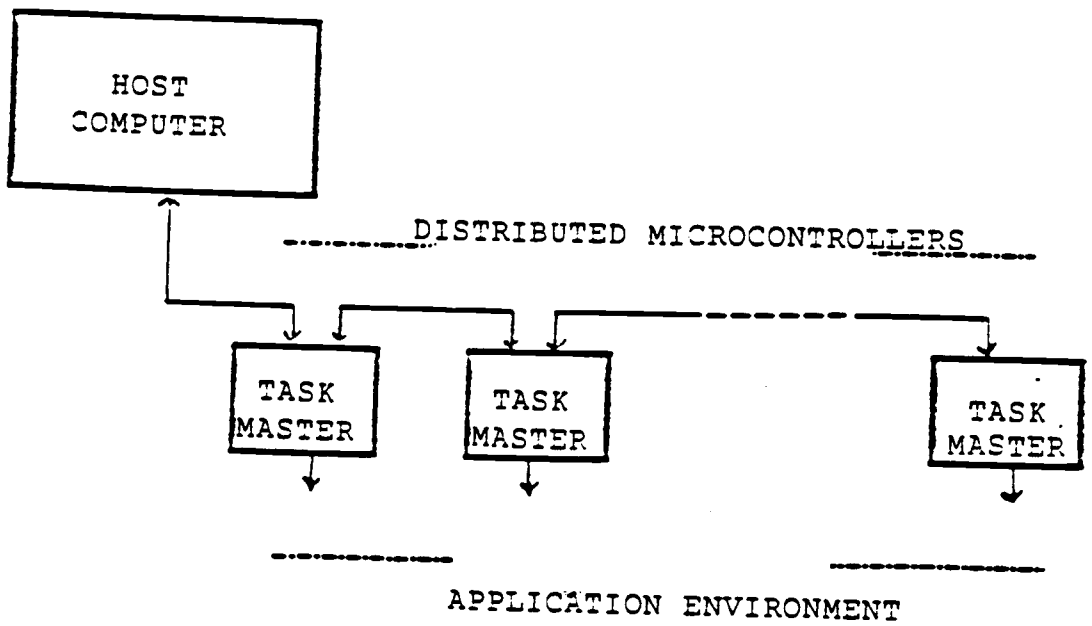


Figure 3.1 TASKMASTER SYSTEM Configuration [HERZ 87].

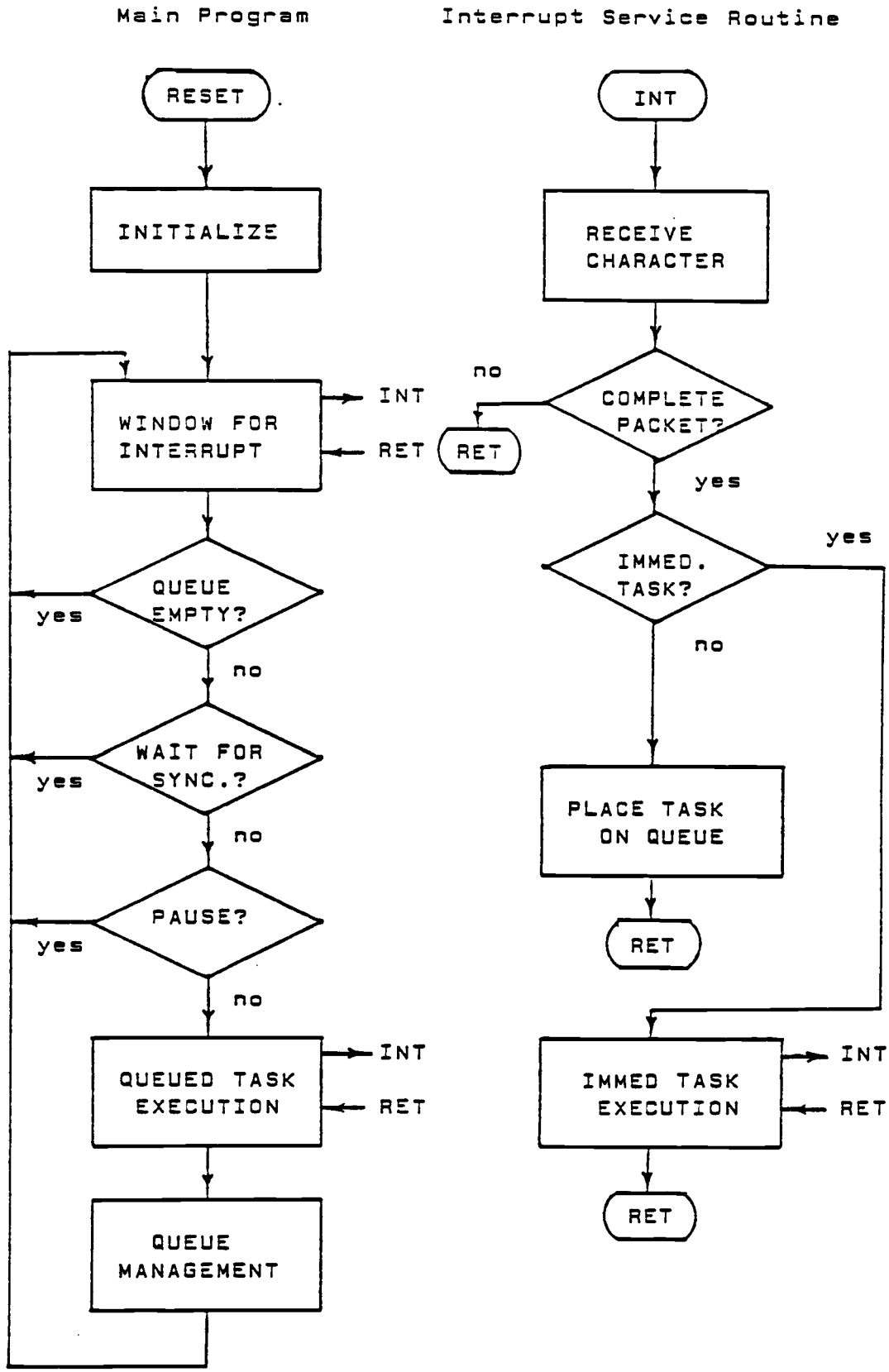


Figure 3.2 TASKMASTER Operating System Flowchart [HERZ 87].

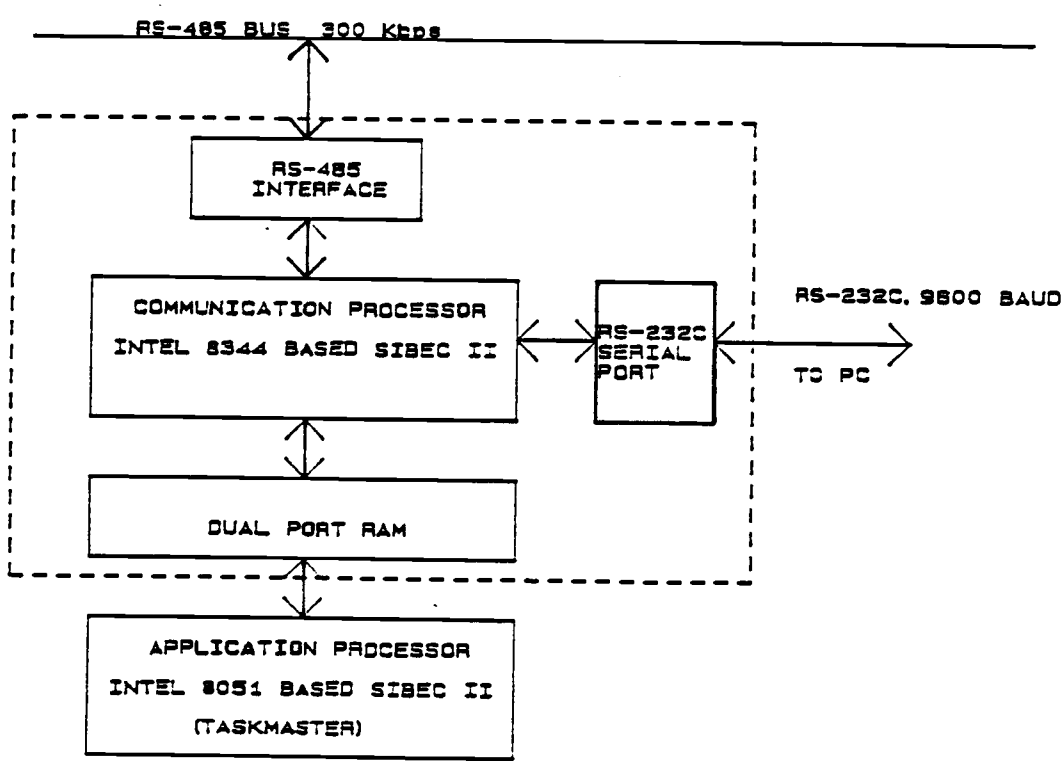


Figure 3.3 Physical Structure of the MPCC.

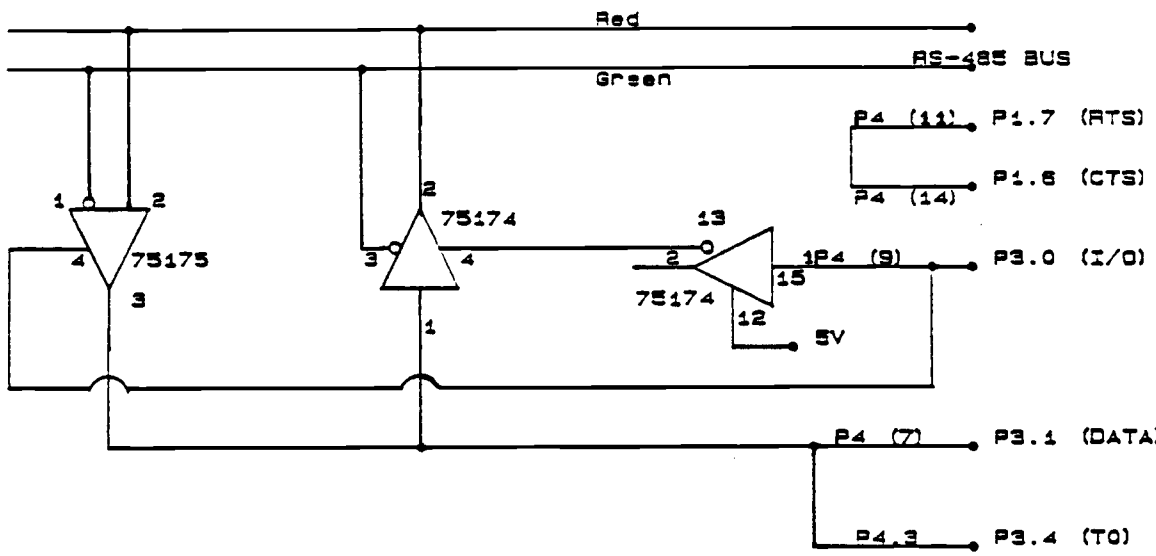


Figure 3.4 RS-485 Bus Interface and 8344 SIU Interface Circuit.

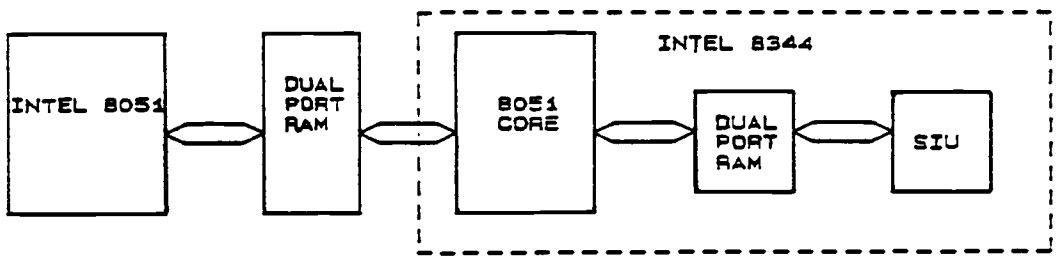


Figure 3.5 Interfacing Of The 8051,8344 and The SIU through Dual Port RAMs.

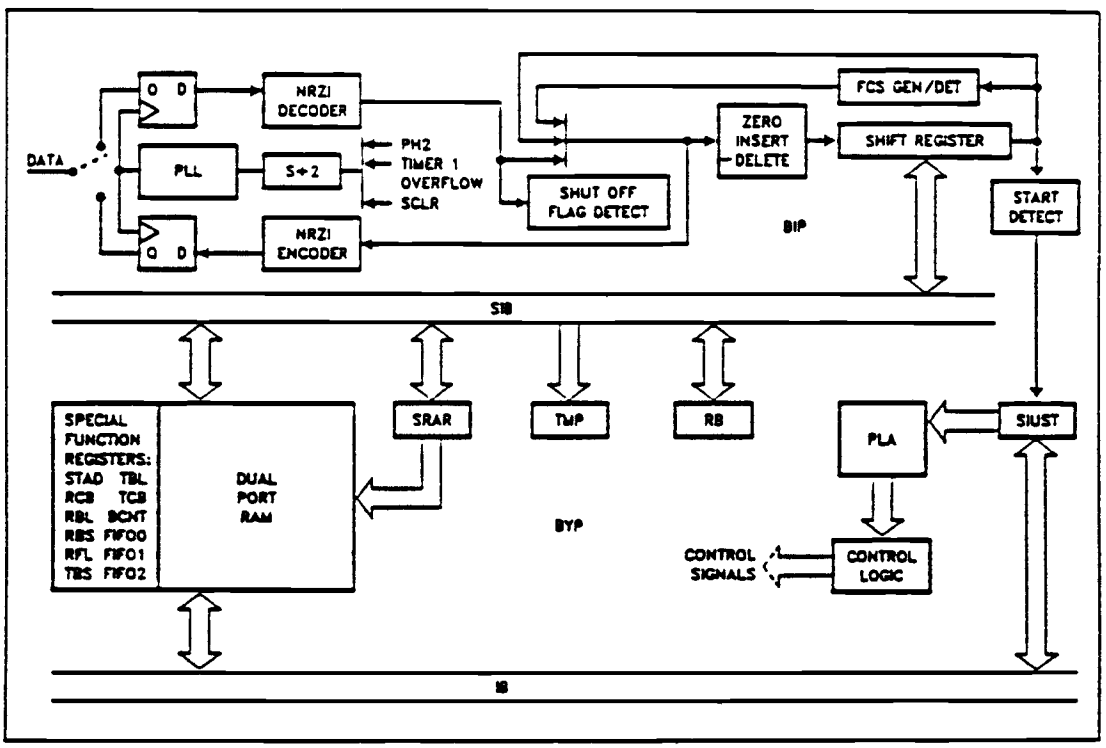


Figure 3.6 SIU Block Diagram [INTE 86A].

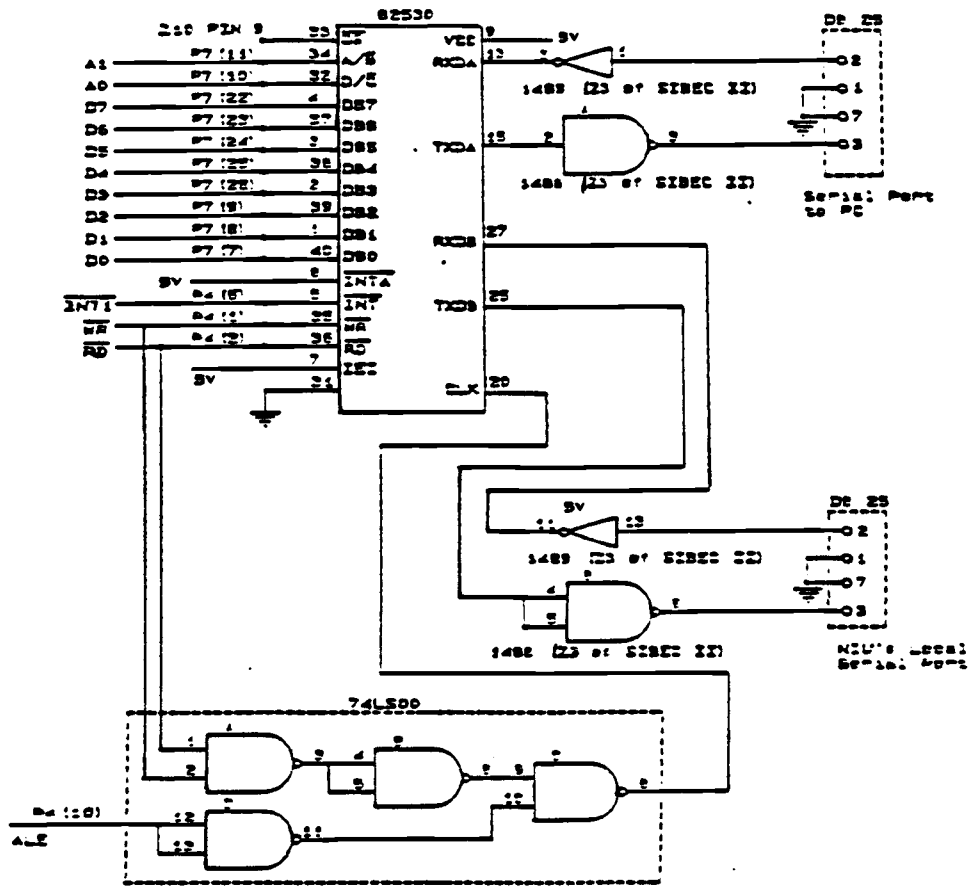


Figure 3.7 MPCC/PC RS-232C Bus Interface.

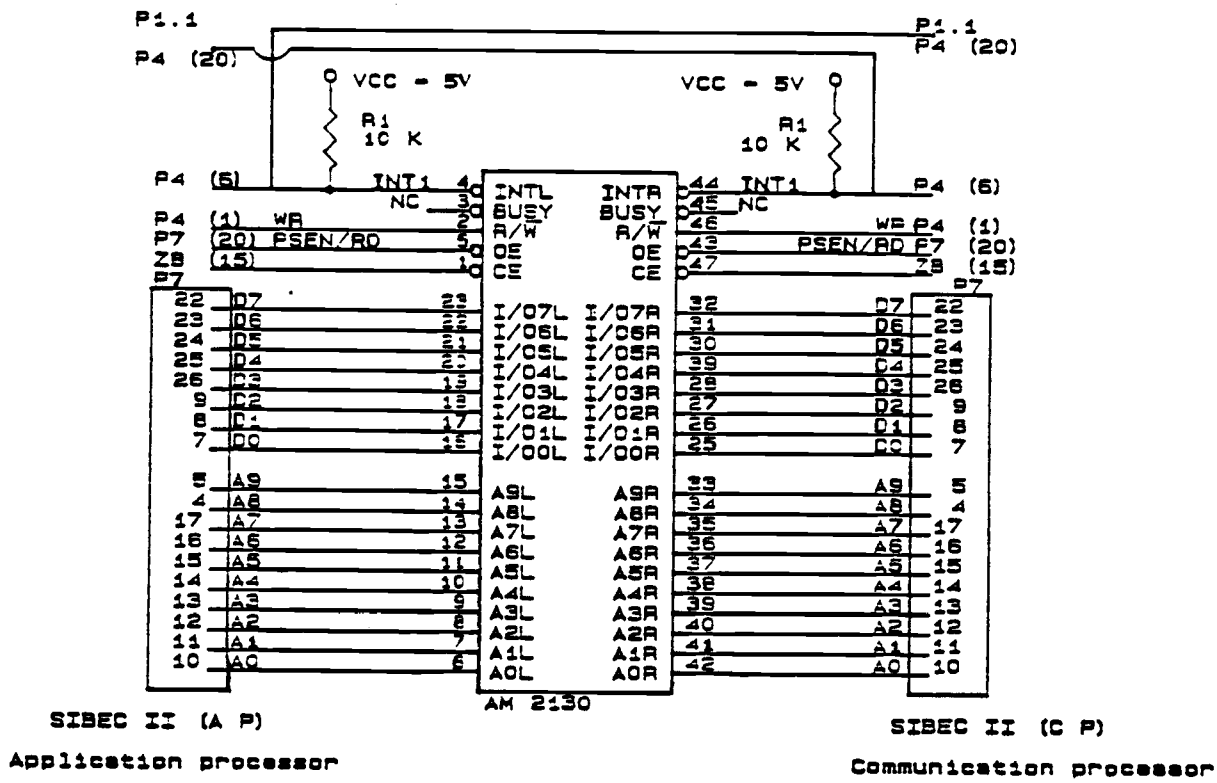


Figure 3.8 Dual Port RAM to Interface The 8051 and 8344.

CHAPTER 4

SOFTWARE FOR THE MPCC

In this chapter, the low-level software design for MPCC is discussed. An overview of the modes of operation of the INTEL 8344 is presented followed by an explanation of the normal and expanded mode of operation. The remaining section deals with the aspects of software developed for this project.

4.1 Modes of operation:

The 8344 can be operated in either the Auto mode or the Flexible mode [INTE 86B]. In the Auto mode the chip responds to many SDLC commands and keeps track of frame sequence numbering automatically without on-chip CPU intervention. In flexible mode, communication tasks are under the control of the CPU.

4.1.1 Auto mode:

In the Auto mode, the 8344 can only be a secondary station operating in the SDLC "Normal Response Mode". The 8344 can not transmit unless it is polled by a primary. In the Auto mode the SIU recognizes and responds to the following commands without the intervention of the on-chip CPU : I (information), RR (receive ready), RNR (receive not ready), and REJ (reject). The SIU also manages Ns and Nr in the control field.

For transmitting a frame, the CPU sets up the SIU and sets the TBF bit. The SIU, upon receipt of a valid poll-frame, transmits the frame. A frame whose poll bit in the control byte is set is a poll frame. After transmission RTS is cleared and the CPU is not interrupted. The reception of frames in Auto mode is similar to that in the Flexible mode. In addition, the SIU sets the RTS bit if the received frame is a poll-frame and increments the Ns (send frame) or Nr (receive frame) counts accordingly.

4.1.2 Flexible mode:

In the Flexible mode all communication are under the control of the CPU. The CPU must handle link access, command recognition/response, acknowledgement, and error recovery. The Flexible mode allows the 8344 to have extended address and control fields, thus providing HDLC support. In the Flexible mode the 8344 can operate as a primary station, since it can transmit without being polled.

For transmitting a frame in the Flexible mode the CPU sets up the SIU and enables it. The SIU formats the data and transmits the desired data in the form of a frame. At the end of the transmission, the SIU clears the RTS bit and interrupt the CPU (SI set), without waiting for a positive acknowledgement. To receive a frame, the CPU enables the SIU by setting the RBE bit. Upon reception of a frame the SIU clears the RBE bit and interrupts the CPU.

MPCC is operated in the flexible mode. This was chosen to give the desired flexibility in the control and address

fields of the frame. This flexibility is important in a distributed environment. Moreover, if communication is to carried out among peers, the flexible mode must be used.

4.2 Methods of operation as related to frame size:

There are two methods of operation relating to frame size:

1. Normal Operation (limited frame size)
2. Expanded Operation (unlimited frame size)

In the normal operation the internal 192 byte RAM is used as the receive and transmit buffer. In this operation the chip supports data rates up to 2.4 Mbps externally clocked and 375 Kbps self-clocked. In Expanded operation the external RAM, in conjunction with the internal RAM is used as the receive and transmit buffer. In expanded operation, the chip supports data rates of 500 Kbps externally clocked and 375 Kbps self clocked. In both cases the SIU can be configured in either the Auto or Flexible mode.

4.2.1 Normal operation:

In the Normal Operation the CPU and the SIU operate in parallel. The SIU handles the serial communication tasks while the CPU processes the on-chip transmit and receive buffer, services interrupt routines, or performs the local real time processing. Figure 4.1 illustrates the flow of data when internal RAM is used as the transmit/receive buffer.

4.2.2 Receive state sequence:

Figure 4.2 shows the receive state sequence. When the an opening flag is detected by the bit processor, the FLAG state is executed. In this state the BYP loads the special RAM (SRAR) register with the contents of the RBS. SRAR thus points to the internal RAM. In the ADDRESS state the received address is compared to the contents of the STAD. If the address doesn't match the reception is aborted else the BYP moves into the CONTROL state. In this state the received byte is moved into the RCB register. Note that the only action taken in this state is that a received byte, processed by the bit processor, is moved to RCB. There is no other hardware task performed. This property is used in the Expanded mode of operation.

The next two states PUSH-1 and PUSH-2, will be executed if the Frame check sequence option is selected. In these two states the first and second bytes of the information field are pushed into the 3-byte FIFO and RFL is set to zero. The 3-byte FIFO is used as a pointer to move received bytes into the internal RAM. The FIFO prevents transfer of CRC bytes and the closing flag to the receive buffer. When the ending of the flag is received, the contents of the FIFO are FLAG, FCS1, and FCS0. In the DMA-LOOP state the byte processor pushes a byte from SR to FIFO0, moves the contents of FIFO2 to the internal RAM addressed by the contents of the SRAR, increments the SRAR and RFL registers. The BOV-LOOP state is executed if there is buffer overrun.

At the end of the reception, if FCS option is used, the closing and the FCS bytes will remain in the 3-byte FIFO. The contents of the RCB register are used to update the NSNR (receive/send count) register. The SIU updates the STS register and sets the serial interrupt.

4.2.3 Transmit state sequence:

Figure 4.3 shows the transmit state sequence. Setting RTS bit puts the SIU in the transmit mode. When the CTS pin goes active, the BYP goes into the START-XMIT state.

In the START-XMIT state the opening flag is moved into the RAM Buffer (RB) register. If the Pre-Frame Sync (PFS) option is selected , the PFS1 and PFS2 states will be executed to transmit either 00H or 55H. In this state the Zero Insertion Circuit (ZID) is turned off while the PFS is sent and then it is turned back on.

In the ADDRESS (SIUST=0A0H) and CONTROL (SIUST=0A8H) states, TCB and the first information byte are loaded into the RB register. Note that in the CONTROL state, none of the registers are incremented, and ZID and FCS GEN/CHK are not turned on or off. The procedures in the DMA-LOOP are similar to that of the DMA-LOOP in the receive state diagram. The FCS1 and FCS2 are executed to transmit the Frame Check Sequence bytes generated by the FCS generator, and the END-FLAG stage is executed to transmit the closing flag.

The two ABORT-SEQUENCE states are executed only if transmission is aborted by the CPU (RTS or TBF bit of the

STS register is cleared) or by the serial data link (CTS signal goes inactive or shut-off occurs in loop mode).

4.3 Expanded operation:

In the Expanded operation the on-chip CPU monitors the SIU state and moves data from/to external buffer to/from the internal RAM and registers while reception/transmission is taking place. If the CPU must service an interrupt during reception or transmission, the chip can shift to normal operation. Figure 4.1 shows the flow of data when external RAM is used.

4.3.1 Transmission and reception in expanded mode:

During transmission or reception of a frame, while the bit processor is processing a byte, the byte processor, after 16 CPU states, is is a standby mode, and the internal registers and the internal bus are not used. The period between each byte boundary, when the byte processor is in the standby mode, can be used to move data from/to external RAM to/from the internal RAM for transmission/reception. The SIUST can be monitored to find the beginning of each byte boundary. By writing into the SIUST the BYP can be forced to perform a specific state.

For transmission, the byte processor is put in the transmit mode and the state of the SIUST is monitored by checking the contents of the SIUST register. When the BYP reaches the desired state and goes into standby, the CPU loads the first byte of the frame into the internal RAM

buffer and moves the SIU into the CONTROL state. This is repeated for every byte until the end of data is reached.

For reception, the SIU is forced to repeat the CONTROL state until the end of the data is received. The end of the data can be a special character or the length of the received and transmitted frame can be decided upon before the transmission/reception of a frame takes place. The above procedure forces the SIU into believing that it is receiving a control byte. The received data can be moved to the external memory from the RCB register.

It is important to note that to write into the SIUST register, the data must be complemented [INTE 86A]. For example, if you wish to write 18H to the register, you should write 0E7H to the SIUST register. The data read from the SIUST is, however 18H.

4.4 Main loop:

The Main loop of the operating system polls the INTEL 82530 (SCC) for data continuously. Upon reset the operating system initializes the 82530 and the 8344. The Timer0 is set up in counter mode. A value of 0F8H is loaded into the TL0. Since the data pin (P3.1) of the 8344 is connected to the timer T0 pin (P3.4) eight transitions on the data line will set the T0 interrupt. The eight transitions are ensured before the reception of a frame by choosing the PFS option. Thus there is an interrupt before the reception of every frame.

Interrupts INT1, SI and ET0 are enabled. A friendly blink from the LED indicates that every thing is working fine and the initialization sequence is complete. The flow chart for the Main loop is given in Figure 4.4 and Figure 4.5.

The users can interact with the MPCC through a set of commands. The command format is " { x " where " x " can be any of the following letters:

"R" Receive. This command puts the MPCC in the receive mode. It expects the data to begin with a "[" and end with a "]" brackets. The received data is put in the transmit buffer for onward transmission.

"B" Blink. A friendly blink of an LED on the MPCC

"C" Command. The MPCC receives a command from the PC and passes it on the application processor for execution. The format of the command is similar to that used by the taskmaster.

"T" Transmit. This command puts the MPCC in the transmit mode. The MPCC executes the transmit subroutine.

"D" Dump. This command dumps the contents of the receive buffer on to the screen of the PC.

4.5 Serial Interrupt routine:

In this subroutine the T0 is re-initialized. 0F8H is loaded into TL0, The TR0 and ET0 bits are set. This routine is also responsible for clearing the SI generated when there

is either a reception or transmission of a frame. The receiver is enabled by setting the RBE bit. The flow chart for this routine is given in Figure 4.6

4.6 TIMER0 Interrupt routine:

This routine is responsible for receiving the data from the bus in the expanded mode. The Timer0 interrupt is set before a frame is received. The SIUST is manipulated to make the SIU believe that it is receiving the control byte until the end of the data mark is received. The received data is transferred to the external buffer while the BYP is in the standby mode. The CPU has to check the first byte with the STAD register to make sure that the received data is for it. It should be noted that all the nodes on the bus will be interrupted when there is activity on the bus. However the checking of the address of the received data takes only a few instruction and should not be a load on the CPU. The flow chart is given in Figure 4.7.

4.7 External Interrupt routine:

This routine is responsible for servicing a command packet available in the Dual-Port RAM. The interrupt can be cleared only by reading data from the locations 0E3FEH and 0E3FFH and returning from the interrupt. The flow chart is given in Figure 4.8.

4.8 The transmit subroutine:

The transmit subroutine is responsible for transmitting the data in the Expanded mode of operation. Before the start of the routine the Timer0 is turned off. This prevents

the SIU from detecting its own PFS and generating the Timer0 interrupt (the P3.11 is used for both receiving and transmitting data). The chip is put in transmit mode by setting RTS and TBF bits. Information is moved from the external RAM into the TBS register. The SIUST is monitored for standby mode and then if there are more information bytes to be transmitted move the BYP is placed into CONTROL state. The above steps are repeated until all the data is transmitted. At the end of the data field the BYP is moved into the standby mode of the CONTROL state and normal operation is resumed. Please see Figure 4.9 for its flow chart.

4.9 Mail transfer:

A simple mail transfer routine has been successfully implemented in the expanded mode. A packet of size greater than 192 bytes was transferred. The "}" symbol is used as the data field delimiter.

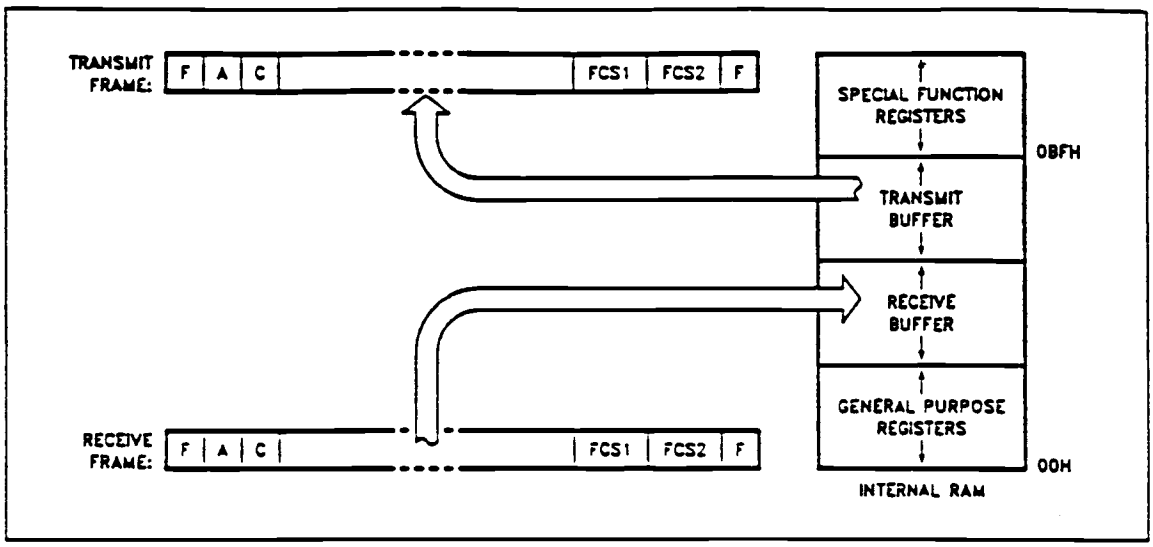
4.10 Modification of the TASKMASTER:

The TASKMASTER running on the application processor has been modified. The original TASKMASTER gets its command through the serial port. This interrupts the 8051 processor at the reception of every character of the command packet. In the modified TASKMASTER the 8051 is interrupted only once per command packet. Since the packet is available in the dual port RAM the TASKMASTER fetches the characters of the command packet successively and services them. However this modification of the TASKMASTER is not complete as some tasks

require to send information to the host. In this setup this is done by passing the information through the communication processor on to the host. However since message passing from the application to the communication processor is not implemented, all the tasks requiring access to the host will not work.

4.11 Task Implementation:

Task 15, a simple task is implemented in the form an LED blink on one of the ports of the microcontroller. The command packet is recognized by the node when it receives a "{" from the host. The node waits for the task number and then executes it. The command end is indicated by "}". The node, on reception of the "{", puts the received command starting from 0E000H in the dual port RAM. After the reception of the "}" the node generates an interrupt to the applications processor by writing into location 0E3FEH and 0E3FFH. This scheme is also used to implement remote tasks.



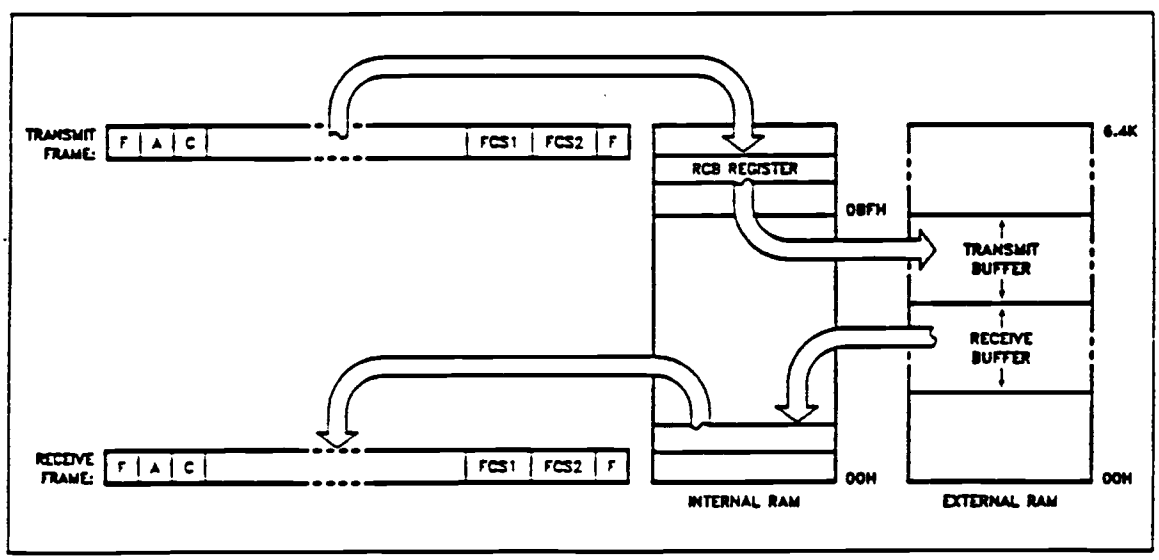


Figure 4.1 Transmission/Reception Data Flow using Internal and External RAM [INTE 86A].

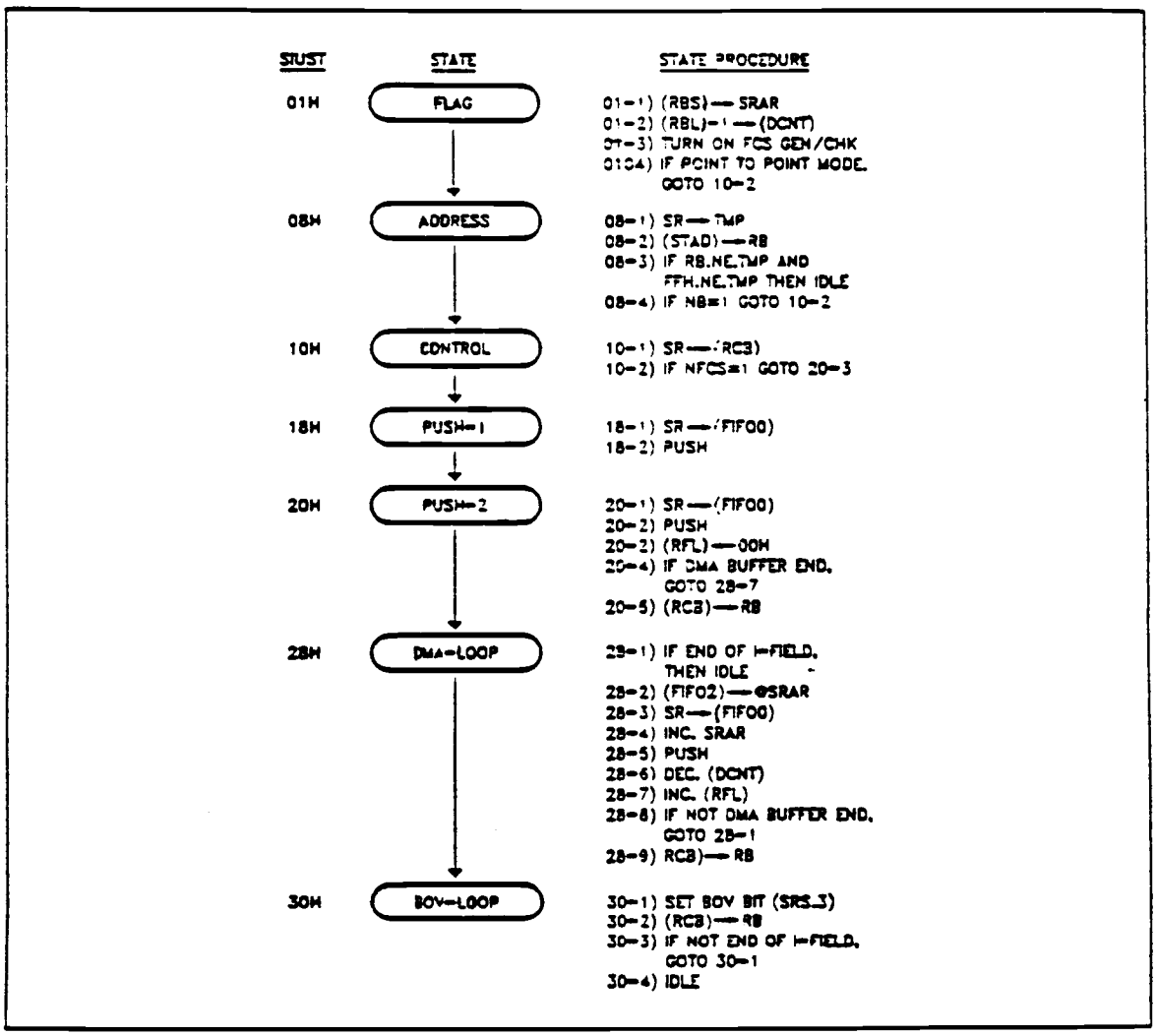


Figure 4.2 Receive State Diagram [INTE 86A].

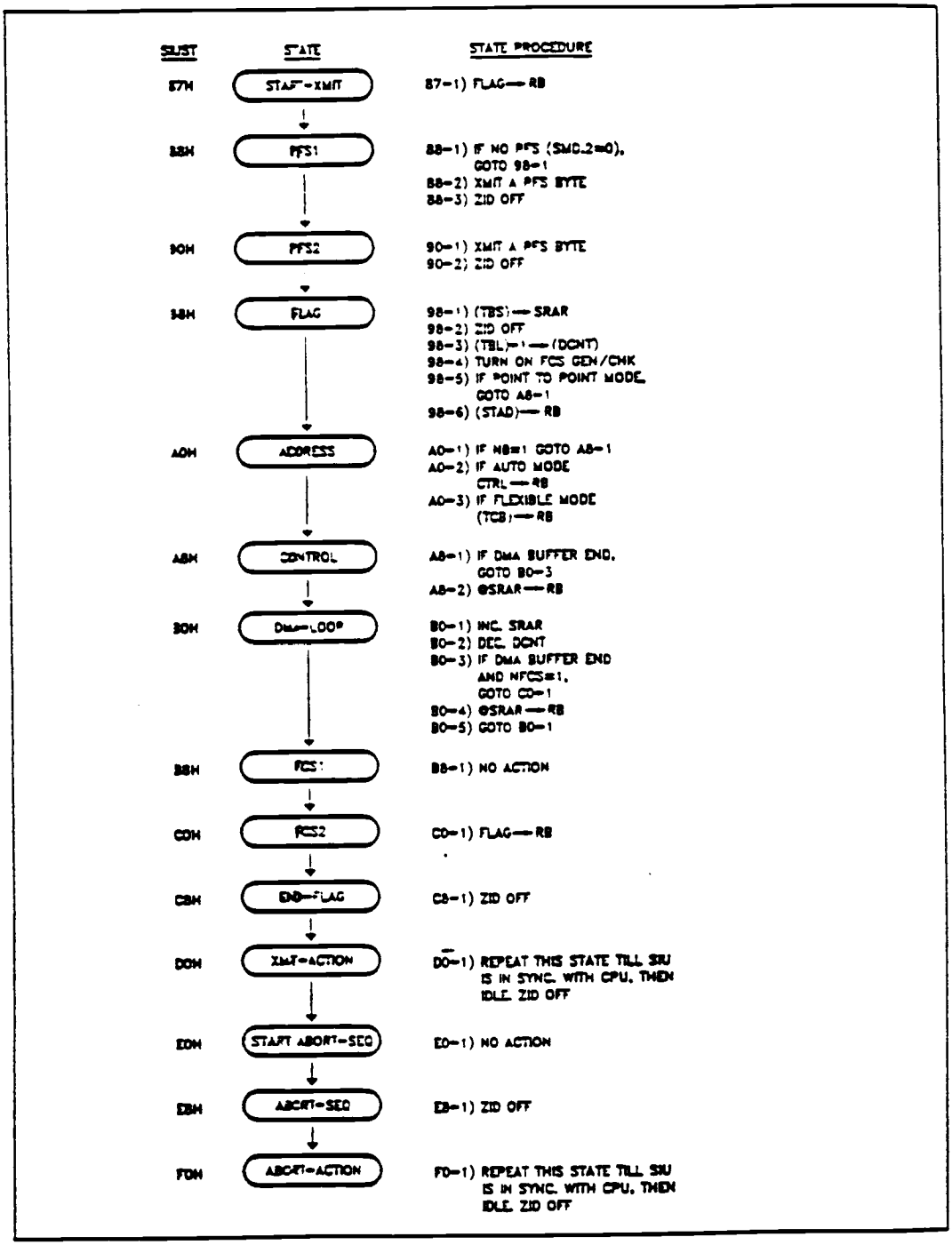


Figure 4.3 Transmit State Diagram [INTE 86A].

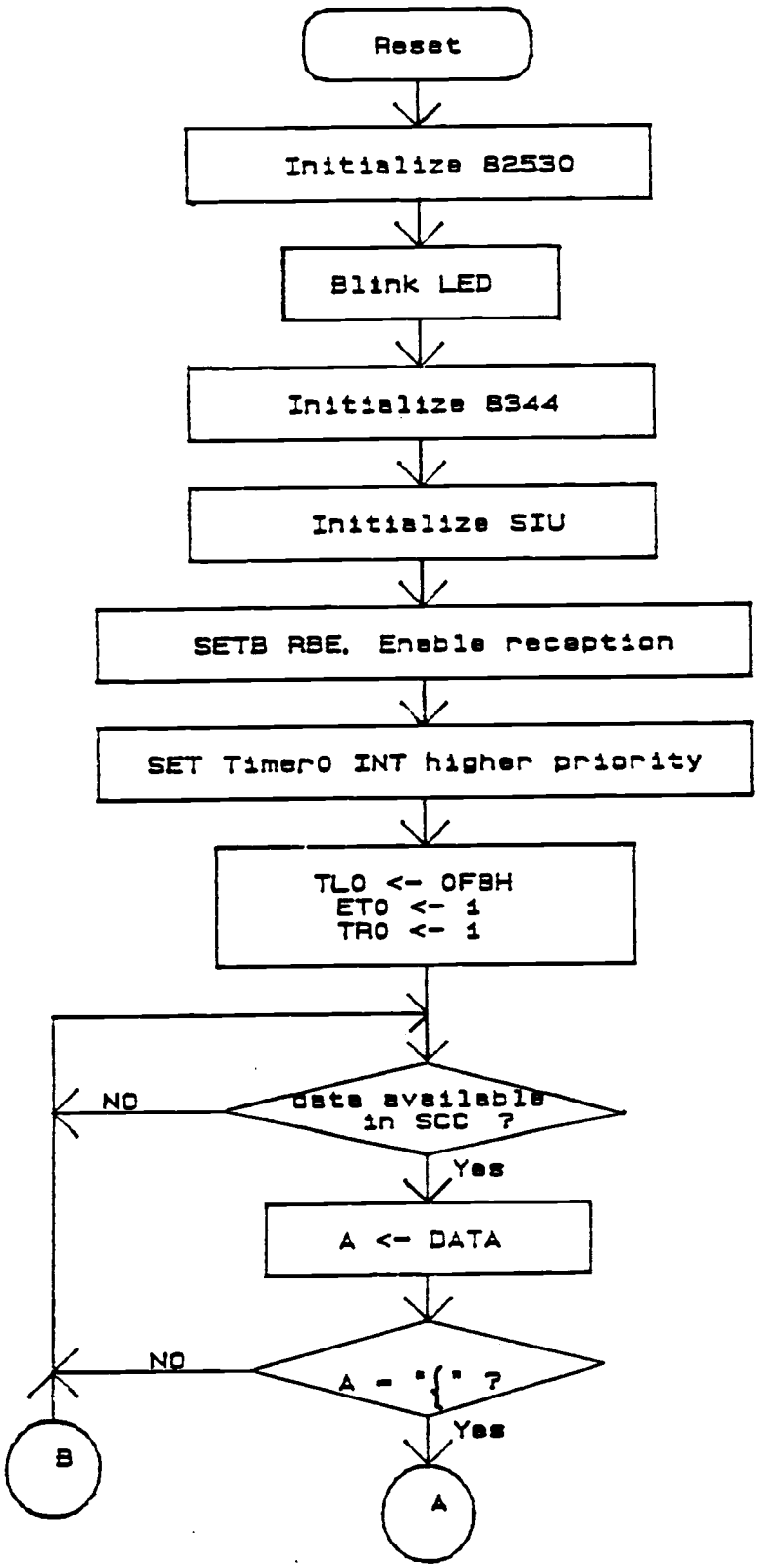


Figure 4.4 Main Loop

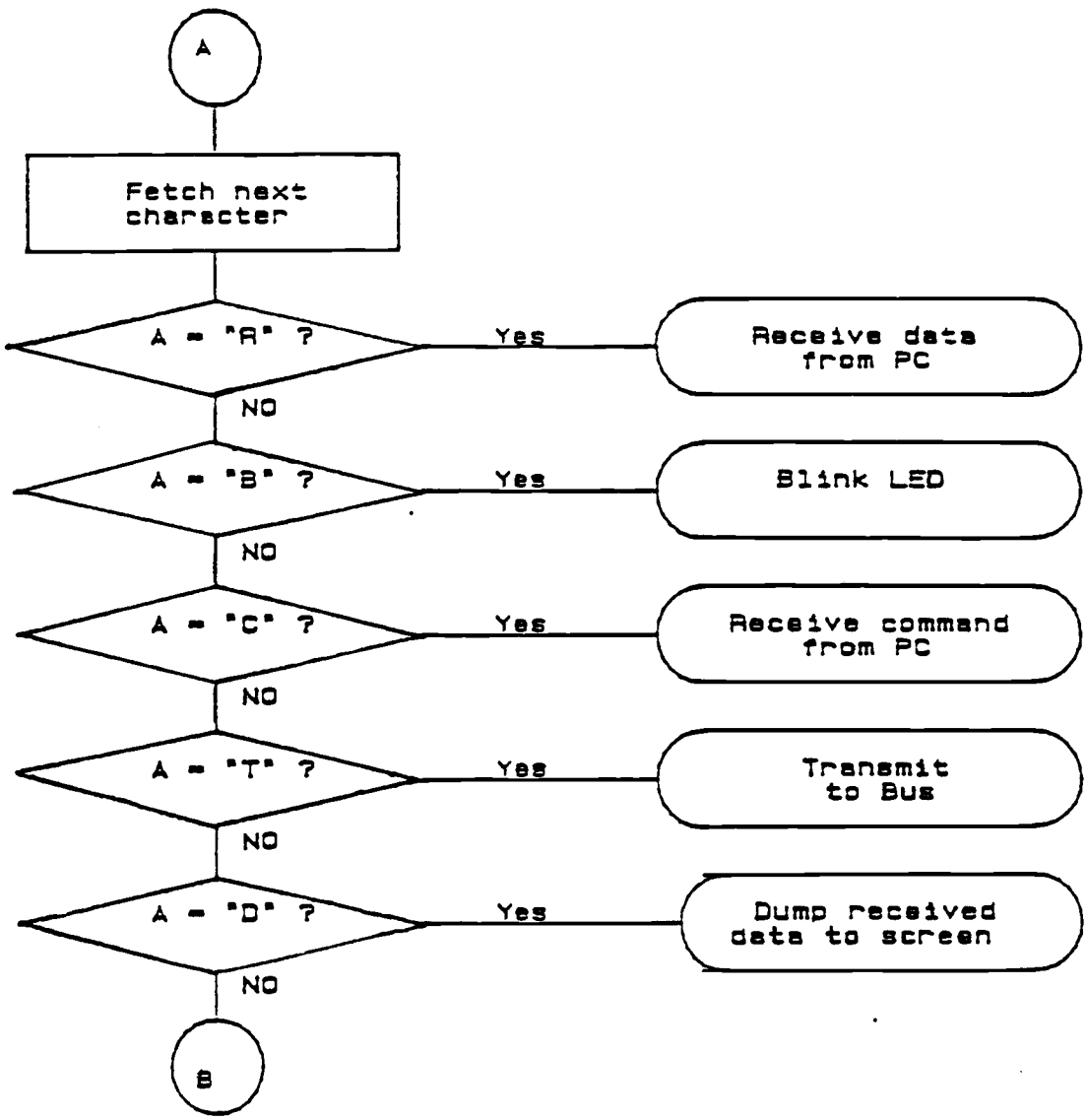


Figure 4.5 Main Loop Continued.

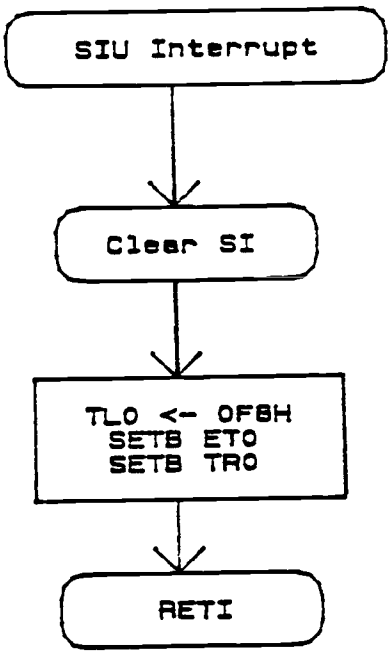


Figure 4.6 Serial Interrupt Routine.

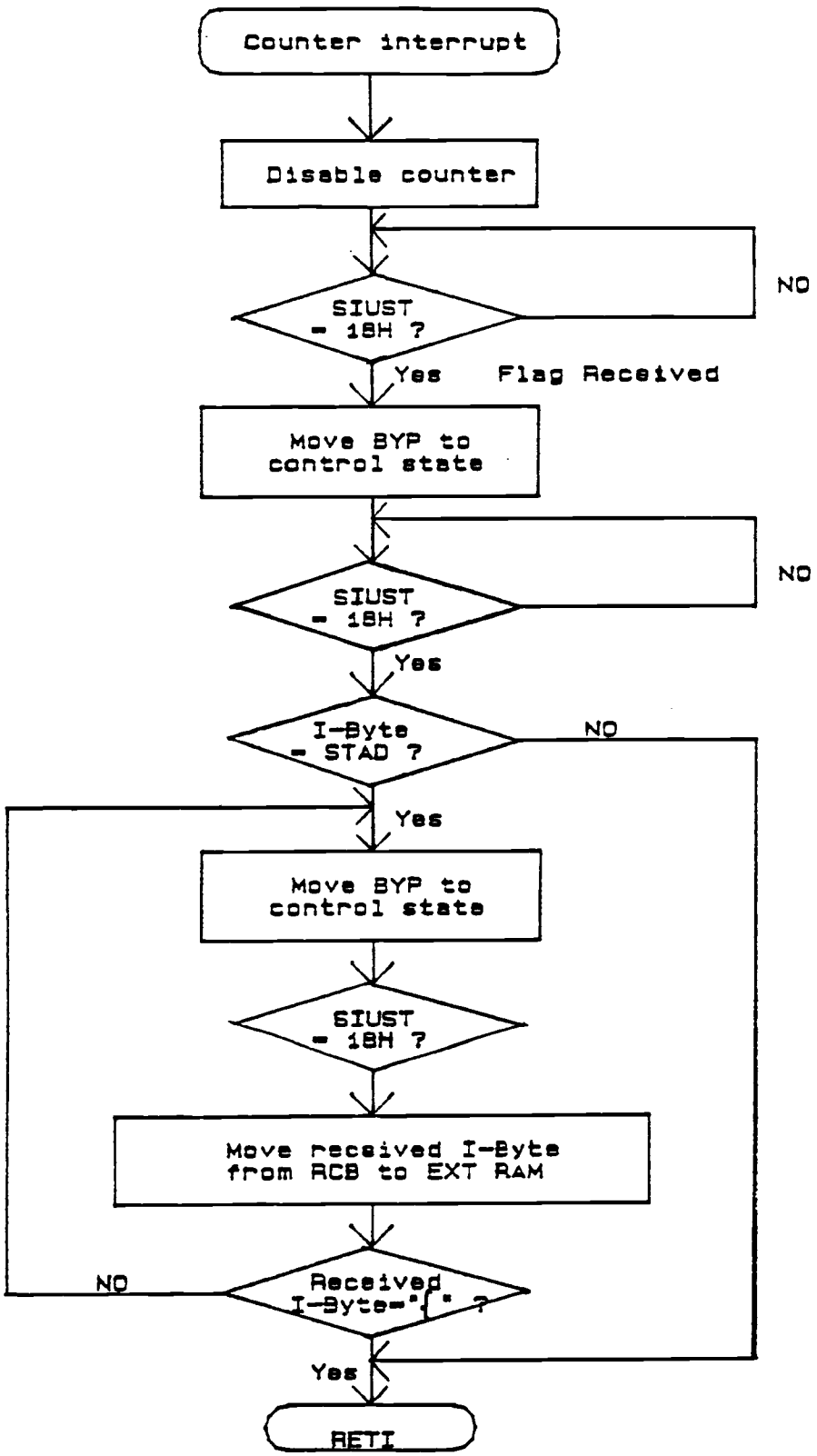


Figure 4.7 Timer0 (Receiver) Interrupt Routine.

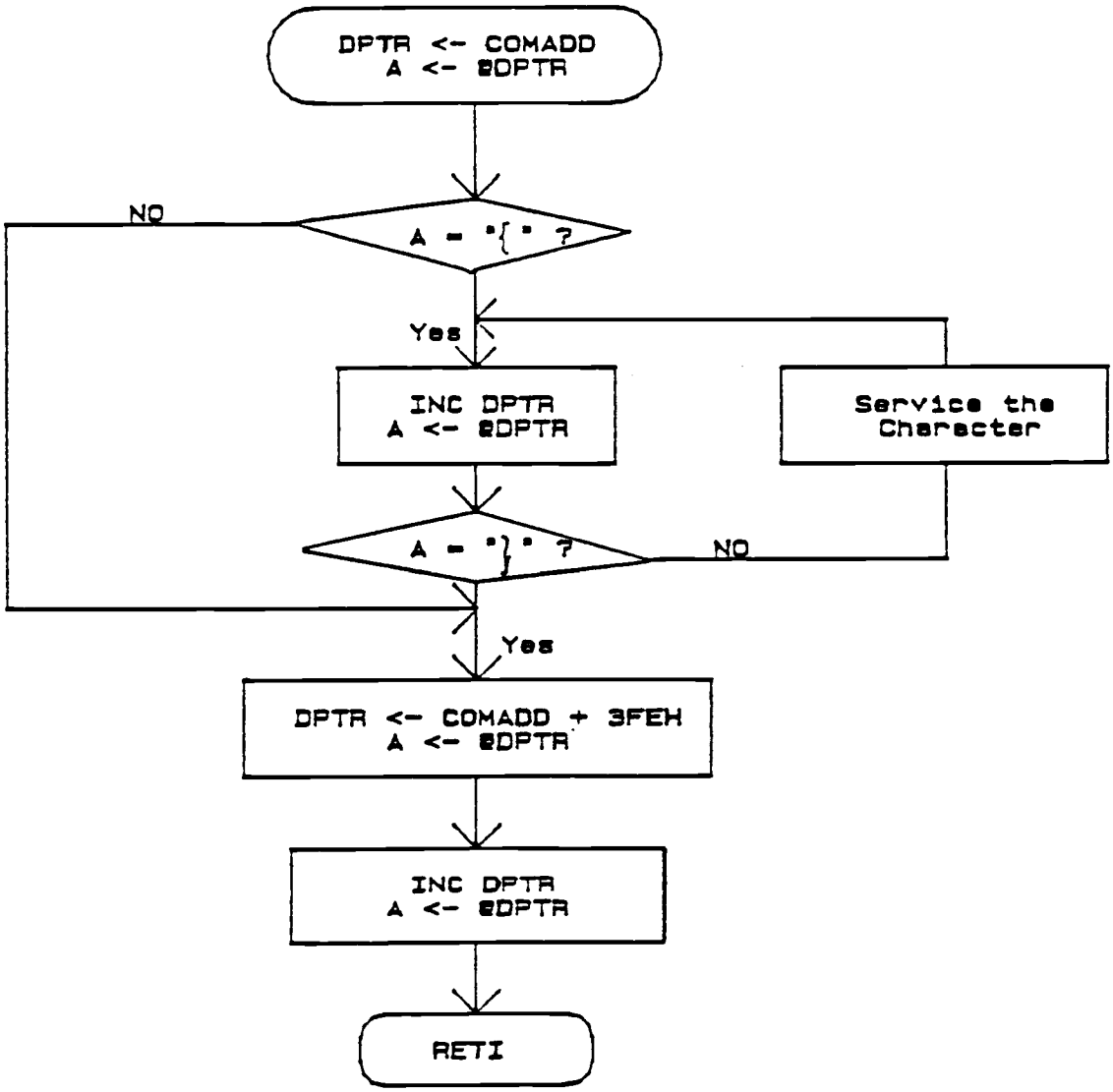


Figure 4.8 External Interrupt Routine (INT1).

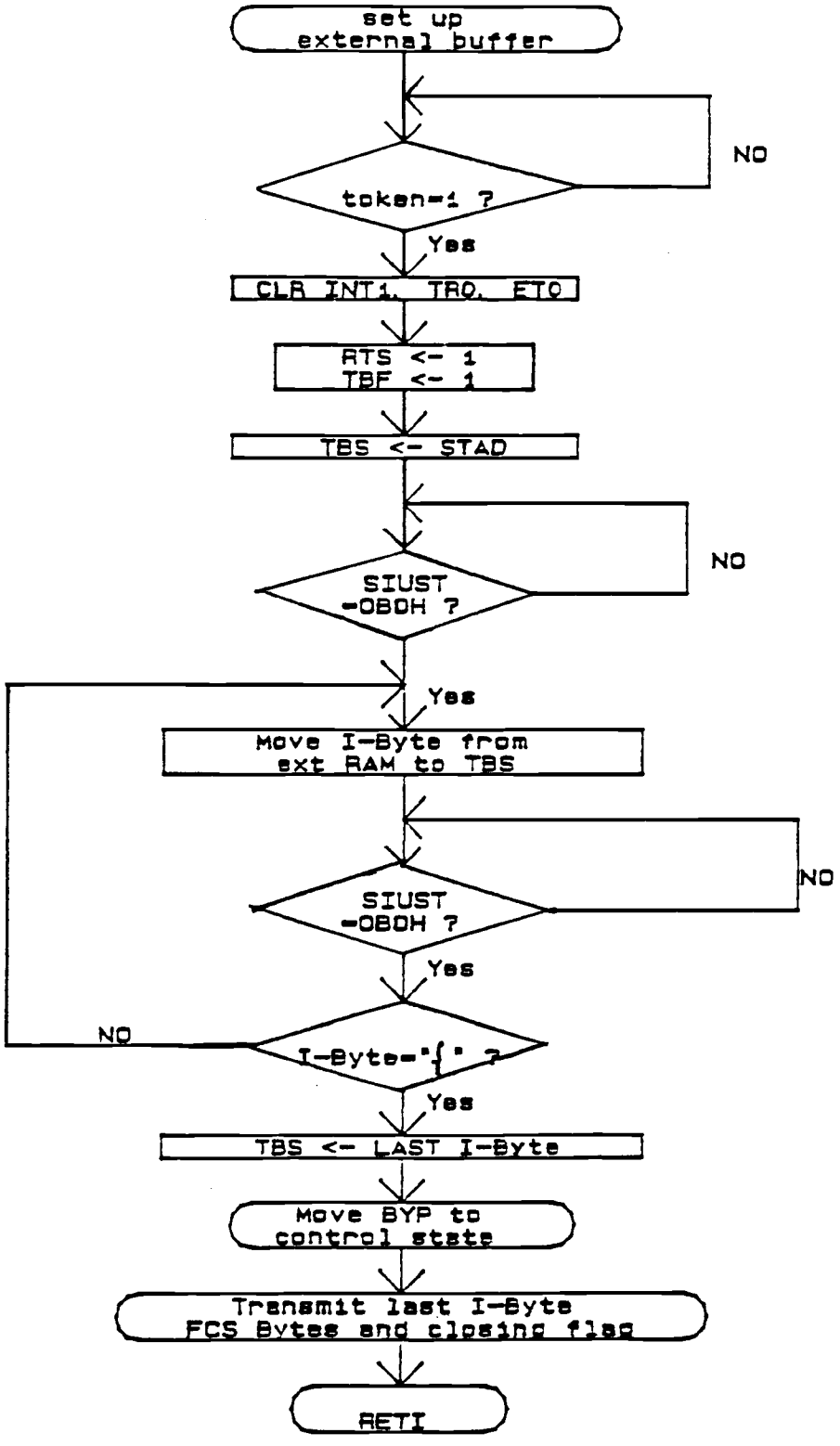


Figure 4.9 Transmit Subroutine.

CHAPTER 5

MDP-44 A Monitor Debugger program for the 8044

A debugger/monitor program has been developed to help develop and debug software for the 8044 microcontroller. The need for developing this program arose from the following facts:

(i) The SIU of the 8044 operates in a synchronous mode while the available software for the IBM PC uses asynchronous communication. Thus it is necessary to use an off chip asynchronous serial port.

(ii) It is necessary to "peep" into the SIU registers so as to understand its functionality.

(iii) The SIU port should be free so that we can experiment with the SIU features and understand its operation fully.

(iv) There is no such program available commercially for the 8044 microcontroller.

Keeping these points in view, the program was developed. The external serial port was provided by the 82530 SSC which is also being used for communication with the PC. The 82530 SSC has been initialized to operate in asynchronous polling, self clocked and at a baud rate of 9600. This baud rate is much higher compared with MDP-51

which operates at 1200 baud rate maximum. MDP-51 is a monitor debugger program used with the INTEL 8051 family of microcontrollers. This program helps us download and run a program from the system RAM. This saves time and effort in programming and erasing EPROMs. Besides this the contents of the registers can be viewed and changed. Thus it creates a ideal environment for software development and debugging.

A routine was written to display the contents of all the SIU registers on the screen. This enables us to understand the operation of SIU. Moreover the values of these registers can be changed using the 'R' command. Also added to the routine is a help command that displays the commands available to the users. A summary of the commands is given below:

? -- Help Menu.

G -- Run the user program.

R -- Display and modify SFR contents.

S -- Dump SIU register to screen.

D -- Dump internal memory to screen.

DX -- Dump external memory to screen.

H -- Hex arithmetic.

E -- Enter.

The "S" command displays the following special function registers of the SIU on the screen along with their current contents.

STAD The station address register

TBS The transmit buffer start address

TBL The transmit buffer length

TCB The transmit control byte

RBS The receive buffer start address

RBL The receive buffer length

RFL The length of the received field

RCB The received control byte register

SIUST The SIU state register

SMD The serial mode register

STS The Status/Command register

NSNR The send/receive count register

Thus this program provides us with ways of debugging the software and understand the operation of the SIU. It should be added that SIUST should be changed with caution as it is the state register for the SIU and may put the SIU in an undesirable state.

CHAPTER 6

CONCLUSION AND SUGGESTIONS

In this project a high speed, reliable, low cost node for communication and control was developed. TASKMASTER was modified to some extent to receive and service tasks stored in the dual port RAM instead of from the serial port of the INTEL 8051 microcontroller. The INTEL 8344 was successfully used in the expanded mode. The receive and transmit buffers length can theoretically be up to 64K. Data communication speeds of up to 300 Kbps have been achieved in contrast to 9600 baud achieved in COLANs I - IV. There has been a great reduction in the communication software due to the availability of the intelligent SIU on-board the INTEL 8344. Simple tasks were executed both from the local and the remote hosts. Mail exchange was also implemented and tested using the expanded mode of operation. Error detection was improved due to presence of the on-chip CRC-CCITT circuitry. MDP-44 a monitor debugger program was developed as an aid for future developers who wish to work on the INTEL 8044 microcontroller.

The following suggestions are made for future development:

6.1 Increasing the data rates:

The MPCCs can exchange information at 300 Kbps. This can be improved up to 2.4 Mbps by using an external clock.

It may be noted that the RS-485 can support data rates of up to 10 Mbps.

6.2 Completing the modification of TASKMASTER:

In the current version of the MPCC the TASKMASTER running on the application processor can not "talk" with the application processor. Since some tasks in the TASKMASTER task library exchange information with the host computer it is important that both the communication and the application processor can "talk" with each other. This can be easily achieved.

6.3 8344 and the Token Bus:

The Auto mode of operation in the 8344 can be used to advantage in the logical ring initializing process. This can be implemented by operating the token holder as the master and the rest as slaves in Auto mode. This allows automatic generation of response from the slave nodes to a poll frame from the master without CPU intervention.

6.4 Completion of the Task library:

Additional tasks are to be written to improve queue management and improve mail handling techniques. Tasks may be written to communicate results from one distributed process on one node to another node without host computer intervention. This would allow complicated distributed tasks to be executed.

BIBLIOGRAPHY

AMD 87 Advanced Micro Devices. " AM2130/AM2140, 1024 X 8 Dual-Port Static Random Access Memories", Preliminary Information leaflet, 1987.

DIGI 84 Young SOHN & Charles Gopen. " Networking with 8044 ", Digital design, May 1984, pp. 136-137.

EUM 87 Eum, D. "COLAN III, A Control Oriented LAN using CSMA/CD protocol". Unpublished master's thesis, Oregon State University, Corvallis, Oregon, December 1986.

FRED 86 Fred Jennings. Practical Data Communications Modems, Networks and Protocols. Blackwell 1986.

HERZ 86 J.H. Herzog. "TASKMASTER Operating Manual", 1986. (unpublished).

HERZ 87 J.H. Herzog. "A Design Methodology for Distributed Microprocessors in Real Time Control Applications" Paper presented at Second International Conference on Computers and Applications, Beijing, People's Republic of China, June 24-26, 1987.

IEEE 82 "IEEE 802 Local Area Network Standard Draft. IEEE Standard 802.4, Token-Passing Bus Access Method and Physical Layer Specifications, Draft D". December 1982.

IEEE 83 "IEEE 802 Local Area Network Standard Draft. IEEE Standard 802.3, CSMA/CD Access Method and Physical Layer Specifications, Draft D". December 1983.

INTE 86A Intel Corporation. Microcontroller Handbook, 1986.

INTE 86B Parviz Khodadadi, "Flexibility in Frame Size with the 8044", chapter 4, Intel Application note, August 1986.

KAO 87 Kao S. "Design of COLAN II, A Control Oriented Local Area Network". Unpublished master's thesis, Oregon State University, Corvallis, Oregon, September 1987.

KAI 84 Hwang,K., and F.A. Briggs. Parallel computer Architecture. Mc Graw-Hill, New York, 1984.

LESZ 87 Leszek Reiss. Introduction to Local Area Networks with microcomputer experiments, Prentice-Hall Inc., 1987.

STAL 84 William Stalling. Local Networks. Macmillan Publishing Company, 1984.

TANE 81 A. S. Tannenbaum. Computer Networks. Prentice -Hall Inc., 1981.

THYE 88 Yong Thye. "COLAN IV, A Local Area Network for Communications and Control". Unpublished master's thesis, Oregon State University, Corvallis, Oregon, February 1988.

ZHEN 86 Y.P.Zheng,"A Simple Local Area Network, COLAN (Control Oriented Local Area Network)." Unpublished master's thesis, Oregon State University, Corvallis, Oregon, December 1986.