National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

A CSMA/CD PORTABLE DATA SYSTEM USING REED-SOLOMON CODES

Chivinh Phan

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

September 1992

ISBN 0-315-97600-4

Canada

ABSTRACT

A CSMA/CD PORTABLE DATA SYSTEM USING REED-SOLOMON CODES

Chivinh Phan

The current proliferation of personal computers, data terminals and integrated computer aided manufacturing and control systems creates an ever increasing demand of indoor wireless communications for portable/mobile data terminals.

This thesis presents a low-cost Carrier-Sense-Multiple-Access/ Collision Detection (CSMA/CD) portable data system for indoor applications. An adaptive error control coding scheme using Reed-Solomon (RS) codes is introduced to combat random and burst errors occurring in an indoor wireless environment. It is indicated that this adaptive coding scheme is more appropriate to the varying and almost unpredictable behavior of indoor wireless channels.

The performance of the portable data communication systems using CSMA/CD is analyzed. The data traffic and acknowledgment response traffic are combined into one channel. The average throughput and packet delay are derived under a Poisson finite user population packet arrival model. The effects of channel behaviour are also considered. The analytical results are verified by experimental ones. An experimental CSMA/CD portable data system using a newly developed programmable RS(31,K) codec (encoder/decoder) chip set is described.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Tho Le-Ngoc for his patient guidance and support, to Dr. S. Hamidreza Jamali, Dr. A. Benyamin-Seeyar and Mr. Ken Deegan for their suggestions and valuable comments, to all my colleagues for helping me in this challenging research. I would also like to thank National Science and Engineering Research Council of Canada (NSERC), to Formation des Chercheurs et aide a la Recherche (FCAR) for providing me with post-graduate scholarships, and the Department of Electrical and Computer Engineering, Concordia University for providing me the Teaching Fellowships. Finally, many thanks to my parents for their understanding and encouragement.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

a	Poisson arrival rate.
ASK	Amplitude Shift Keying.
$\mathbf{b}_k$	Binary sequence of the ASK transmitter.
BCC	Binary Redundant Cycle Checksum.
$\mathbf{C}_i$	Codeword.
CSMA/CD *with ACK*	Carrier Sense Multiple Access and Collision Detection with Acknowledgment.
D	Channel delay.
e	Number of errors in the receiving codeword.
FEC	Forward Error Coding.
$\mathbf{f}_b$	baud rate.
$\mathbf{f}_{ij}^{bT}$	the probability that the state of channel changes from i to j and that a retransmission packet is unsuccessfully transmitted during a transmission time due to the error in the channel.
$\mathbf{f}_{ij}^{nT}$	The probability that the state of channel changes from i to j and that a new packet is unsuccessfully transmitted during a transmission time due to the error in the channel.
$\mathbf{ftt}_{ij}^{nTc}$	The probability that the state of channel changes from i to j and that a new packet is unsuccessfully transmitted during a transmission time due to the collision with another new packet.
$\mathbf{ftb}_{ij}^{nTc}$	The probability that the state of channel changes from i to j and that a new packet is unsuccessfully transmitted during a transmission time due to the collision with another retransmission packet.
$\mathbf{fbt}_{ij}^{nTc}$	The probability that the state of channel changes from i to j and that a retransmission packet is unsuccessfully transmitted during a transmission time due to the collision with another new packet.
$\mathbf{fbb}_{ij}^{nTc}$	The probability that the state of channel changes from i to j and that a retransmission packet is unsuccessfully transmitted during a transmission time due to the collision with another retransmission packet.

G	Channel traffic.
$\mathbf{I}_i^*(s)$	Laplace transform of the probability distribution function of the idle period.
K	The length of information byte in the codeword.
LSPF	Laplace Transform of Probability distribution Function.
M	Number of users in the system.
$\mathbf{N}_c$	Number of character per packet.
$\mathbf{N}_b$	The total number of bits per character.
$\bar{\mathbf{n}}$	Average channel backlog.
$\mathbf{n}(t)$	The number of backlogged users at time t.
N	The length of the codeword C_i.
$\mathbf{N}_T$	Total number of packets transmitted to obtain a successful transaction.
$\mathbf{N}_{TT}$	Total number of packet transmitted during the entire operation.
$\mathbf{N}_{NT}$	Total number of successful packet transmitted during the entire operation.
$\mathbf{N}_o$	The number of backlogged users at the beginning of a cycle.
$\mathbf{N}_B$	The number of backlogged users at the beginning of a busy period.
$\mathbf{N}_C$	The number of backlogged users immediately after the first colliding user starts transmission.
$\mathbf{N}_\tau$	The number of backlogged users at τ units of time after the beginning of a busy period.
$\mathbf{N}_E$	The number of backlogged users at the end of a cycle.
$\mathbf{N}_{Tr}$	The number of unsuccessful packets due to the channel noise.
$\mathbf{N}_{Tc}$	The number of unsuccessful packets due to the channel collision.
OOK	ON-OFF keying type.
$\boldsymbol{\pi}_i$	The steady state probability of system in state i.
$\mathbf{P}_{ij}$	The steady state transition probability that the channel is moved from state i to state j.
$\mathbf{P}_N(t)$	The probability that the transmission of a new packet is started at the beginning of the busy interval given that the state of system at the beginning of the cycle is i.

$P_{\bar{s}}(t)$	The probability that no station senses the channel during the propagation delay.
$P_{acc}(t)$	The probability that new packets are generated during this transmission time.
Q	Successful Transmission Probability.
RDN	Read signal.
RS31KED	Reed-Solomon (31,K) encoder/decoder.
SRR	Super-Regenerative Receiver.
SOT	Start of Text.
sc_{ij}^{n}	The probability that the state of the channel changes from i to j and that a new packet is successfully transmitted during a transmission time.
sc_{ij}^{b}	The probability that the channel state changes from i to j and a retransmission packet is successfully transmitted during the transmission time.
T	The packet transmission time.
T_c	Collision time.
T_d	Time out interval.
T_M	Measurement time.
T_i	Data packet transmission time.
T_w	Acknowledgment packet transmission time.
T_u	Transmission period of the unsuccessful packets.
r	Mean of retransmission delay.
ρ	Number of erasures in the receiving codeword.
S	Channel throughput.
WRN	Write signal.
$w^{*(1)}(s)$	The LSPF of the packet delay in case (1).
$w_j^{*(1)}(s)$	The LSPF of the packet delay in subcase (1).
$w_{ij}^{*(2)}(s)$	The LSPF of the packet delay in subcase (2).
$w_j^{*(3)}(s)$	The LSPF of the packet delay in subcase (3).
$w_j^{*(4)}(s)$	The LSPF of the packet delay in subcase (4).
$w_j^{*(5)}(s)$	The LSPF of the packet delay in subcase (5).
$w_j^{*(6)}(s)$	The LSPF of the packet delay in subcase (6).
λ_s	(Experimental) Arrival Rate.

λ_t	Average throughput rate.
τ	Propagation delay.

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Recently, indoor wireless communications have become an interesting area of research and development. Wireless communications for voice, video, fascimile, and data are needed; with a special emphasis on data transmission due to the increasing popularity of workstations and Personal Computers (PCs) and the need to network them together in an office environment. Depending on applications, the required capacity covers a wide range of speeds from as low as a few hundred bauds (such as in data acquisition and control for alarm and monitoring, point-of-sale data transfer) to several megabits per second (wireless LAN).

One of the driving force behind the expansion of wireless communication is the problems associated with cable. The problems are: 1) Installing a new cable can be very costly, especially in some old buildings that contain asbetos or other dangerous materials. 2) Troubleshooting cable for breaks and bad connections is not only time consuming tasks, but frustrating as well. 3) Despite the promise of structured wiring, a cable must be connected and disconnected to accommodate moving offices and users. These problems make the installation and on going maintenance difficult and expensive.

Wireless communication could provide solution for these problems as wireless systems are more portable and flexible in office reconfiguration and reorganization. Wireless systems consist of a wireless transmission and reception device attached to a network adaptive card (Figure 1.1). They make moving workstations and PCs connected to wireless network a much simple process than relocating a wire based system. Now the users

can move their workstations and PCs freely around the office without spending lot of time in wiring and rewiring the networks. Thus, wireless systems can be a more cost effective, long term solution to office networking needs.

The second, possibly more powerful force behind this growth is the increasing use of the portable/mobile PCs such as laptop, notebook, or palmtop. With portable PCs, we can make our travel time become productive. But unfortunately, the portable PCs need to hook up with a telephone in order to transfer information between the PCs and networks. This is sometimes problematic when we are not near a telephone. With the wireless networking, these problems can be eased.

Wireless communication now plays a major role in the development of the portable/mobile PCs. Now, we can use the portable computers and wireless networking to help us be more productive while we are traveling.

The challenge is in developing wireless systems that can provide the equivalence, if not better performance, of traditional wire based methods. Regarding to the present technology, now this challenge can be done. However, since wireless radio communications use microwaves, the signal is subjected to interference and random noise due to the unusual characteristics of the indoor propagation environment. The environment includes such things as reflections of signal against the walls, and other objects arriving at an antenna at different intervals. These phenomena cause severe degradation in radio system performance, requiring a careful design to overcome them.

Some solutions have been proposed to overcome the above problems. For example, low capacity Local Area Wireless Network (LAWN) systems employs a spread spectrum technique to transmit data between its units [1]. Ram mobile device makes use of radio transmission and packet switching technique to provide service with low speed communcation [2]. These devices can transmit up to 200 meters indoor and 800 meters outdoor. Their primary applications include tracking and tagging of manufacturing status and parts, telemetry, two way communication links, and short data burst between computers.

Although these systems can provide superior performance in combatting the random noise and interference, the costs of these systems are very high due to the complexity of their designs.

The objective of this research is to introduce and study a low-cost, low capacity wireless data communication system for applications. The proposed wireless data communication system can operate in various topologies: mesh, star, tree, etc. With this objective in mind, the introduced system makes use of a simple radio transmitter/receiver, and an adaptive Reed-Solomon coding scheme to combat both random and burst errors due to noise and interference. A Carrier Sense Multiple Access with Collision Detection (CSMA/CD) scheme is used for the data communication between the terminals. Both data traffic and respond signal (ACK) share the same radio channel using an Asynchronous, Character-Oriented transmission technique.

The main contributions of this thesis are:

1) The performance and feasibility evaluation of a low-cost, low capacity wireless data communication system.

2) The performance analysis of CSMA/CD with both data and acknowledgment signals sharing the same radio channel (CSMA/CD with ACK) as a method for multiplexing terminals on a packet switched multiple access radio channel.

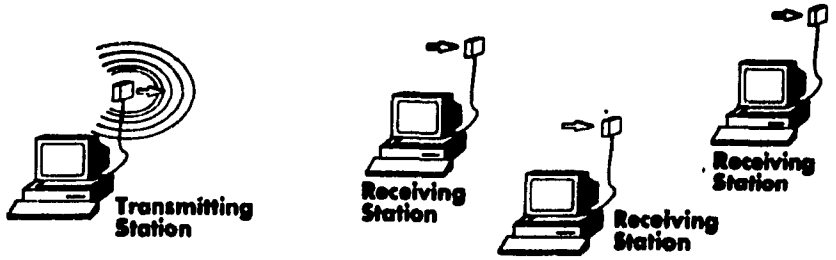


Figure 1.1 : Wireless Communication System.

1.2. THE SCOPE/PLAN OF THE THESIS

The scope of this thesis is to evaluate the performance and feasibility of the proposed system using both analytical and experimental works. The organization of this thesis is as follows.

This chapter provides an introduction to the thesis.

Chapter 2 presents a description of the proposed wireless data communication system. The operation of the proposed system will first be discussed. Then, the major elements of this system will be described in detail. Finally, experimental results on the transmission characteristics will be presented and discussed along with the reasons for using an adaptive Reed-Solomon coding scheme.

Chapter 3 presents the analysis of the throughput and delay performance of the proposed system. We first describe the analytical model of CSMA/CD with ACK scheme. Using this model, the throughput and delay performance of the proposed system will then be derived under the Poisson finite user population arrival model. The effects of transmission characteristics will also be considered in this analysis.

Chapter 4 provides the experimental results on the system performance. In this chapter, we will first describe the experimental data communication system. Then, the measurement techniques used to compute the system throughput and delay will be presented. Finally, the measured results will be presented and compared with the analytical ones which are given in the Chapter 3. The feasibility and applicability of the proposed system are also discussed.

Chapter 5 contains the conclusions and suggestions for further research.

CHAPTER 2

A LOW-COST WIRELESS DATA NETWORK

The objective of this chapter is to describe the operation and elements of the proposed low-cost wireless data communications system, and its measured transmission characteristics. With this objective in mind, we will divide this chapter into three main sections. Section one will provide the overall system operation. The physical description of the low-cost wireless data communication device will be presented in the section two. Section three will discuss the system transmission behavior over a radio channel. We conclude this chapter by showing how the Reed-Solomon (RS) code can be used to improve the system performance.

2.1. SYSTEM OPERATION

This system can be used in point-to-multipoint or mesh configurations. In a point-to-multipoint arrangement, the system consists of one central station and several remote terminals. Data transmission among the remote terminals has to be directed through the central station. On the other hand, the mesh configuration allows every terminal to send data directly to every other terminal in the system. In an environment with relatively short propagation delay such as indoor wireless communication, the mesh configuration can provide a highly efficient channel utility for a modest complexity [3]. In the following section, we will describe the mesh configuration, its data formats, and the character oriented transmission of the wireless data communication system.

2.1.1. System Configuration

Consider an environment consisting of a finite number of users communicating with each other over a radio channel of limit bandwidth using the mesh configuration, the transmission errors are basically due to two major causes:

1) Random noise and interference on the radio channel.

2) Packet collision during the transmission.

To combat the interference of random noise on the radio channel, a hybrid Stop and Wait Automatic Repeat Request (Hybrid ARQSW) scheme is used. This scheme is to combine the Reed-Solomon Forward Error Control Coding (RS) with the positive/negative acknowledgment of each correctly/uncorrectly received message. The RS code can correct e symbol errors and ρ erasures where e and ρ can be programmed. Each packet will be encoded by a Reed-Solomon encoder before passing to the transmitter. At the receiver side, after receiving the packet, the packet will be sent to the Reed-Solomon decoder for error correction and detection. If no error is found, a positive acknowledgment (ACK) response packet will be encoded and send back to the sender. If the decoder detects the errors, it will first attempt to locate and correct the errors. If the errors can be corrected, the coded positive acknowledgment response will be sent back to the sender. On the other hand, if the errors cannot be corrected, the coded negative acknowledgment response will be transmitted back to the sender to indicate that an error in the received message has been detected. The packet is considered successfully transmitted if and only if, within an appropriate time-out period, the sender receives a coded positive acknowledgment response in which no error is found or the errors are corrected. Otherwise, the packet is assumed to be lost and will then be retransmitted. To avoid a continuously repeated conflicts, the unsuccessful packet will be scheduled for transmission at some later time according to the exponential random distributed retransmission delay.

To deal with the packet collision problem, a Carrier-Sense-Multiple-Access with

Collision Detection (CSMA/CD) protocol is used to reduces the chance of collision in the random-access channel by allowing terminals to sense the channel status. The operation of the CSMA/CD scheme can be described as follow: a user with a packet ready for transmission senses the channel and proceeds as follows.

(i) If the channel is sensed idle, the user initiates transmission by sending the SYNC characters. Concurrently it senses the channel and go to step (iii).

(ii) If the channel is sensed busy, then the user schedules the retransmission of the packet to some later time according to the exponential random distributed retransmission delay and repeats the procedure.

(iii) If one SYNC character is not properly received (i.e., the received character is erased), the sender declares a collision, stops its transmission and the packet is scheduled for retransmission at some later time according to the exponential random distribution delay. The user then repeats the algorithm.

(iv) If two SYNC characters are successfully received, the sender considers that the channel is occupied by itself and continues its transmission until the end of the packet. Go to the next step.

(v) The sender waits for the acknowledgment from the destination terminal. If a positive ACK is received, the transaction is completed. Otherwise the packet will be scheduled for transmission in the future by an exponential random distribution delay and the user will repeat the algorithm.

In the above mentioned transmission procedure, we can see some special features different from the original CSMA/CD technique. When the channel is sensed idle, terminals with message ready for transmission will initiate their transmission. Channel sensing is performed in a finite time window. The decision to initiate the transmission also takes a finite time interval. Therefore, if two terminals decide to transmit their messages, the transmission of their initial SYNC characters is not exactly at the same time. The decision made in step (ii) and (iii) gives the terminal which transmit first, a chance to capture

the channel and to complete its transmission. Furthermore, the acknowledgment scheme in step (v) allows the success of a communication transaction.

In step (ii) if collision is detected, the transmission is aborted and the message is scheduled for retransmission. Non-persistent, p-persistent, or 1-persistent protocols [3] can be applied for retransmission. The selection of the protocol depends on the monitored traffic load.

2.1.2. Data Format

The format of data packet is shown in Fig. 4.2. It consists of three parts. The first part contains two SYNC characters to indicate the beginning of a packet. These characters are also used for collision resolution which was discussed in previous section. The second part is the header. It has a fixed length and uses a fixed coding rate. As the header is relatively short and contains control information, a low coding rate is applied to provide a high detection performance in all circumstances. The header has six characters: source identifier, destination identifier, packet type, packet length, coding rate, and the Binary Cyclic Checksum (BCC). The third part (information) contains the message itself. The packet length and coding rate applied to the third part are given in the header and can be changed on the packet basis. The BCC is used for ensuring the integrity of the transmitted data. For short inquiry or response, the packet contains only SYNC characters and the header. In this case, the field of packet type carries short commands or responses.

2.1.3. Character Oriented Transmission

Two major transmission modes are used in data communication, namely, synchronous mode, and asynchronous mode. In a synchronous transmission mode, the receiver has to recover the data clock synchronized with the transmit clock.

On the other hand, in an asynchronous character-oriented transmission mode, whenever the data character is sent by the transmitter, start bit is added to the front of data bits,

and one or two stop bits to the end of the packet. Receiver clock has the same usually nominal frequency as the transmit clock but need not be synchronized to the latter. The start and stop bits are used to absorb the frequency difference. In addition, an even or odd parity bit is inserted prior to the stop bits for error detection as showned in Fig. 2.1.
This character will then be sent out with a fixed rate at the transmitter output. On the receiver side, the start, parity, stop bits for each character will be examined and removed before delivering the character to the output buffer. The synchronous transmission mode is much more efficient than the asynchronous one and hence often used when the data rates are very high such as in the case of fiber optic communication, satellite communication. On the other hand, the asynchronous mode is useful for low speed applications where low-cost, low-complexity receiver is a primary requirement. Aiming to provide a low-lost solution to a low capacity wireless indoor application, the asynchronous character-oriented transmission mode is adopted in our portable data communication system.

2.2. ELEMENTS OF THE PORTABLE DATA COMMUNICATION SYSTEM

2.2.1. System Descriptions

The portable data communication system consists of four main blocks: The Non-Coherent Super-Regenerative Receiver, the simple Amplitude Shift Keying (ASK) Transmitter, the RS-232 interface and the Reed-Solomon RS(31,K) (encoder/decoder) (RS31KED) card. These four main blocks are incorporated into a personal computer (PC) as shown in the Figure 2.2.

The receiver has a whip antenna that hangs out of the box. The received ASK signal is demodulated into a binary signal by the receiver. This binary signal is then fed to the RS-232 interface board where it is converted to the appropriate RS-232 levels (±12v). The RS-232 signal is then passed to the I/O controller card which is mounted inside the I/O slot of the personal computer (Figure 2.2). The I/O controller card will

generate the interrupt to signal the system board enabling the input/output interrupts of the RS31KED card. After that, the I/O controller will send the received data to the RS31KED chip set to decode the incoming data. After decoding the data, the RS31KED will send the decoded information to the system board for further processing. In the meantime, the signal board will disable the input/output interrupts of the decoder.

On the other hand, if the computer has data ready to transmit, its system board will enable input/output interrupts of the encoder on the RS31KED chip set and send the sequence of data to RSS31KED for encoding. After encoding the data, the system board will disable the input/output of the encoder, enable the transmitted interrupt on the I/O controller card, and transmit the data to the I/O controller. The I/O controller will then send the sequence of data to the RS-232 interface where the voltage level is converted to an appropriate binary signal which is subsequently fed to the non-coherent ASK transmitter where it is correspondingly modulated. The modulated signal is radiated via a printed-loop antenna.

The transmitter, Super-Regenerative receiver and the RS-232 interface are placed in a plastic box as illustrated in Figure 2.3. The RS31KED card is plugged into the I/O slot inside the personal computer.

In the following sections, we will describe the operation of the transmitter and receiver, as well as the use of RS31KED card.

2.2.2. Low Cost ASK Transmitter:

In order to transmit digital information over a bandpass channel we have to transfer the information to a carrier wave of appropriate frequency. There are several techniques available to modulate a digital signal. For low-cost application, the non-coherent Amplitude Shift Keying (ASK) is employed. An example of the binary sequence and its corresponding ASK modulated carrier is illustrated in Figure 2.4. One can see that the ASK signal is an ON-OFF Keying (OOK) type.

The physical size of the non-coherent ASK transmitter is 1" × 1". Its block diagram is shown in Fig. 2.5. The output of the interface driver is a binary sequence $\{b_k\}$ at a rate f_b and duration T_b. This sequence modulates the carrier generated by a Quartz SAW (Surface Acoustic Wave) Resonator at resonant frequency of 308 MHz. The high level of a binary signal will bias the transistor of the Quartz SAW Resonator while a low level binary signal will not, thus allowing or disallowing an On-Off Keying signal at 308 MHz. The resulting OOK signal is bandpass filtered, amplified by an RF amplifier and propagated by a printed-loop antenna.

2.2.3. Low Cost Super-Regenerative Receiver:

The physical size of the Super-Regenerative receiver is 2" × 3". It consists of an RF oscillator running at or close to the desired receiver center frequency. The RF oscillator is alternated between an oscillating and a non-oscillating condition at a low radio frequency rate known as the quench frequency. It also consists of a low-pass filter. This is illustrated in Figure 2.6.

The operation of the super-regenerative receiver can be explained as follows [4]. Super-generation is based on the application of a controllable positive feedback. The feedback in the regenerative detector is increased beyond the critical point thus permitting self oscillation to occur with rapid build-up. The oscillation is stopped soon afterward allowing the process to continue on continuously with the circuit oscillating in bursts. The oscillation is switched on and off by a voltage signal called the quench voltage. The quench voltage cycles between positive and negative cycles as seen in Figure 2.7. During the positive half cycle the quench voltage is seen to drive the circuit to self oscillate, while in the negative half of the cycle the oscillation is seen to be quenched or stopped. The build-up and stopping oscillation depend on the magnitude of the tunned-circuits time constant relative to the quench-frequency period.

The quench oscillator builds up from an initial value corresponding to the noise vol-

tage in the input circuit to a final value corresponding to an equilibrium value for the oscillator. These quench oscillations die out when the quench voltage becomes small or negative resulting in the quench oscillator ceasing to be in an oscillating condition. From Figure 2.8 the build-up of oscillations can be seen for the case when no signal is applied.

Now suppose that a signal voltage is superimposed upon the system and is larger in magnitude than the thermal agitation noise in the circuit. When the oscillation starts to build up the initial amplitude correspondings to the amplitude of the superimposed signal, rather than the small amplitude of the noise voltage in the circuit. The oscillations , therefore, reach their equilibrium value more quickly than before because of the larger initial amplitude. This is illustrated in Figure 2.10, where we see that with a signal the steady state oscillation reaches a maximum amplitude at a time $t1$ while in the no-signal operation the maximum amplitude of oscillation is reached somewhat later at $t2$. The graphs in Figure 2.9 illustrate the response of a Super-Regenerative Receiver (SRR) in the presense of a modulated signal. The presence of a signal accordingly increases the average area under the envelope of the quench signal that is occupied by the RF oscillations according to the level of the applied signal.

The demodulated signal is then obtained by simply low pass filtering the oscillator output and rejecting RF and quench frequencies. A more detailed block diagram of the SRR is sl.own in Figure 2.11. The SRR provides a simple means of obtaining a very large amount of radio-frequency amplification at frequencies that are difficult to amplify by conventional means. But, super-regeneration has the disadvantage of providing poor selectivity resulting from an excessive bandwidth and center frequency drift due to temperature changes and component aging. It finally provides a characteristic hiss that is present in the absence of a signal as a result of amplified thermal agitation noise. These problems are traced to the LC (Tank Circuit) components that determine the center frequency and bandwidth. It can be overcame by the employment of a narrow bandwidth stable frequency reference (SAW). This component will be employed in the future

design.

2.2.4. The Use of RS31KED Channel Codec

Both encoder and decoder chips are implemented using the low-power, 1.5 micron, 2-layer metal, HCMOS technology. They can be operated in two modes [5]: Stand alone (ST) and microprocessor peripheral (MP). In ST mode, they operate as a stand-alone device, directly in the data symbol stream, at a data rate up to 3 Mb/s(for encoder) or 1.5 Mb/s (for decoder). In MP mode, they can be directly interfaced to any 8-bits, 16-bits, or 32-bits standard microprocessor, microcontroller, or digital signal processor (DSP).

The encoder (RS31KE) operates in a 5-bits symbol Galois field GF(2^5). It accepts a block of K information symbols at the symbol rate defined by the Write signal (WRN) and produces a code word of 31 symbols at the symbol rate defined by the Read signal (RDN). The information block length K, is user-programmable up to the code length.

The error control decoder (RS31KD) also operates in a 5-bits symbol Galois field GF(2^5). It receives a code word of L (length of code word) symbols and erasure flags (if available) at the symbol rate defined by the signal WRN and produces a block of K (length of the information block) decoded symbols at the symbol rate defined by the signal RDN. Both parameters L and K are user-programmable.

The RS(31,K) codec can correct both erasures and errors. It can also be used for punctured codewords as follows. Consider a codeword of 31 symbols denoted as $C_0, C_1, \ldots, C_{30}$. As an example, a punctured codeword of 28 symbols can be formed by dropping 3 symbols C_i, C_j, and C_l. At the receiver, dummy symbols are inserted in positions i, j and l and these dummy symbols are declared as being erased before passing the received codeword to the RS(31,K) decoder. The positions i, j, and l have to be known by the destination terminal. Otherwise, the received codeword will be wrongly decoded. Therefore, the missing positions can be selected in such a way to prevent unintended receiving terminals from detecting the coded message. This feature can be used for data privacy.

2.3. CHANNEL BEHAVIOR

The main problem in the wireless data communication is the interference and random noise over a radio channel. This interference can be caused by the reflection of the signal against the wall (fading) or by the disruption of the other sources. Unlike the cabled system where the effect of the random noise over the transmission line is very small, the interference and noise on the radio channel make data communication unreliable and dramatically decreases the performance of the system. The communication can also be disrupted by a person who accidentally stands in the wrong place. This phenomenon makes the channel become very complex and unpredictable. In other words, the channel is not only disturbed by an Additive White Gaussion noise (AWGN) but also interfered by unpredictable interference such as man-made noise, multipath, reflection etc. As a result, we have encountered many burst errors during the transmission of the data over an indoor radio channel. The propagation characteristic can be nonstationary, and very difficult to be modeled. Therefore, in this thesis, we focus only on the error distribution rather than the propagation model. In the following sections, we will describe the experiments set for measuring the error distribution. From these results, another experimental tests set up to computer the probability of success of a transmitted data over an indoor open channel environment.

2.3.1. Error Distribution

To examine the link performance in an indoor environment, the experiments set for measuring the error distribution were set up in a point-to-point configuration for various conditions [6]. One transceiver acting as a transmitter was connected to a personal computer while the other transceiver acting as a receiver was connected to another personal computer. The first one (sender) continuously transmits data packets of 200 bytes at 1200 and 2400 baud rate, while the second one (receiver) was used to accumulate the data and

to record the error pattern. In this test, we performed various transmission experiments in different environmental conditions [6].

From the experimental set up, one expects that as the physical distance between two terminals in a typical indoor arrangement is relatively short, the signal-to-thermal-noise power ratio is high. Therefore, random errors rarely occur. But instead, our experimental results shows that errors come in bursts mainly due to man-made interference or reflections. It is also noticed that the transmission behavior of an indoor wireless environment is quite varying and almost unpredictable. This leads us to the consideration of an adaptive burst-error correction and detection scheme to improve the link performance. By adaptive, we mean that the number of correctable and detectable errors can be programmed according to the link performance situation. When a good channel performance is detected, a high coding rate is used to obtain a high transmission efficiency. When the channel error rate gets worse, the coding rate is reduced to increase the burst error correction capability of the code. Therefore, the message retransmission due to channel error is minimized. To this end, the RS31KED chip set as described in Section 2.2.4 is used as a tool to combat the interference of random noise over an indoor radio channel.

2.3.2. Successful Transmission Probability

In order to evaluate the possible performance enhancement due to RS codes we set up the experiments to measure the successful transmission probability Q of the system with and without RS coding. The successful transmission probability Q is experimentally measured as the ratio of number of successfully transmitted packets to the number of transmitted packets over a period if transmission time, i.e.,

$Q = Pr\{$ a transmitted packet is successful $\}$

= Total number of successful packets / total number of transmitted packets .

In our experimental set-up, we used three terminals. But only one terminal transmits

and the other two receive. Therefore, there is no transmission contention and unsuccessfully transmitted packets are due to only channel noise and interference.

If $Q = 1$, it indicates that the channel is error free. If $Q < 1$, it indicates that there are transmitted packets destroyed by the random noise and/or interference over the radio channel. Therefore probability Q indicates the rate of the successful transmission only subjected to channel interference and noise but not subjected to the packet collision.

The first computer (sender) sends a fixed length data packet of 62 bytes having the format as described in Section 2.1.2 at 1200 bits/second to the other two computers. The other two computers (receivers) receive the transmitted packets and compared with an previous fixed packet. If the pattern was matched, the receivers sent a positive acknowledgment response back to the sender. Otherwise, a negative acknowledgment response was transmitted back to the sender. The packet was considered to be successfully transmitted if and only if the sender received a positive acknowledgment response within an appropriate time out delay. The sender keeps track of the number of the successfully transmitted packet as well as the total number of the transmitted packet. In those tests, two parameters were varied, namely, the coding scheme and environmental conditions. In the coding scheme, Two main coding techniques, namely, the Binary Redundant Cycle Checksum (BCC) (Uncoded scheme) and the combination of BCC with the Reed-Solomon FEC (Coded scheme) were used. We also set up two environment conditions, namely, setup # 1 and setup # 2. In the setup # 1, three transceivers are located inside one room at a separation of about two meters (Figure 2.12). In setup # 2, two transceivers are placed in the same room while the third transceiver is located in a different room (Fig. 2.13). Between these two rooms, there exists a concrete wall which obstructs the transmission of the radio signal.

Each of the experimental test was carring out for an approximately half an hour. The experimental results are summarized in Tables 2.2 to 2.5. The structure of the table 2.1 has three parts, namely, description, content and result. In the description part, the

table provides the information used to identify the type of collected data. It includes: (1) The coding scheme, (2) the location in which the experimental tests took place and (3) the testing environmental condition or the configuration. In the content part, the total number of successful transmitted packets, the total number of the transmitted packets, and the probability of success in each individual test are given in the column 1, 2 and 3, respectively. Finally, the result part shows the overall successful transmission probability Q of the system.

2.3.3. Discussion

For the set up # 1 of Figure 2.12 one can see that the transceivers are located very close to each other. The interference and random noise on the transmitted data over the access channel have small effects. Therefore, the successful transmission probability Q is very high. The results of the experiments in Tables 2.2 and 2.4 show that the successful transmission probabilities Q are 0.993 without RS coding and 0.996 with RS coding. RS coding is not so helpful as Q without RS coding is already very good. On the other hand, due to the fact that the distance among those transceivers is increased in the case of the set up # 2 (Figure 2.13), the transmitted packets are subjected more often to interference and random noise. Their effects have tremendously degraded the performance of the system. From Table 2.3, one can see that the successful transmission probability Q is around 0.91 which is 0.09 lower than that of in the case of setup # 1 without RS coding. When the RS code is used, the successful transmission probability Q (Table 2.5) is increased to 0.989 which is only 0.01 lower than that of the setup # 1 with RS coding. Comparing the probability of success Q in case of uncoded and coded setting # 1, we can see that in the case of setup # 1, the packet rejection rate due to only channel errors is 0.003. The system throughput without and with burst-error correction coding is quite similar. However, as the errors occur more often, burst-error correction coding helps to improve the system throughput. This is indicated in Table 2.5 when the packet rejection rate due to only

channel error increases to 0.09 (Table 2.3). This shows that the RS code can be used in the data communication system to improve the performance of the system by combating the interference of noise over an indoor radio channel.

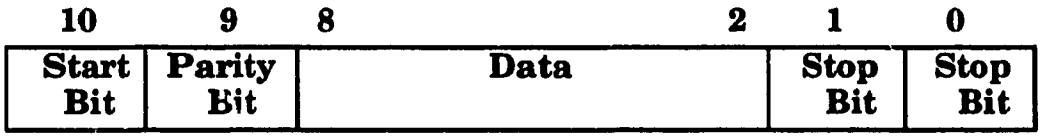


Figure 2.1: Character Format in an Asynchronous Character-Oriented Transmission Mode.

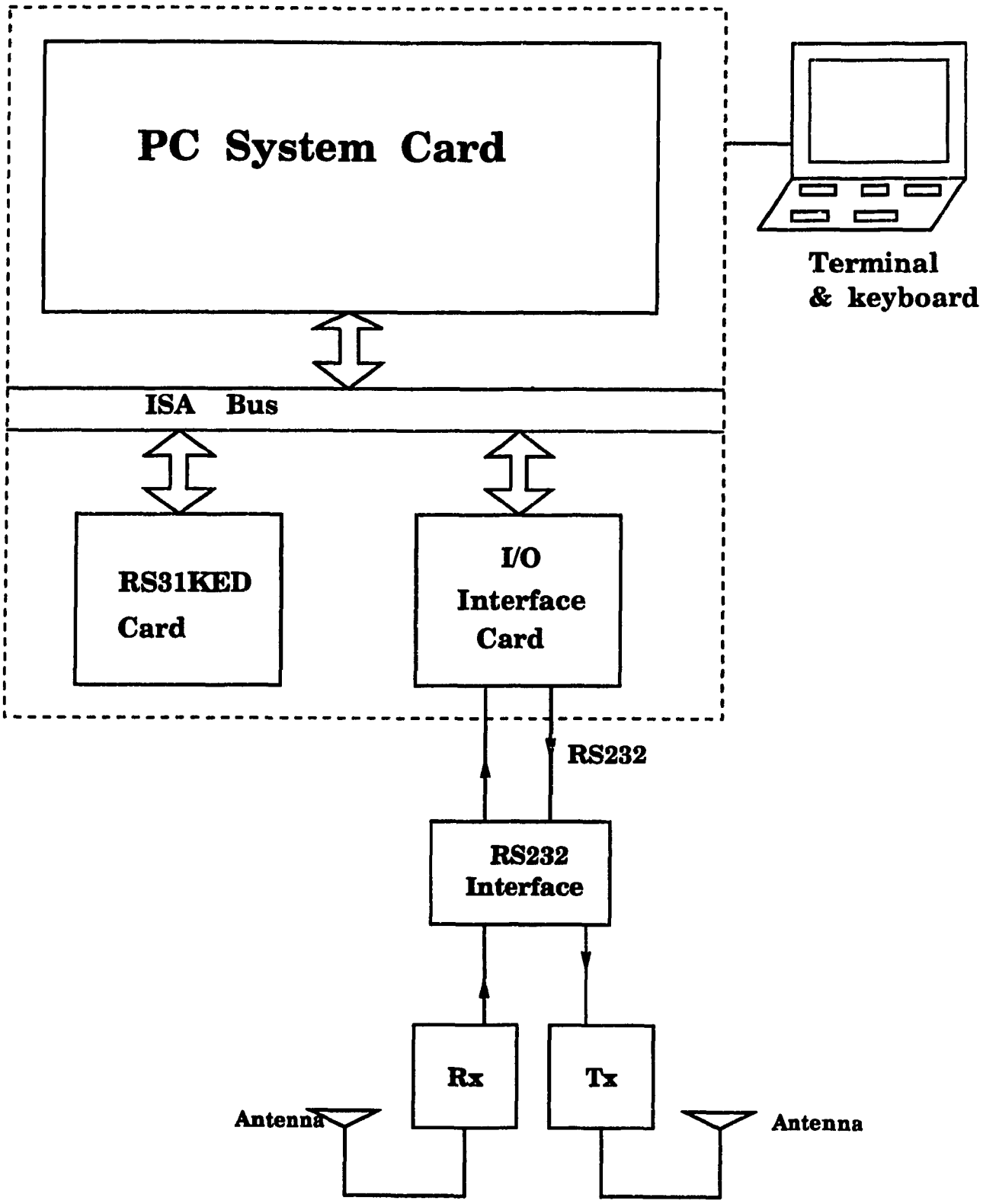


Figure 2.2 : Block Diagram of the Portable Data Communication System.

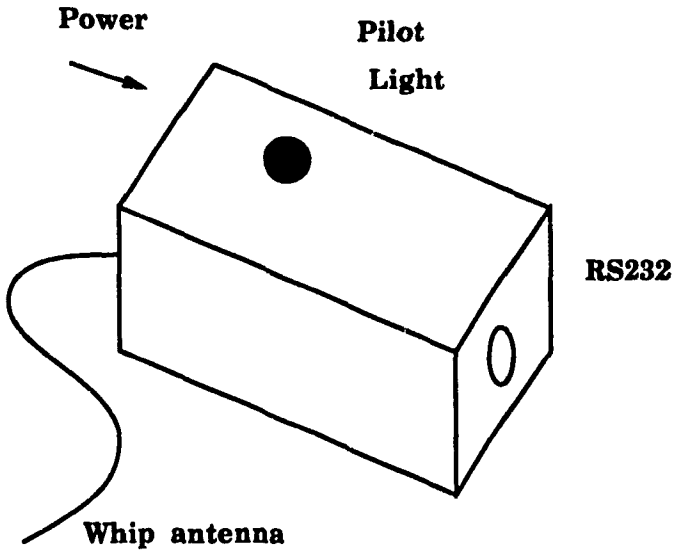


Figure 2.3 : Box Containning Hardware.

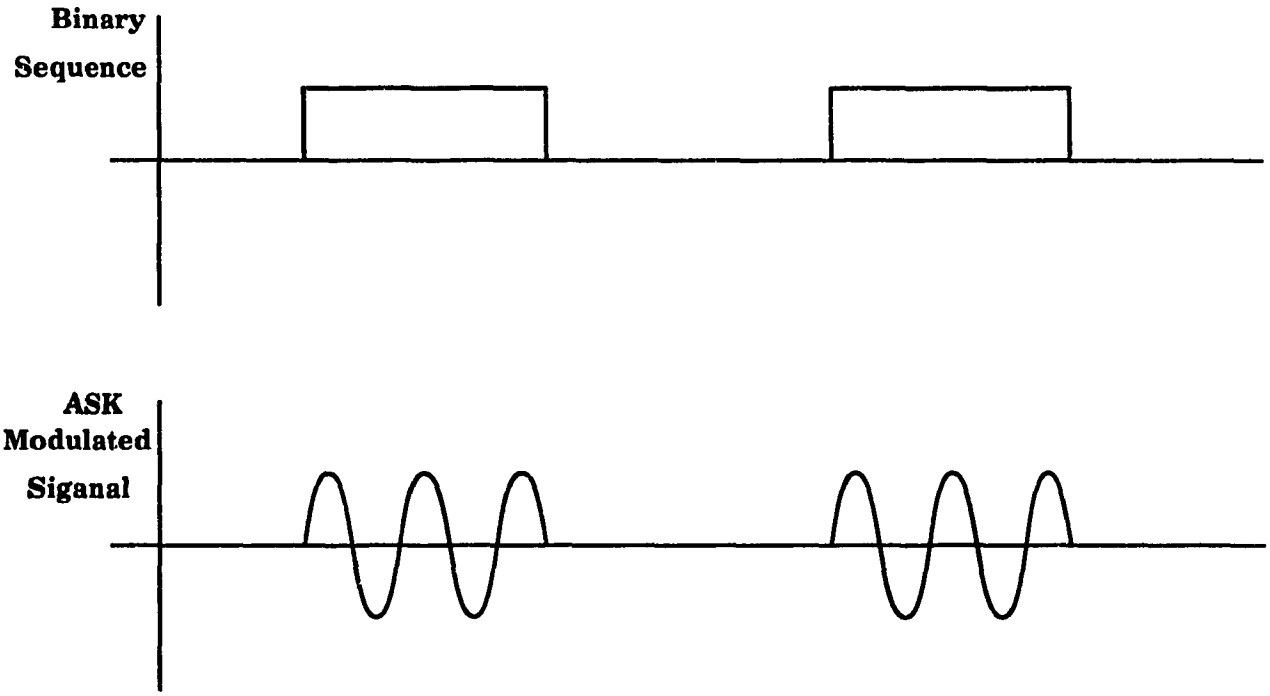


Figure 2.4 : Binary and ASK Modulated Signal.

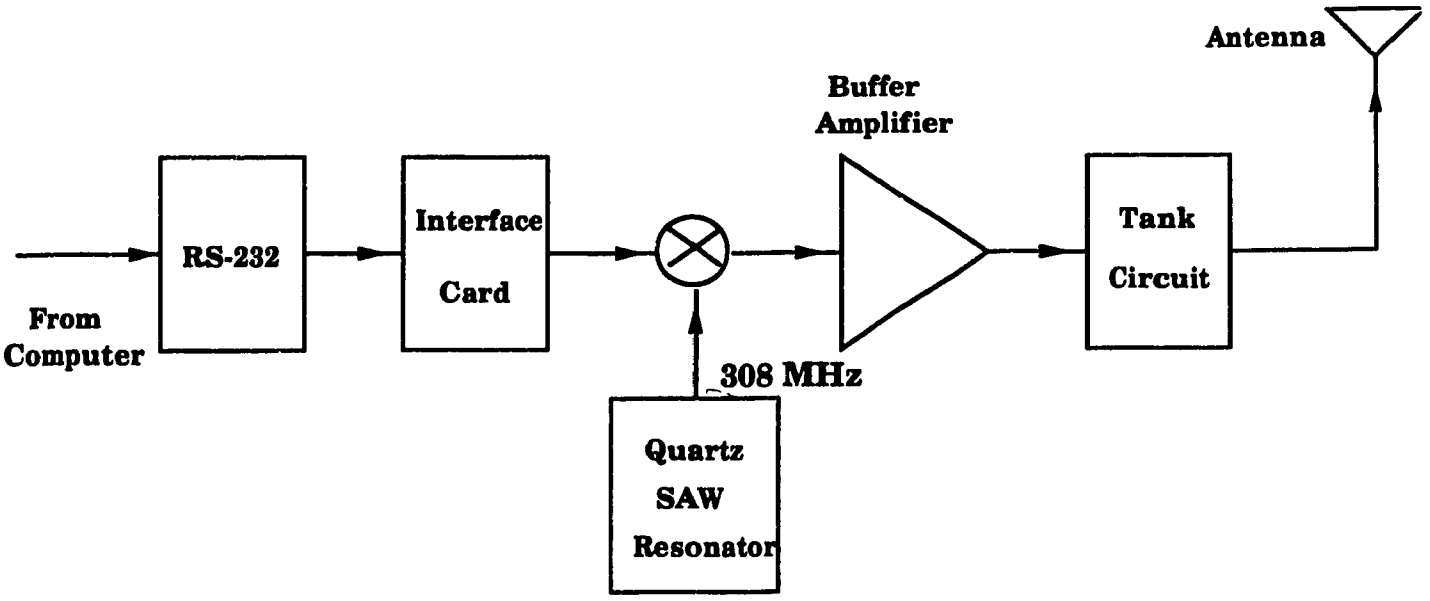


Figure 2.5: Block Diagram of ASK Transmitter.

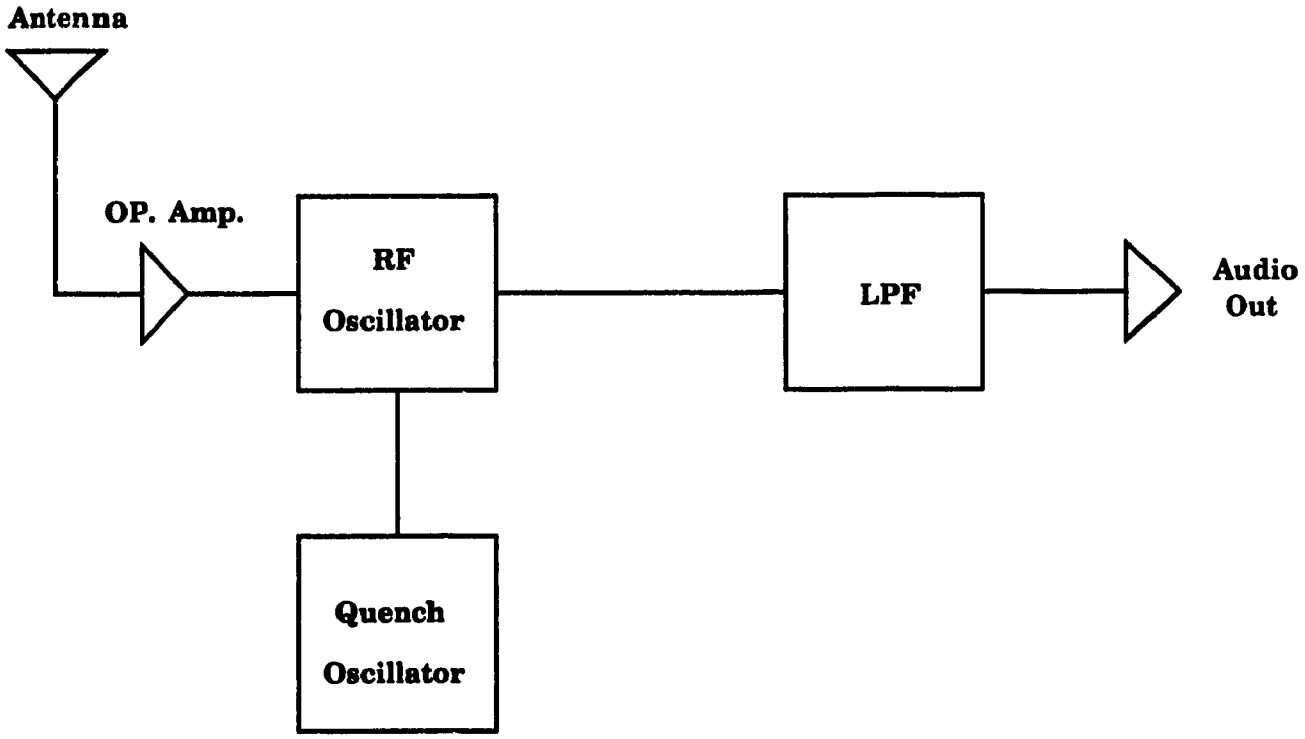


Figure 2.6: Block Diagram of Super-Regenerative Receiver.

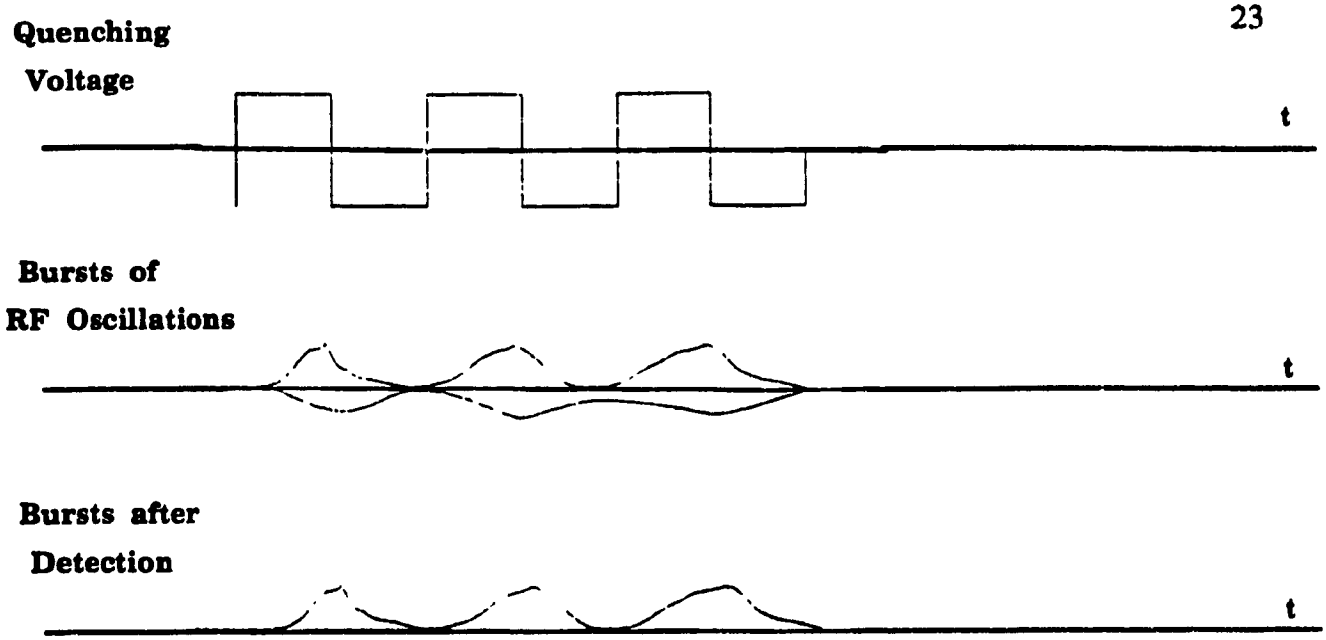


Figure 2.7: Oscillation in a non-signal State.

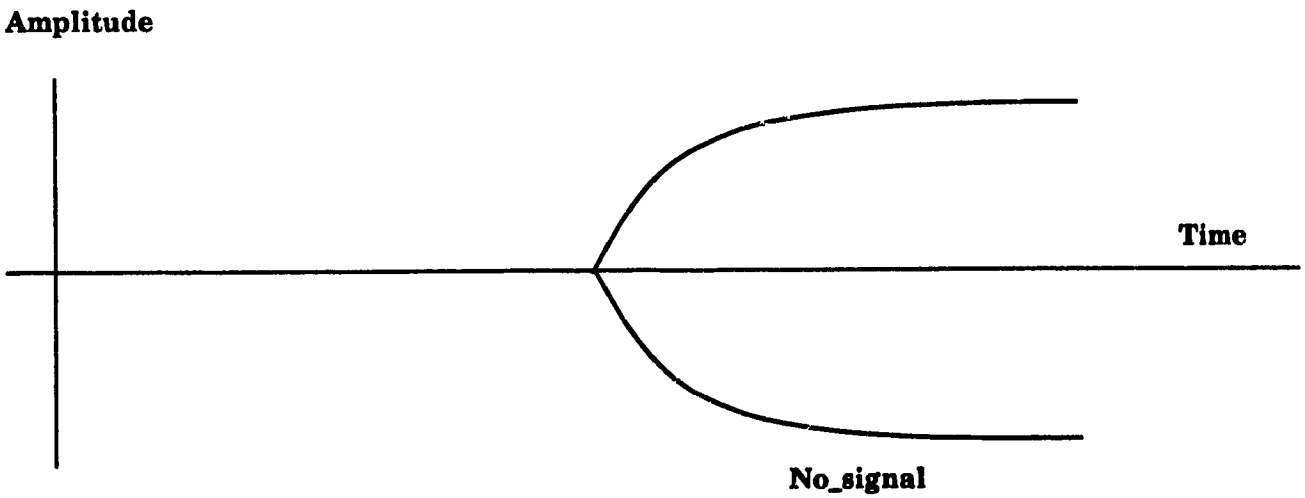


Figure 2.8: Build-up of Oscillation With No-signal Applied.

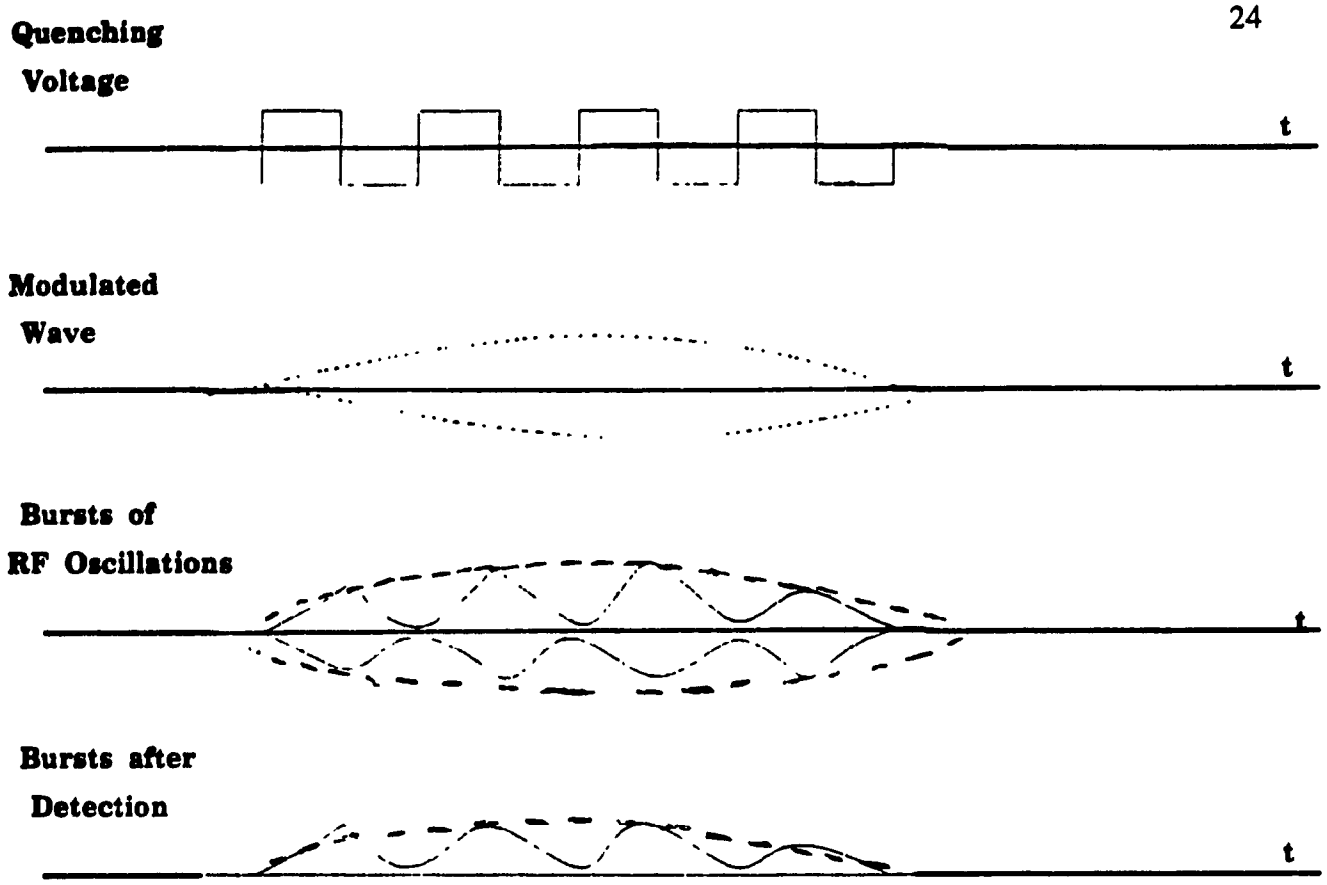


Figure 2.9: Oscillation in a Signal State.

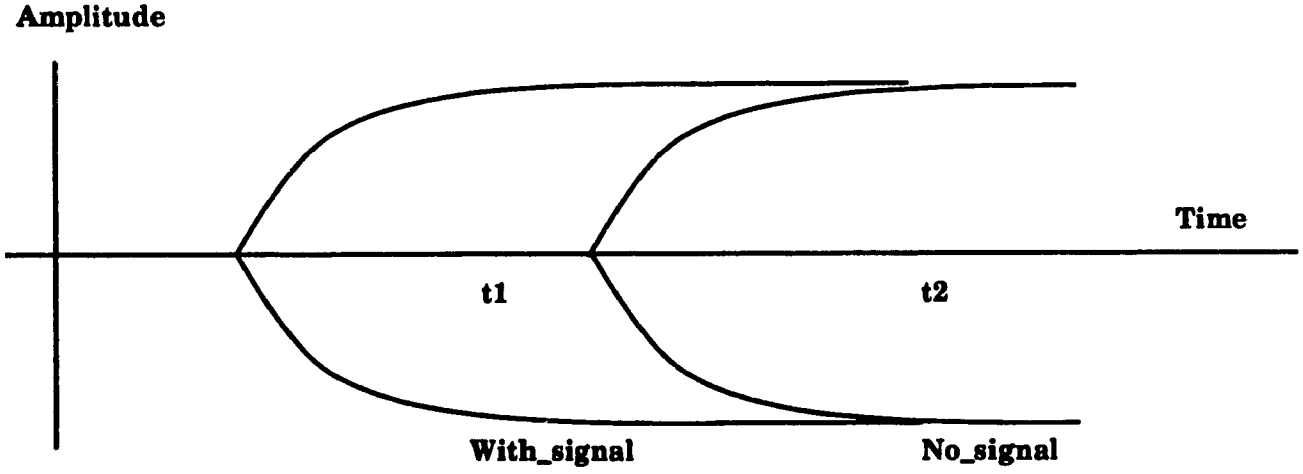


Figure 2.10: Comparision of Oscillation Build-up With Signal.

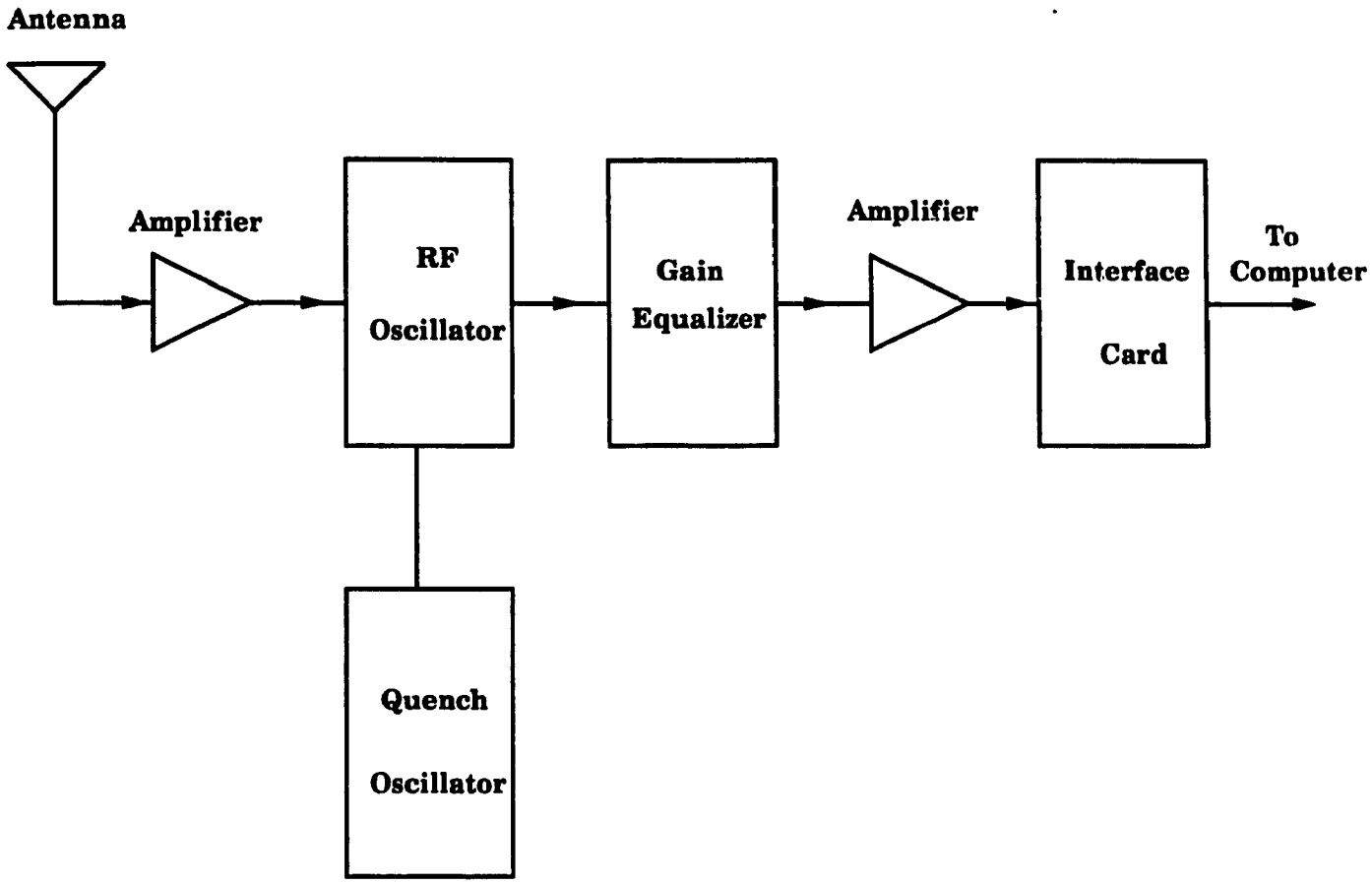


Figure 2.11: Detail Block Diagram of Receiver.

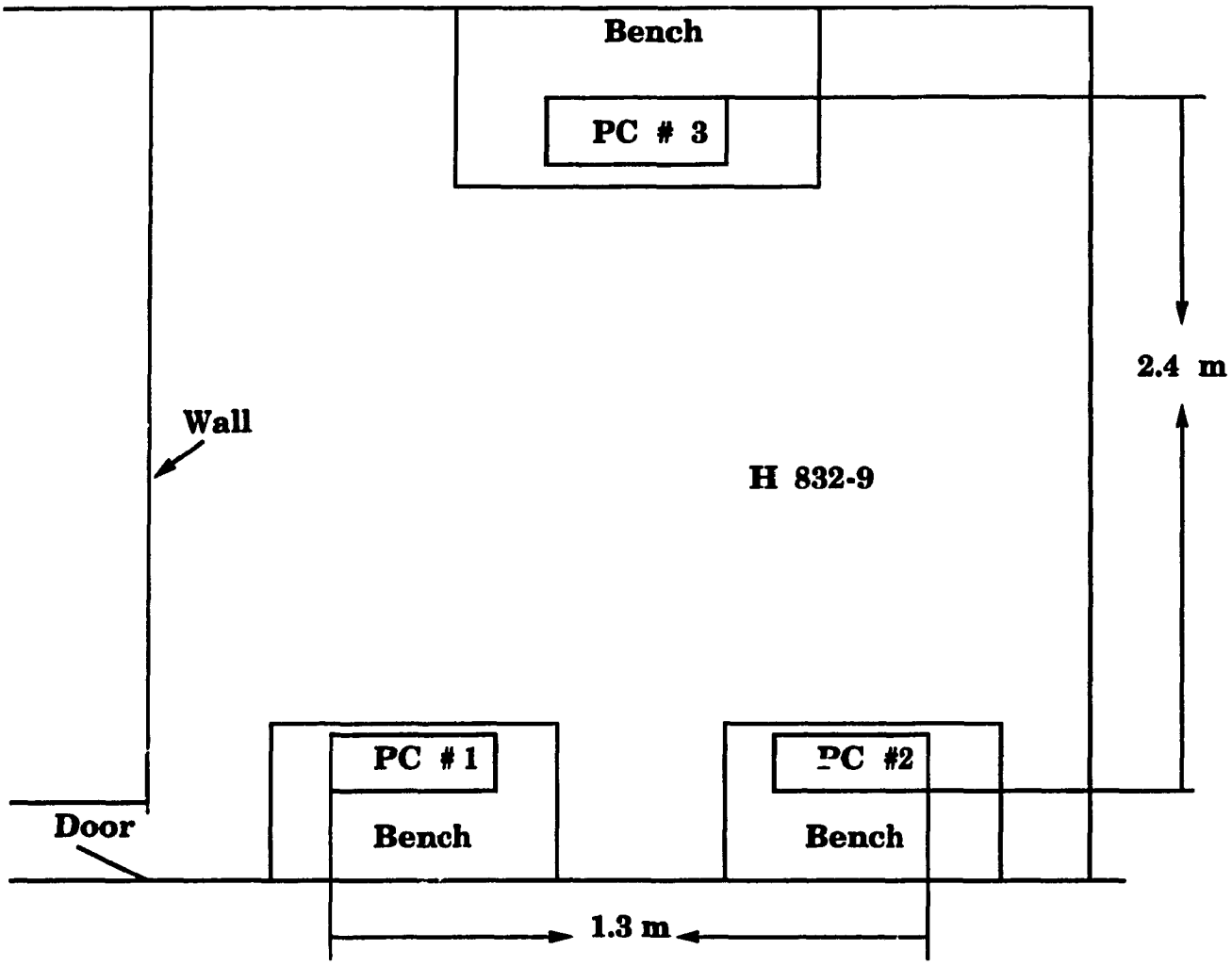


Figure 2.12: Test setup for distribution error # 1

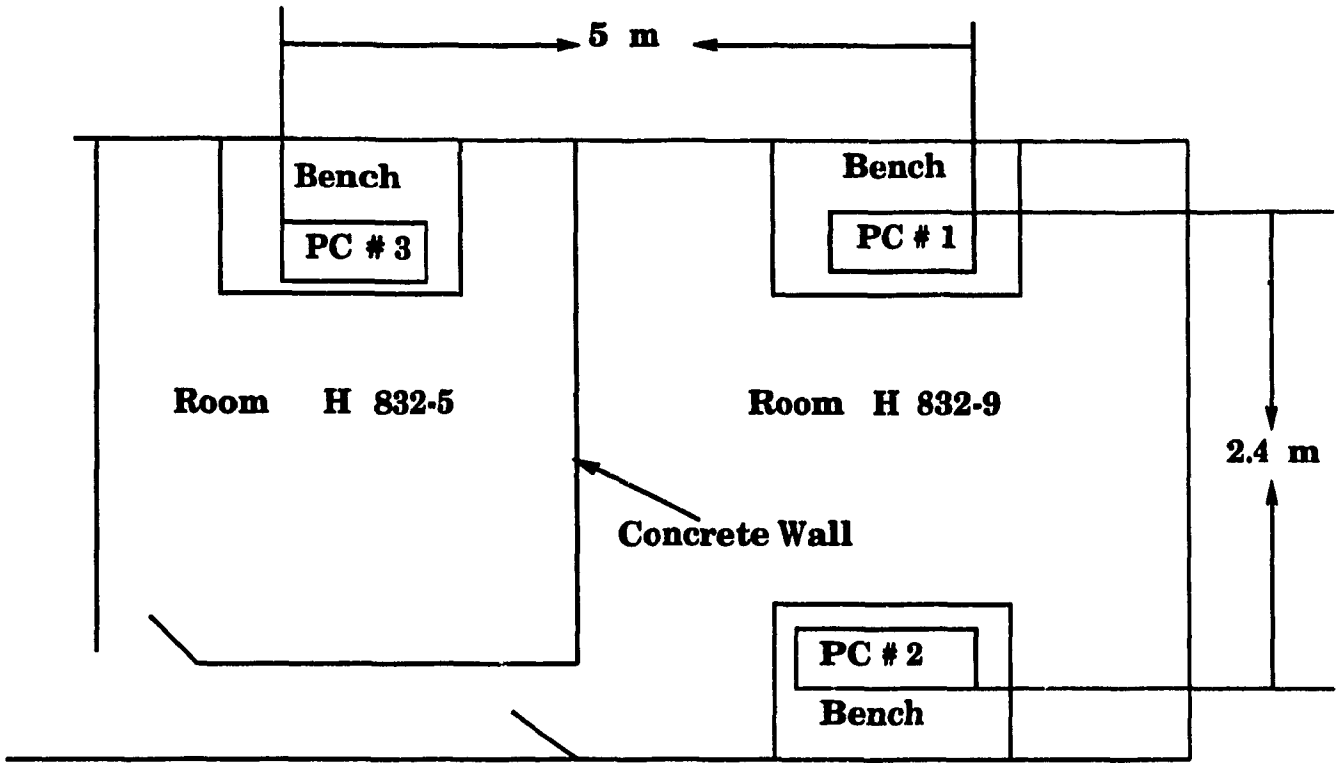


Figure 2.13: Test setup for distribution error # 2

Table 2.1 : The structure of the Sample table for Distribution Error Setting.

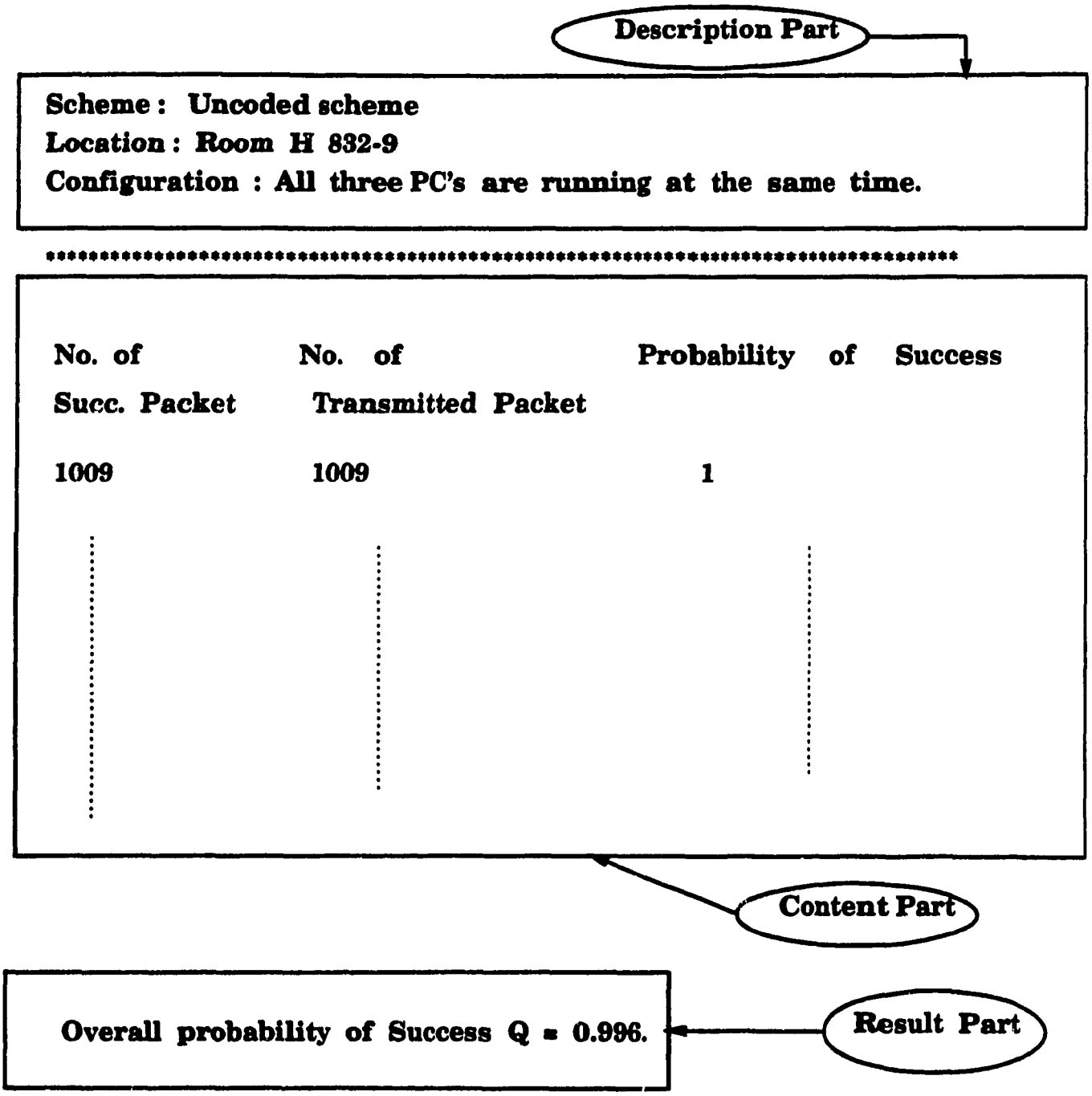


Table 2.2 : Uncoded Error Distribution Setup Test # 1

Scheme: Uncoded.
Location: room 832-9.
Configuration: 1 machine is running and the other two are listening.

No. of Succ. Message	No. of Messages	Probability of Success
1372	1429	0.960112
1423	1424	0.999298
1433	1433	1.000000
1385	1389	0.997120
1109	1116	0.993728
1100	1100	1.000000
1081	1082	0.999076

Overall Successful Transmission Probability Q = 0.993

Table 2.3 : Uncoded Error Distribution Setup Test # 2

Scheme: Uncoded.
Location: rooms 832-9 and 832-5.
Configuration: 1 machine is running and the other two are listening.

No. of Succ. Message	No. of Messages	Probability of Success
1183	1300	0.910
1349	1363	0.987
1241	1405	0.883
1165	1352	0.862

Overall Successful Transmission Probability Q = 0.91

Table 2.4 : Coded Error Distribution Setup Test # 1

Scheme: Coded.
Location: room 832-9.
Configuration: 1 machine is running and the other two are listening.

No. of Succ. Message	No. of Messages	Probability of Success
1009	1009	1.000000
1003	1003	1.000000
1011	1013	0.998026
999	1013	0.986129
1029	1029	1.000000
992	998	0.993988

Overall Successful Transmission Probability Q = 0.996

Table 2.5 : Coded Error Distribution Setup Test # 2

Scheme: Coded.
Location: rooms 832-9 and 832-5.
Configuration: 1 machine is running and the other two are listening.

No. of Succ. Message	No. of Messages	Probability of Success
1063	1070	0.9935
1025	1031	0.9942
1098	1113	0.9870
969	987	0.9810

Overal Successful Transmission Probability Q = 0.9889

CHAPTER 3

SYSTEM THROUGHPUT AND DELAY ANALYSIS

The objective in this chapter is to analyze the throughput and delay of the system proposed in Chapter two. This system makes use of the Carrier Sense Multiple Access and Collision Detection (CSMA/CD) scheme. In addition, data and acknowledgment transmissions share the same channel. We will first develop the model of an un-slotted CSMA/CD with acknowledgment(CSMA/CD with ACK). The effects of channel errors are also included in the model for two cases: without and with Reed-Solomon encoding/decoding. The throughput and delay performance of the system are derived under the Poisson finite user population arrival model.

3.1. THE ANALYTICAL MODEL

We consider a finite population of M users sharing a common channel. Each user possesses a single packet buffer. All data packets are assumed to have the same size, requiring T_t seconds for transmission. Acknowledgement (ACK) packet requires T_w seconds for transmission where $T_w < T_t$. Due to the short distance, the propagation delay between any two users is very small and almost constant τ. By setting the propagation delay τ equivalent to the largest possible distance, we gets lower (i.e. pessimistic) bounds on the performance. Every user follows the unslotted CSMA/CD with ACK protocol as described in Chapter 2. In order to gain insight into the relation between transmission of new packets and the retransmission of old ones we build the following packet scheduling model. Let every user be in one of two states- *thinking* and *backlogged*. In the thinking state, the user does not have a packet to transmit and does not take part in any scheduling activities. While in this state, the user generates a packet according to the identical

independent Poisson process with a transmission rate of a packets/second and senses the channel. If the channel is busy, the user will move to the backlogged state and the packet is rescheduled for transmission later on according to an exponential distribution with mean of $1/r$. On the other hand, if the channel is idle, the user will start the transmission immediately. If the transmission is successful, the user will remain in the thinking state and the packet generating process will be repeated. Otherwise, the user moves to the backlogged state and schedules the retransmission in the future by an exponential distribution with mean of $1/r$. While in the backlogged state, the user does not generate the packet because it has a packet in transmission or awaiting transmission. It remains in that state until the retransmission of the backlogged packet is successful. At that time it moves back to the thinking state. There is a possibility of collision when more than one user transmits the packets at the same time. We denote T_c as the time necessary for a user to detect the collision and abort the colliding transmission. After the collision, the user will move to the backlogged state and the collided packet is also rescheduled for retransmission in the future by an exponential distribution with mean of $1/r$ (Figure 3.1).

3.2. THROUGHPUT AND DELAY ANALYSIS

During the entire operation, the time on the channel alternates between idle and busy periods (Figure 3.2). The state of the system $n(t)$ is defined as the number of backlogged users at time t. The imbedded point of the system t_k where $k = 0,1,2,..$ is then assigned at the beginning of idle period (or the end of busy period). The time between two consecutive imbedded points (t_k and t_{k+1}) is called a cycle. The state of system at the imbedded point $n(t_{k+1})$ depends on the state of system at the previous imbedded point $n(t_k)$ and the number of users that move from one state to another within the cycle. Since the future state of the system depends upon the present but not upon the past $\{n(t_k), k=0,1,...\}$ is an imbedded Markov chain [5]. Because the number of backlogged users do not exceed M, this chain is also ergodic [6]. Therefore, the steady state

distribution exists. We will first obtain the steady state probability distribution, and then derive the channel throughput S and average delay D as a function of the channel traffic G.

3.2.1. Steady State Probability Distribution

Depending on the observation points in a cycle, the following parameters are used to describe the state of the system.

N_o: the number of backlogged users at the beginning of a cycle.

N_B: the number of backlogged users at the beginning of a busy period.

N_C: the number of backlogged users immediately after the first colliding user starts transmission.

N_τ: the number of backlogged users at τ units of time after the beginning of a busy period.

N_E: the number of backlogged users at the end of a cycle.

Now let us define the following probability elements:

π_i: the steady state probability of system in state i, i.e.

$$\pi_i = Pr\left\{n(t_k)=i\right\}. \tag{3.1}$$

P_{ij}: the steady state transition probability that the channel is moved from state i to state j, i.e.

$$P_{ij} = Pr\left\{n(t_{k+1}) = j \mid n(t_k) = j\right\}, 0 \le i,j \le M. \tag{3.2}$$

P: the matrix whose elements are P_{ij}'s.

$\boldsymbol{\pi}$: the row vector whose elements are π_i's.

Since $\left\{n(t_k), k=0,1,..\right\}$ is the ergodic Markov chain, the steady state probability vector $\boldsymbol{\pi}$ is the unique solution to the finite set of linear equations [3.2]

$$\pi = \pi P, \ with \quad \sum_{i=1}^{M} \pi_i = 1 \tag{3.3}$$

In order to obtain the steady state probability distribution of the system, we will construct the steady state transition probability matrix P as follows.

Let $I_i(t)$ and $I_i^*(s)$ denote the probability distribution function of idle time t and its Laplace transform given that the number of backlogged users at the imbedded point (or the beginning of the cycle) is i ($N_o = i$). We then have i backlogged users and $M-i$ thinking users at this imbedded point. Since the thinking users generate packets with a rate of a packets/sec and backlogged users are rescheduled packets with parameter r, the probability distribution function of packet generated by the thinking users, $P_T(t)$ and rescheduled by the backlogged users, $P_B(t)$ are given as,

$$P_T(t) = exp(-a(M-i)t), \tag{3.4a}$$
$$P_B(t) = exp(-rit). \tag{3.4b}$$

The probability distribution function of idle time is therefore,

$$\begin{aligned} I_i(t) &= 1 - P_T(t) P_B(t), \\ &= 1 - \exp(-[a(M-i)+ri]t), \end{aligned} \tag{3.5}$$

Its Laplace transform is given as,

$$I_i^*(s) = \frac{a(M-i) + ri}{s + a(M-i) + ri}. \tag{3.6}$$

In CSMA/CD with ACK scheme, when the state of the system changes from i to j, the transition probability P_{ij} can be decomposed into three elements, namely, successful transition probability, unsuccessful transition probability due to channel noise and interference, and unsuccessful transition probability due to collision. In the following paragraph, the formula for those transition probabilities are derived.

***a*- Successful Transition Probability:**

A successful transition probability represents the probability that a packet is suc-

cessfully transmitted during a cycle when the state of system changes from i to j. Depending on the type of the successful packet, we have the following two cases,

Case 1: The new arrival packet is successfully transmitted.

Before entering the busy period, the system is in the state i, i.e. $M-i$ thinking users and i backlogged users in the system. At at the beginning of the busy interval the overall arrival rate of the thinking users is $a(M-i)$ and the overall rescheduled rate of the backlogged users is ri. Let $P_N(t)$ be the probability that the transmission of a new packet is started at the beginning of the busy interval given that the state of system at the beginning of the cycle is i, i.e.

$$P_N(t) = Pr\left\{t \le t_k + I, N_B = i \mid N_o = i\right\}. \tag{3.7a}$$

This probability can be obtained as the ratio of the arrival rate of the thinking user over the total arrival rate of the system.

$$P_N(t) = \frac{a(M-i)}{a(M-i)+ri}. \tag{3.7b}$$

Next, in order to let the packet successfully transmit, no other station except itself will sense the channel during the propagation delay τ. Let $P_{\bar{s}}(t)$ be the probability that no station senses the channel during the propagation delay τ given that the the state of the beginning of the busy period is i, i.e.

$$P_{\bar{s}}(t) = Pr\left\{t \le \tau + t_k + I, N_\tau = i \mid N_B = i\right\}. \tag{3.8a}$$

This probability can be obtained as a product of the probability that no thinking user excepts itself senses the channel and the probability that no backlogged user senses the channel during the propagation delay τ.

$$\begin{aligned} P_{\bar{s}}(t) &= exp\,(-a(M-i-1)\tau)\; exp\,(-ri\,\tau), \\ &= exp\,(-(a(M-i-1)+ri)\tau). \end{aligned} \tag{3.8b}$$

During the transmission of the successful packet, the length of transmission period T is equal to the sum of T_i, τ and T_w ($T=T_i + T_w + 2\tau$). Let $P_{acc}(t)$ be the probability that $j-i$ new packets are generated during this transmission time T and the state of the system changes from N_τ to N_E, i.e.

$$P_{acc}(t) = Pr\left\{t \le T+\tau+t_k+I, N_E=j \mid N_\tau=i\right\}. \tag{3.9a}$$

During the cycle, the state of channel is changing from i to j. We then have total of $j-i$ out of $M-i-1$ new packet arrived during this successful transmission period. The probability $P_{acc}(t)$ is given as,

$$P_{acc}(t) = \begin{Bmatrix} M-i-1 \\ j-i \end{Bmatrix} P_{t_T}^{\ j-i} (1-P_{t_T})^{M-j-1}. \tag{3.9b}$$

where P_{t_T} is the probability that a packet is generated during the successful transmission period and is given as,

$$P_{t_T} = 1-exp(-aT). \tag{3.9c}$$

Finally, in order to take into account the interference of noise over the access channel, we recall that in chapter 2, the probability Q of no error during the transmission of the packet was experimental computed. This probability Q will be used as a parameter to evaluate the interference of random noise on the transmission packet.

Let sc_{ij}^n be the probability that the state of the channel changes from i to j and that a new packet is successfully transmitted during a transmission time. Then, this probability can be obtained as the product of $P_N(t), P_{\bar{s}}(t), P_{acc}(t)$, and Q, as the events are jointly independent [7].

$$sc_{ij}^n = \Pr\{\text{New arrival packet is successfully transmitted}, N_E=j \mid N_o=i\}$$
$$= P_N(t)\, P_{\bar{s}}(t)\, P_{acc}(t)\, Q\ .$$

$$sc_{ij}^n = \frac{a(M-i)}{a(M-i)+ri} exp(-(a(M-i-a)+ri)\tau)$$

$$\binom{M-i-1}{j-i}(1-exp(-aT))^{j-i}\,exp(-aT)^{M-j-1}Q,$$

$$(0 \le i \le M-1, i \le j \le M-1)$$

$$sc_{ij}^{n} = 0 \quad Otherwise. \tag{3.10}$$

Case 2: The retransmission packet is successfully transmitted.

Using the same arguments as described above, we have,

$P_r(t)$ = Pr{a retransmission packet is started}

$$= Pr\left\{t \le t_k + I, N_B = i-1 \mid N_o = i\right\}.$$

$$= \frac{ri}{a(M-i)+ri}. \tag{3.11}$$

$P_{\bar{s}}(t)$ = Pr{no station senses the channel during a propagation delay}

$$= \Pr\{\, t \le t_k + I, N_\tau = i-1 \mid N_B = i-1 \,\}$$

$$= exp(-(a(M-i)+r(i-1)))\tau. \tag{3.12}$$

$P_{acc}(t)$ = Pr{$j-i+1$ new packets are generated during the successful transmission time T}

$$= \Pr\{\, t \le T+\tau+t_k+I, N_E = j \mid N_\tau = i-1 \,\}$$

$$= \binom{M-i}{j-i+1}(1-exp(-aT))^{j-i+1}exp(-aT)^{M-j-1}. \tag{3.13}$$

Pr{no error during the successful transmission of the packet} = Q. (3.14)

Let sc_{ij}^{b} denote the probability that the channel state changes from i to j and a retransmission packet is successfully transmitted during the transmission time. Then, this probability can be obtained as follow [8]

sc_{ij}^{b} = Pr{Packet retransmission is successful, $N_E = j \mid N_o = i$}

$$= P_r(t)\,P_{\bar{s}}(t)\,P_{acc}(t)\,Q\,.$$

$$sc_{ij}^{b} = \frac{ri}{a(M-i)+ri}exp(-(a(M-i)+r(i-1))\tau)$$

$$\binom{M-i}{j-i+1}(1-exp(-aT))^{j-i+1}exp(-aT)^{M-j-1}Q,$$

$$(1 \le i \le M, i-1 \le j \le M-1)$$

$$sc_{ij}^{b} = 0 \quad Otherwise. \tag{3.15}$$

***b*- Unsuccessful Transition Probability Due to Channel noise and Interference:**

A unsuccessful transition probability due to the random noise is the probability that a packet is unsuccessfully transmitted during a cycle due to the the interference of random noise on the transmitted packet when the state of channel changes from i to j. Depending on the types of unsuccessful packet, we have the following two cases.

Case 1: The new arrival packet is unsuccessfully transmitted

Using the same arguments as described above, we have,

$P_N(t)$ = Pr{a transmission of the new arrival packet is started}

$$= \Pr\{ t \le t_k + I, N_B = i \mid N_o = i \}$$

$$= \frac{a(M-i)}{a(M-i)+ri}. \tag{3.16}$$

$P_{\bar{s}}(t)$ = Pr{no station senses the channel during a propagation delay}

$$= \Pr\{ t \le t_k + I, N_\tau = i \mid N_B = i \}$$

$$= exp(-(a(M-i-1)+ri)\tau). \tag{3.17}$$

$P_{acc}(t)$ = Pr{$j-i$ new packets are generated during the successful transmission time Tu }

$$= \Pr\{ t \le T_u + \tau + t_k + I, N_E = j \mid N_\tau = i \}$$

$$= \binom{M-i-1}{j-i-1}(1-exp(-aT_u))^{j-i-1}exp(-aT_u)^{M-j}. \tag{3.18}$$

where T_u is the length of transmission period during the transmission of the unsuccessful packet having a time out delay T_d and is given as

$$T_u = T_i + T_d + \tau. \tag{3.19a}$$

Pr{errors occur during the successful transmission of the packet} = $1-Q$. (3.19*b*)

Let f_{ij}^{nT} be the probability that the state of channel changes from i to j and that a new packet is unsuccessfully transmitted during a transmission time due to the error in the channel. In such a case, this probability can be obtained as follows [8]:

$f_{ij}^{nT} = \Pr\{\text{a new arrival packet is unsuccessfully transmitted}, N_E = j \mid N_o = i\}$

$= P_N(t)\, P_{\bar{s}}(t)\, P_{acc}(t)\,(1-Q)$.

$$f_{ij}^{nT} = \frac{a(M-i)}{a(M-i)+ri} exp\,(-(a(M-i-1)+ri)\tau) \begin{bmatrix} M-i-1 \\ j-i-1 \end{bmatrix} (1-exp\,(-aT_u))^{j-i-1}\ (exp\,(-aT_u))^{M-j}\ (1-Q),$$

$$(0 \le i \le M-1, i+1 \le j \le M)$$

$$f_{ij}^{nT} = 0 \quad Otherwise. \tag{3.20}$$

Case 2: A retransmission packet is unsuccessfully transmitted

Using the same arguments as described above, we have,

$P_r(t) = \Pr\{\text{a transmission of the retransmission packet is started}\}$

$= \Pr\{ t \le t_k + I, N_B = i-1 \mid N_o = i \}$

$$= \frac{ri}{a(M-i)+ri}. \tag{3.21}$$

$P_{\bar{s}}(t) = \Pr\{\text{no station senses the channel during a propagation delay}\}$

$= \Pr\{ t \le t_k + I, N_\tau = i-1 \mid N_B = i-1 \}$

$$= exp\,(-(a(M-i)+r(i-1))\tau). \tag{3.22}$$

$P_{acc}(t) = \Pr\{j-i \text{ new packets are generated during the successful transmission time } Tu\}$

$= \Pr\{ t \le T_u + \tau + t_k + I, N_E = j \mid N_\tau = i-1 \}$

$$= \begin{bmatrix} M-i \\ j-i \end{bmatrix} (1-exp\,(-aT_u))^{j-i}\, exp\,(-aT_u)^{M-j}. \tag{3.23}$$

$\Pr\{\text{errors occur during the successful transmission of the packet}\} = 1-Q.$ (3.24)

Let f_{ij}^{bT} be the probability that the state of channel changes from i to j and that a

retransmission packet is unsuccessfully transmitted during a transmission time due to the error in the channel. This probability can be obtained as follows [8]

f_{ij}^{bT} = Pr{a retransmission packet is unsuccessfully transmitted, $N_E = j \mid N_o = i$ }

$= P_r(t)\, P_{\bar{s}}(t)\, P_{acc(t)}(1-Q)$.

$$f_{ij}^{bT} = \frac{ri}{a(M-i)+ri} exp\,(-(a(M-i-1)+ri)\tau)$$
$$\begin{bmatrix} M-i \\ j-i \end{bmatrix} (1-exp\,(-aT_u))^{j-i} \;\; (exp\,(-aT_u))^{M-j}\,(1-Q),$$

$$(1 \le i \le M, i \le j \le M)$$

$$f_{ij}^{bT} = 0 \quad Otherwise. \tag{3.25}$$

***c* - Unsuccessful Transition Probability Due to the Packet Collision:**

The unsuccessful transition probability due to the packet collision is the probability that a packet is unsuccessfully transmitted due to the packet overlapping error when the state of channel changes from i to j. In the CSMA/CD with ACK system, the behavior of the model when the collision occurs can be completely described by specifying the kinds of two users: the initial user who initiates the busy period, and the collided user who firstly collides with the ongoing transmission. In addition, depending on the type of the transmission packet, each initial or collided user can be in either two cases: a user with a new arrival packet or a user with a retransmission packet Therefore, we have four different cases of collisions as follow.

Case 1: The initial user with the New Arrival Packet Collides with the Collided User with the New Arrival Packets

Using the same argument as described before, we have,

$P_N(t)$ = Pr{the transmission of a new packet is started}

$$= \Pr\{\, t \le t_k + I, N_B = i \mid N_o = i \,\}$$

$$= \frac{a(M-i)}{a(M-i)+ri}. \tag{3.26}$$

$P_s(t)$ = Pr{ At least one station besides the initial station senses the channel during a propagation delay}

$$= \Pr\{\, t \le t_k + I, N_\tau = i \mid N_B = i \,\}$$

$$= 1 - exp\,(-(a(M-i-1)+ri)\tau). \tag{3.27}$$

$P_{N'}(t)$ = Pr{ the transmission of other new packet is started}

$$= \Pr\{\, t \le t_k + I, N_C = i \mid N_\tau = i \,\}$$

$$= \frac{a(M-i-1)}{a(M-i-1)+ri}. \tag{3.28}$$

$P_{acc}(t)$ = Pr{ $j-i-2$ new packets are generated during the collision detection time T_c }

$$= \Pr\{\, t \le T_c + \tau + t_k + I, N_E = j \mid N_C = i \,\}$$

$$= \begin{bmatrix} M-i-2 \\ j-i-2 \end{bmatrix} (1 - exp\,(-a(T_c+\tau)))^{j-i-2} exp\,(-a(T_c+\tau))^{M-j}. \tag{3.29}$$

Let ftt_{ij}^{nTc} be the probability that the state of channel changes from i to j and that a new packet is unsuccessfully transmitted during a transmission time due to the collision with another new packet. This probability can be obtained as follows [8]:

ftt_{ij}^{nTc} = Pr{ a initial user with a new arrival packet collides with a collided user with retransmission packet, $N_E = j \mid N_o = i$ }

$$= P_N(t)\, P_s(t)\, P_{N'}(t)\, P_{acc}(t).$$

$$ftt_{ij}^{nTc} = \frac{a(M-i)}{a(M-i)+ri}\,[1 - exp\,(-(a(M-i-1)+ri)\tau)]$$

$$\frac{a(M-i-1)}{a(M-i-1)+ri}$$

$$\begin{bmatrix} M-i-2 \\ j-i-2 \end{bmatrix} (1 - exp\,(-a(Tc+\tau)))^{j-i-2} exp\,(-a(Tc+\tau)(M-j)),$$

$$(0 \le i \le M-2, i+2 \le j \le M)$$

$$ftt_{ij}^{nTc} = 0 \quad Otherwise. \tag{3.30}$$

Case 2: The initial user with the New Arrival Packet Collides with the Collided user

with the retransmission packet.

Using the same argument as described before, we have,

$P_N(t)$ = Pr{the transmission of a new packet is started}

$$= \Pr\{\, t \le t_k + I, N_B = i \mid N_o = i \,\}$$

$$= \frac{a(M-i)}{a(M-i)+ri}. \tag{3.31}$$

$P_s(t)$ = Pr{At least one station besides the initial station senses the channel during a propagation delay}

$$= \Pr\{\, t \le t_k + I, N_\tau = i \mid N_B = i \,\}$$

$$= 1 - exp\,(-(a(M-i-1)+ri)\tau). \tag{3.32}$$

$P_r(t)$ = Pr{the transmission of other retransmission packet is started}

$$= \Pr\{\, t \le t_k + I, N_C = i-1 \mid N_\tau = i \,\}$$

$$= \frac{ri}{a(M-i-1)+ri}. \tag{3.33}$$

$P_{acc}(t)$ = Pr{$j-i-1$ new packets are generated during the collision detection time T_c }

$$= \Pr\{\, t \le T_c + \tau + t_k + I, N_E = j \mid N_C = i-1 \,\}$$

$$= \binom{M-i-1}{j-i-1} (1 - exp\,(-a\,(T_c+\tau)))^{j-i-1} exp\,(-a\,(T_c+\tau))^{M-j}. \tag{3.34}$$

Let ftb_{ij}^{nTc} be the probability that the state of the channel changes from i to j and that a new packet is unsuccessfully transmitted during a transmission time due to the collision with another retransmission packet. This probability can be obtained as follows [8]

ftb_{ij}^{nTc} = Pr{a initial user with a new arrival packet collides a collided user with a retransmission packet, $N_E = j \mid N_o = i$ }

$$= P_N(t)\, P_s(t)\, P_r(t)\, P_{acc}(t).$$

$$ftb_{ij}^{nTc} = \frac{\dfrac{a(M-i)}{a(M-i)+ri}}{\dfrac{ri}{a(M-i-1)+ri}} \left[1 - exp\,(-(a(M-i-1)+ri)\tau)\right]$$

$$\binom{M-i-1}{j-i-1}(1-exp(-a(Tc+\tau)))^{j-i-1}exp(-a(Tc+\tau)(M-j)),$$

$$(1 \le i \le M-1, i+1 \le j \le M)$$

$$ftb_{ij}^{nTc} = 0 \quad Otherwise. \tag{3.35}$$

Case 3: The initial User With a Retransmission Packet Collides with the Collided User with a New Arrival Packet:

Using the same argument as described before, we have,

$P_r(t)$ = Pr{the transmission of a new packet is started}

$$= \Pr\{ t \le t_k + I, N_B = i-1 \mid N_o = i \}$$

$$= \frac{ri}{a(M-i)+ri}. \tag{3.36}$$

$P_s(t)$ = Pr{At least one station besides the initial station senses the channel during a propagation delay}

$$= \Pr\{ t \le t_k + I, N_\tau = i-1 \mid N_B = i-1 \}$$

$$= 1-exp(-(a(M-i)+r(i-1)))\tau. \tag{3.37}$$

$P_N(t)$ = Pr{the transmission of other new packet is started}

$$= \Pr\{ t \le t_k + I, N_C = i-1 \mid N_\tau = i-1 \}$$

$$= \frac{a(M-i)}{a(M-i)+r(i-1)}. \tag{3.38}$$

$P_{acc}(t)$ = Pr{$j-i-1$ new packets are generated during the collision detection time T_c }

$$= \Pr\{ t \le T_c + \tau + t_k + I, N_E = j \mid N_C = i-1 \}$$

$$= \binom{M-i-1}{j-i-1}(1-exp(-a(T_c+\tau)))^{j-i-1}exp(-a(T_c+\tau))^{M-j}. \tag{3.39}$$

Let fbt_{ij}^{nTc} be the probability that the state of channel changes from i to j and that a retransmission packet is unsuccessfully transmitted during a transmission time due to the collision with another new packet. This probability can be obtained as follows [8]:

fbt_{ij}^{nTc} = Pr{a initial user with retransmission packet collides with the collided user with

new arrival packet, $N_E = j \mid N_o = i \}$

$$= P_r(t)\, P_s(t)\, P_{N'}(t)\, P_{acc}(t).$$

$$fbt_{ij}^{nTc} = \frac{ri}{a(M-i)+ri} \cdot \frac{a(M-i)}{a(M-i)+r(i-1)}$$
$$[1-exp(-(a(M-i)+r(i-1))\tau)]$$
$$\begin{bmatrix} M-i-1 \\ j-i-1 \end{bmatrix} (1-exp(-a(Tc+\tau)))^{j-i-1} exp(-a(Tc+\tau)(M-j)),$$

$$(1 \le i \le M-1, i+1 \le j \le M)$$

$$fbt_{ij}^{nTc} = 0 \quad Otherwise. \tag{3.40}$$

Case 4: A initial user with a Retransmission Packet Collides with the Collided User with the Retransmission Packet.

Using the same argument as described before, we have,

$P_r(t)$ = Pr{the transmission of a retransmission packet is started}

$$= \Pr\{\, t \le t_k + I, N_B = i-1 \mid N_o = i \,\}$$

$$= \frac{ri}{a(M-i)+ri}. \tag{3.41}$$

$P_s(t)$ = Pr{At least one station besides the initial station senses the channel during a propagation delay}

$$= \Pr\{\, t \le t_k + I, N_\tau = i-1 \mid N_B = i-1 \,\}$$

$$= 1-exp(-(a(M-i)+r(i-1))\tau). \tag{3.42}$$

$P_{r'}(t)$ = Pr{the transmission of other retransmission packet is started}

$$= \Pr\{\, t \le t_k + I, N_C = i-2 \mid N_\tau = i-1 \,\}$$

$$= \frac{r(i-1)}{a(M-i)+r(i-1)}. \tag{3.43}$$

$P_{acc}(t)$ = Pr{$j-i$ new packets are generated during the collision detection time T_c }

$$= \Pr\{\, t \le T_c + \tau + t_k + I, N_E = j \mid N_C = i-2 \,\}$$

$$= \begin{bmatrix} M-i \\ j-i \end{bmatrix} (1-exp(-a(T_c+\tau)))^{j-i} exp(-a(T_c+\tau))^{M-j}. \tag{3.44}$$

Let fbb_{ij}^{nTc} be the probability that the state of channel changes from i to j and that a retransmission packet is unsuccessfully transmitted during a transmission time due to the collision with another retransmission packet. This probability can be obtained as follows [8]:

fbb_{ij}^{nTc} = Pr{a initial user with a retransmission packet collides with the collided user with a retransmission packet, $N_E = j \mid N_o = i$}

$= P_r(t)\, P_s(t)\, P_r(t)\, P_{acc}(t)$.

$$fbb_{ij}^{nTc} = \frac{ri}{a(M-i)+ri}\,[1-exp(-(a(M-i)+r(i-1))\tau)]$$

$$\frac{r(i-1)}{a(M-i)+r(i-1)}$$

$$\binom{M-i}{j-i}(1-exp(-a(Tc+\tau)))^{j-i}\, exp(-a(Tc+\tau)(M-j)),$$

$$(2 \le i \le M, i \le j \le M)$$

$$fbb_{ij}^{nTc} = 0 \quad Otherwise. \tag{3.45}$$

After obtaining all the state transition probabilities of each category, we will derive the state transition probability of the system P_{ij} from state i to j as follow.

$$p_{ij} = sc_{ij} + f_{ij}, \tag{3.46}$$

where

$$sc_{ij} = sc_{ij}^{n} + sc_{ij}^{b}, \tag{3.47}$$

$$f_{ij} = f_{ij}^{nT} + f_{ij}^{bT} + ftt_{ij}^{nTc} + ftb_{ij}^{Tc} + fbt_{ij}^{Tc} + fbb_{ij}^{Tc}. \tag{3.48}$$

Substitute each element of the matrix P by a state transition probability given in the equation (3.44) and solve the set of equations $\boldsymbol{\pi} = \boldsymbol{\pi} P$ *with* $\sum_{i=1}^{M} \pi_i = 1$. We obtain the

steady state probability of the system π_i.

3.2.2. Channel Throughput

By using the equations (3.10), (3.20), (3.25), (3.30), (3.35), and(3.40), the average throughput rate λ_t is obtained as the ratio of the time in which a packet is transmitted successfully during a cycle average (transmission time) over all cycles (thinking and back-logged time) to the average cycle time [8]. Thus

$$\lambda_t = \frac{\sum_{i=0}^{M} \pi_i sc_i}{\sum_{i=0}^{M} (\pi_i [\frac{1}{d_i} + sc_i T + f_i^{Tc} (Tc+\tau) + f_i^{T_*} + \tau])}, \tag{3.49a}$$

where

$$d_i = (M-i)a + ir. \tag{3.49b}$$

sc_i (or f_i) is the probability that transmission is successful (or unsuccessful) during a cycle given that the channel is in state i. We have,

$$sc_i = \sum_{j=i-1}^{M-1} sc_{ij}, \tag{3.50a}$$

$$f_i = f_i^{Tc} + f_i^{T}, \tag{3.50b}$$

where:

$$f_i^{Tc} = \sum_{j=i}^{M} (ftt_{ij}^{nTc} + ftb_{ij}^{Ntc} + fbt_{ij}^{nTc} + fbb_{ij}^{nTc}), \tag{3.50c}$$

$$f_i^{T} = \sum_{j=i}^{M} (f_{ij}^{nT} + f_{ij}^{bT}). \tag{3.50d}$$

Then the channel throughput S and traffic G of the system is given as,

$$S = \lambda_t T, \tag{3.51}$$

$$G = a\ T. \tag{3.52}$$

where T is the packet transmission time.

3.2.3. Average Transmission Delay

In this section, we will derive the average transmission delay D versus channel traffic G. Then by combing this result and Eqs (3.49a) through (3.52), we will obtain the Delay-traffic analysis of our system.

We recall that the imbedded Markov point of the system is assigned at the beginning of the idle period during the cycle. We define the packet delay as the time from the arrival of the tagged packet to the completion of its transmission. Depending on the arrival time and the success or failure in the transmission of the tagged packet, we have the following 2 cases.

Case 1: The tagged packet is immediately transmitted upon its arrival and completed its transaction during its first transmission (Figure 3.3). The Laplace transform of probability distribution function (LSPF) of the packet delay $W^{*(1)}(s)$ is given by

$$W^{*(1)}(s) = exp\,(-(T_i + \tau)s). \tag{3.53}$$

Recall T_i is the data packet transmission time.

Case 2: The tagged packet is not able to transmit immediately upon its arrival or if it is transmitted immediately upon its arrival, it can not complete the transaction during the first transmission. From Figure 3.4, we can see that the packet delay can be divided into two parts, namely, the initial delay and the main delay. The initial delay is the time from the arrival of the tagged packet to the first imbedded point. The main delay is the time from the first imbedded point to the completion of the tagged packet transmission. Assume that the system has j backlogged packet (including the tagged packet) at the first imbedded Markov point. We denote $D_j^*(s)$ be the LSPF of the main delay, and $W_j^*(s)$ be LSPF of the packet delay. Classifying the unsuccessful tagged packet according to its experienced initial delay, we have the following six subcases of the unsuccessful tagged

packet arrivals as follow.

Subcase 1: The tagged packet is immediately transmitted upon its arrival and unsuccessfully completed its transaction during the first transmission due to the random noise error. The LSPF of packet delay $W_j^{*(1)}(s)$ is obtained as.

$$W_j^{*(1)}(s) = exp\,(-(T_u+\tau)s)\,D_j^*(s). \tag{3.54}$$

Recall T_u is the length of unsuccessful transmission period due to the random noise error.

Subcase 2: The tagged packet initiates a busy period upon its arrival and collide with another packet while transmitting given that the state of channel changes from i to j in the cycle. We have the LSPF of packet delay $W_{ij}^{*(2)}(s)$ as

$$W_{ij}^{*(2)}(s) = X_i^*(s)\,exp\,(-(T_c+\tau)s)\,D_j^*(s). \tag{3.55a}$$

where $X_i^*(s)$ is the Laplace transform of X_1, which is the Pdf of time interval from the beginning of a busy period to the start of transmission of the first colliding user, and is given as

$$X_i^*(s) = \frac{a(M-i-1)+ri}{s+a(M-i-1)+ri}\,(1-exp\,(-(s+a(M-i-1)+ri)))$$
$$(1-exp\,(-a(M-i-1)+ri)) \qquad (0\le i\le M-1). \tag{3.55b}$$

Subcase 3: The tagged packet is transmitted immediately upon its arrival and firstly collides with the ongoing transmission. The LSPF of packet delay $W_j^{*(3)}(s)$ is given as

$$W_j^{*(3)}(s) = exp\,(-(T_c+\tau)s)\,D_j^*(s). \tag{3.56}$$

Subcase 4: The tagged packet arrives during the successful transmission period of the another packet. The LSpdf of the packet delay $W_j^{*(4)}(s)$ is obtained as

$$W_j^{*(4)}(s) = Z^*(s/T)\,D_j^*(s). \tag{3.57a}$$

where $Z^*(s/T)$ is the LS of $z(t/T)$, which is the Pdf of time interval from the arrival of the tagged packet to the end of the ongoing successful transmission, is given as

$$Z^*(s/T) = \frac{a}{s-a}\,\frac{exp\,(-dT)-\exp(-sT)}{1-exp\,(-dT)}. \tag{3.57b}$$

Subcase 5: The tagged packet arrives during the unsuccessful transmission period of another packet due to the packet collision error. The LSPF of packet delay $W_j^{*(5)}(s)$ is

expressed as

$$W_j^{*(5)}(s) = Z^*(s/T_c)\, D_j^*(s). \tag{3.58a}$$

where $Z^*(s/T_c)$ is the LS of $z(t/T_c)$, which is the Pdf of time interval from the arrival of the tagged packet to the end of ongoing unsuccessful transmission due to the packet collision error, is expressed as

$$Z^*(s/T_c) = \frac{a}{s-a}\,\frac{exp(-a(T_c+\tau))-\exp(-s(T_c+\tau))}{1-exp(-a(T_c+\tau))}. \tag{3.58b}$$

Subcase 6: The tagged packet arrives during the unsuccessful transmission period of another packet due to the random noise error. The LSPF of packet delay $W_j^{*(6)}(s)$ is given as

$$W_j^{*(6)}(s) = Z^*(s/T_u)\, D_j^*(s). \tag{3.59a}$$

where $Z^*(s/T_u)$ is the LS of $z(t/T_u)$, which is the Pdf of time interval from the arrival of the tagged packet to the end of ongoing unsuccessful transmission due to the random noise error, is expressed as

$$Z^*(s/T_u) = \frac{a}{s-a}\,\frac{exp(-dT_u)-\exp(-sT_u)}{1-exp(-dT_u)}. \tag{3.59b}$$

Given that the state of the system changes from i to j during the initial delay period, then there is only one way for the tagged packet to arrive at the user with probability sc_{ij}^n in the case 1; with probability f_{ij}^{nT} in subcase 1; with probabilities ftt_{ij}^{nTc} and ftb_{ij}^{nTc} in subcase 2; and with probabilities ftt_{ij}^{nTc} and fbt_{ij}^{nTc} in subcase 3. In the subcase 4, since we have $j-i$ packets accumulated at the end of the first transmission cycle with the probability sc_{ij}^n, and $j-i+1$ packets with probability sc_{ij}^b, there are $j-i$ ways for the tagged packets to arrive at the user with the probability sc_{ij}^n and $j+i+1$ ways with the probability sc_{ij}^b. In subcase 5, we have $j-i-2$ ways for the tagged packet arrived with the probability ftt_{ij}^{nTc}; $j-i-1$ ways with probabilities ftb_{ij}^{nTc} and fbt_{ij}^{nTc}; and $j-i$ ways with the probability fbb_{ij}^{nTc}. In subcase 6, these are $j-i$ ways for the tagged packets to arrive with the probability f_{ij}^{nT} and $j-i-1$ ways with the probability f_{ij}^{nb}. Let $A_i^{(m)}$ be the

numbers of backlogs over the cycle starting with the state i and classifying into case m (where $m=1$) and subcase n (where $m=n+1$ and $0 \le n \le 6$). We have,

$$A_i^{(m)} = \begin{cases} \sum_{j=i}^{M-1} sc_{ij}^{n}(T_i+\tau) & \text{if } m=1 \\ \sum_{j=i+1}^{M} f_{ij}^{nT}(T_u+\tau) & \text{if } m=2 \\ \sum_{j=i+1}^{M} (ftt_{ij}^{nTc}+ftb_{ij}^{nTc})(T_c+\tau)\tilde{X}_i & \text{if } m=3 \\ \sum_{j=i+1}^{M} (ftt_{ij}^{nTc}+fbt_{ij}^{nTc})(T_c+\tau) & \text{if } m=4 \\ \sum_{j=i-1}^{M} (sc_{ij}^{n}(j-i)+sc_{ij}^{b}(j-i+1))\tilde{T} & \text{if } m=5 \\ \sum_{j=i}^{M} (ftt_{ij}^{nTc}(j-i-2)+(ftb_{ij}^{nTc}+fbt_{ij}^{nTc})(j-i-1)+fbb_{ij}^{nTc}(j-i))\tilde{T}_c & \text{if } m=6 \\ \sum_{j=i}^{M} (f_{ij}^{nT}(j-i-1)+f_{ij}^{nT}(j-i)) & \text{if } m=7 \end{cases} \tag{3.60}$$

where $\tilde{X}_i$, $\tilde{T}$, $\tilde{T}_c$, and $\tilde{T}_u$ are the means of the random variable obeying the probability distributions giving by equations (3.55b), (3.57a), (3.58b), and (3.59b), respectively.

Denoted A_i be the expected sum of backlogs over the cycle starting with state i. We have,

$$A_i = A_{idle} + \sum_{m=1}^{7} A_i^{(m)}. \tag{3.61}$$

where A_{idle} is the average length over a cycle starting with state i and is given by

$$A_{idle} = i(1/(a(M-i)+ri)+sc_i T+f_i^T T_u+f_i^{Tc} T_c+\tau). \tag{3.62}$$

The average channel backlog $\bar{n}$ is computed as the ratio of the expected sum over all cycles to the average cycle length and is given as

$$\bar{n} = \frac{\sum_{i=0}^{M} \pi_i A_i}{\sum_{i=0}^{M} \pi_i (1/(a(M-i)+ri)+sc_i T + f_i^T T_u + f_i^{Tc} T_c + \tau)}. \quad (3.63)$$

By Little's result, the average packet delay D is obtained as

$$D = \frac{\bar{n}}{\lambda_t}. \quad (3.64)$$

3.2.4. Numerical Results And Discussions

The above analytical results are used to evaluate the throughput and delay performance of the system proposed in Chapter two. To allow a comparison of theoretical and experimental results to be presented in Chapter four, numerical values of system parameters are taken from the experimental set up as follows:

Number of users in the system $M = 3$.

The transmission time T of the data packet of 704 bits at 1200 baud rate, of the acknowledgment response of 363 bits at 1200 baud rate , and of the system delay is

$$T = 1.2078 \; seconds.$$

The time-out delay T_d is given as,

$$T_d = 0.6039 \; seconds.$$

The collision time T_c is calculated as the sum of the transmission time of the 2 SYN characters and the system delay. It is given as,

$$T_c = 0.4941 \; seconds.$$

The probability that no channel error occurs during the transmission (Q) is obtained from the measured results of four experimental configurations summarized in the Table 3.1.

For the case of 1 persistent CSMA/CD with ACK, the mean of the exponential

retransmission delay is equal to zero ($r \rightarrow \infty$). The radio propagation delay τ is negligible for indoor applications, τ=0. The $r\tau$ converges to a constant C. C is set to 0.3.

The numerical results for the throughput as a function of the channel traffic for four experimental configurations are plotted in Figures 3.5 and 3.6. The corresponding of delay-versus-channel-traffic curves are plotted in Figures 3.7 and 3.8.

Referring to Figures 3.5 and 3.6, one can see that the throughput performance of the system improved with the introduction of RS codec. In the experimental setup #1, the throughput performance for two cases, without RS codec (Q = 0.993) and with RS codec (Q = 0.996) is almost identical as the improvement in Q is negligible (from Q = 0.993 to Q = 0.996). The throughput improvement is less than 0.3%. On the other hand, in the experimental setup # 2, the throughput improvement with RS codec is noticeable from 67.5% (without RS codec) to 89% (with RS codec). Throughput improvement also implies delay improvement as shown in Figure 3.7 and 3.8. In the test setup # 1, the delay improvement with RS codec is small (under 3%). The delay improvement becomes pronounced in the test setup # 2 (above 35%). This indicates that RS codec can be adaptively used to improve both throughput and delay performance when the channel error is high. From our experiments, channel error varies in time and location and is quite unpredictable. It is suggested that, in this case, an adaptive RS codec to change the error correcting capacity according to the channel error situation is an appropriate choice to achieve high throughput and short delay in a varying environment.

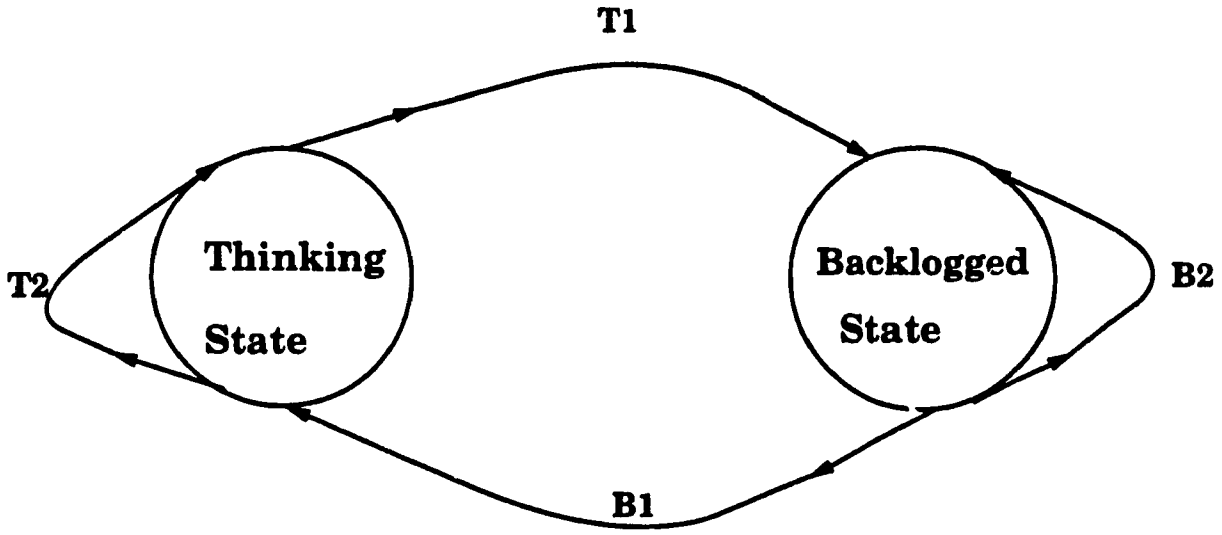


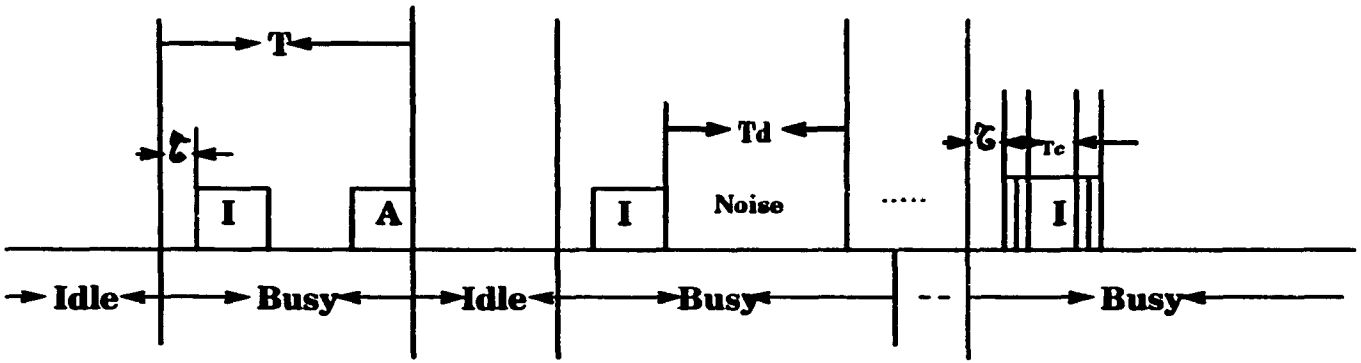
Figure 3.1 State transition of the CSMA/CD with ACK.

T1 : The Transmission of a newly generated packet is failed due to the packet collision or random noise on the channel.

T2 : The newly arrival packet is successfully transmitted or No packet is generated

B1 : Packet is successfully retransmitted

B2 : Awaited for the transmission or the retransmission is failed due to Collission or random noise on the channel



I : Information packet.
A: Acknowledgment packet.

Figure 3.2 : Idle and busy period in the unslotted CSMA/CD with ACK scheme.

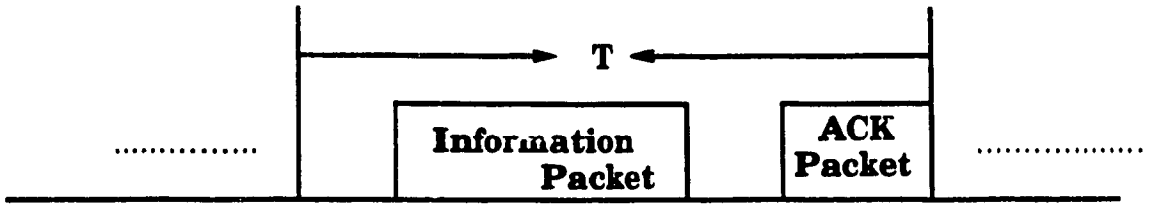


Figure 3.3 : The case 1 of the packet delay.

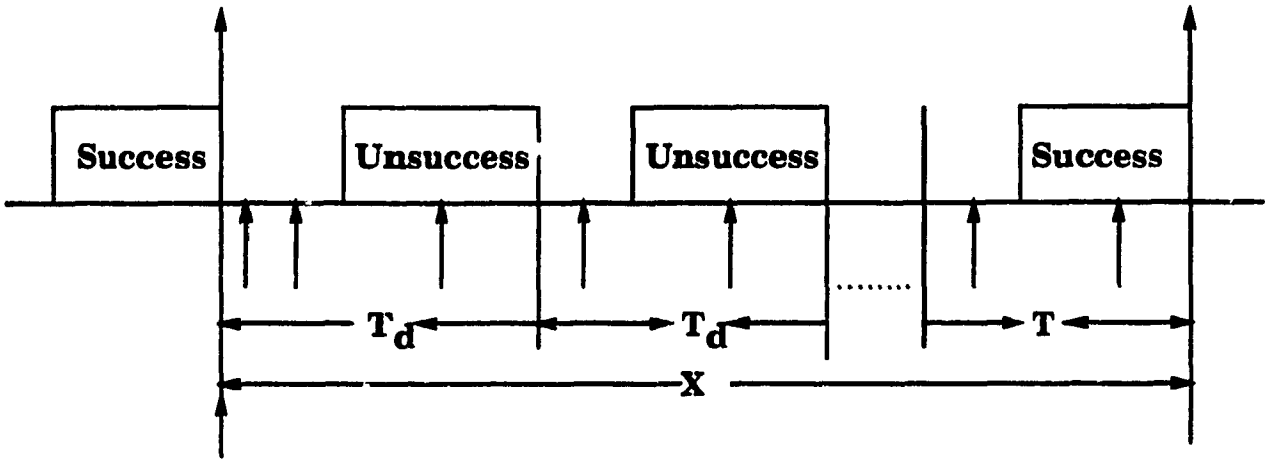


Figure 3.4 : The Subcase 1 to 5 of the packet delay.

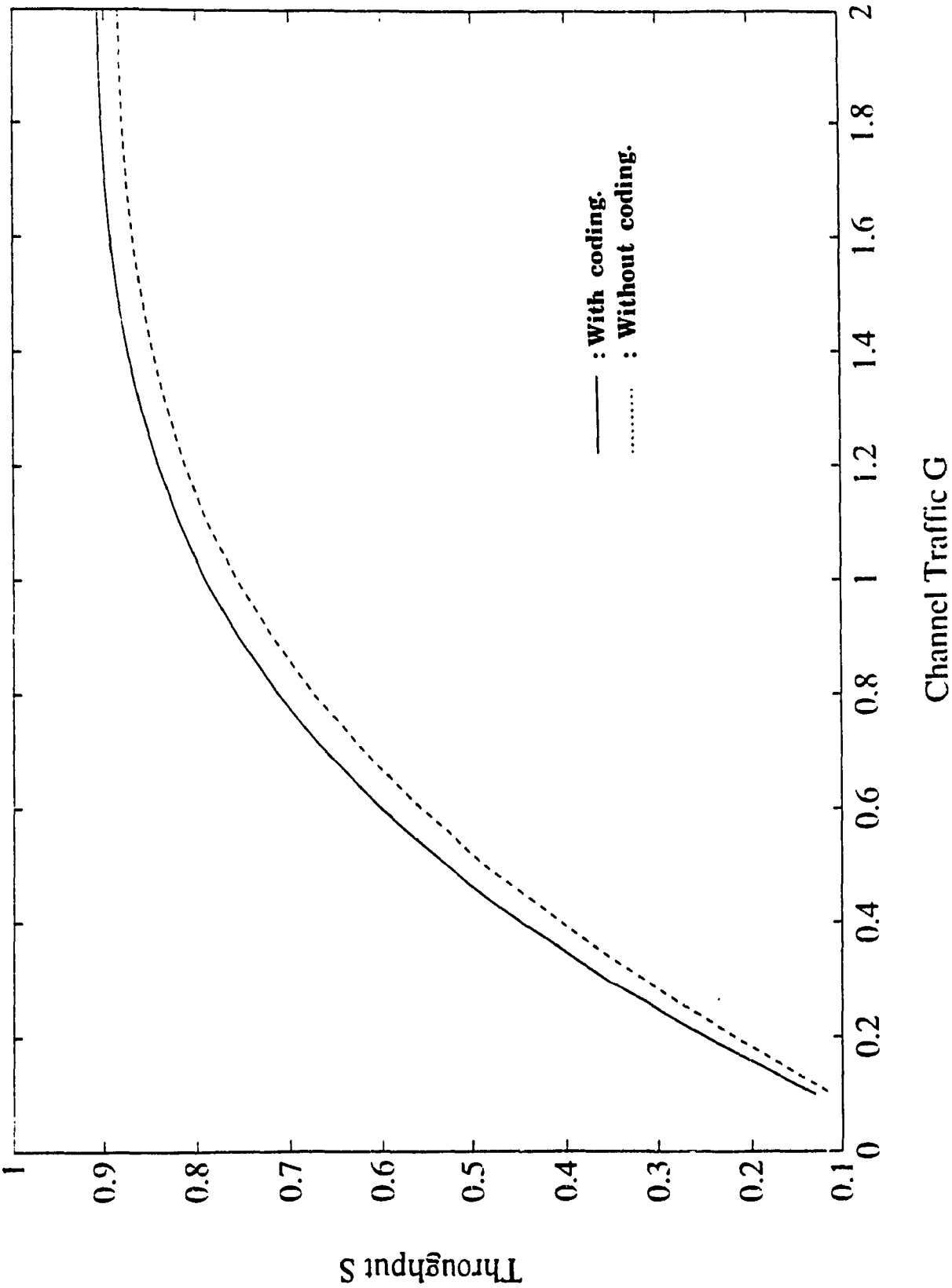


Figure 3.5 : Throughput Performance for the setup # 1

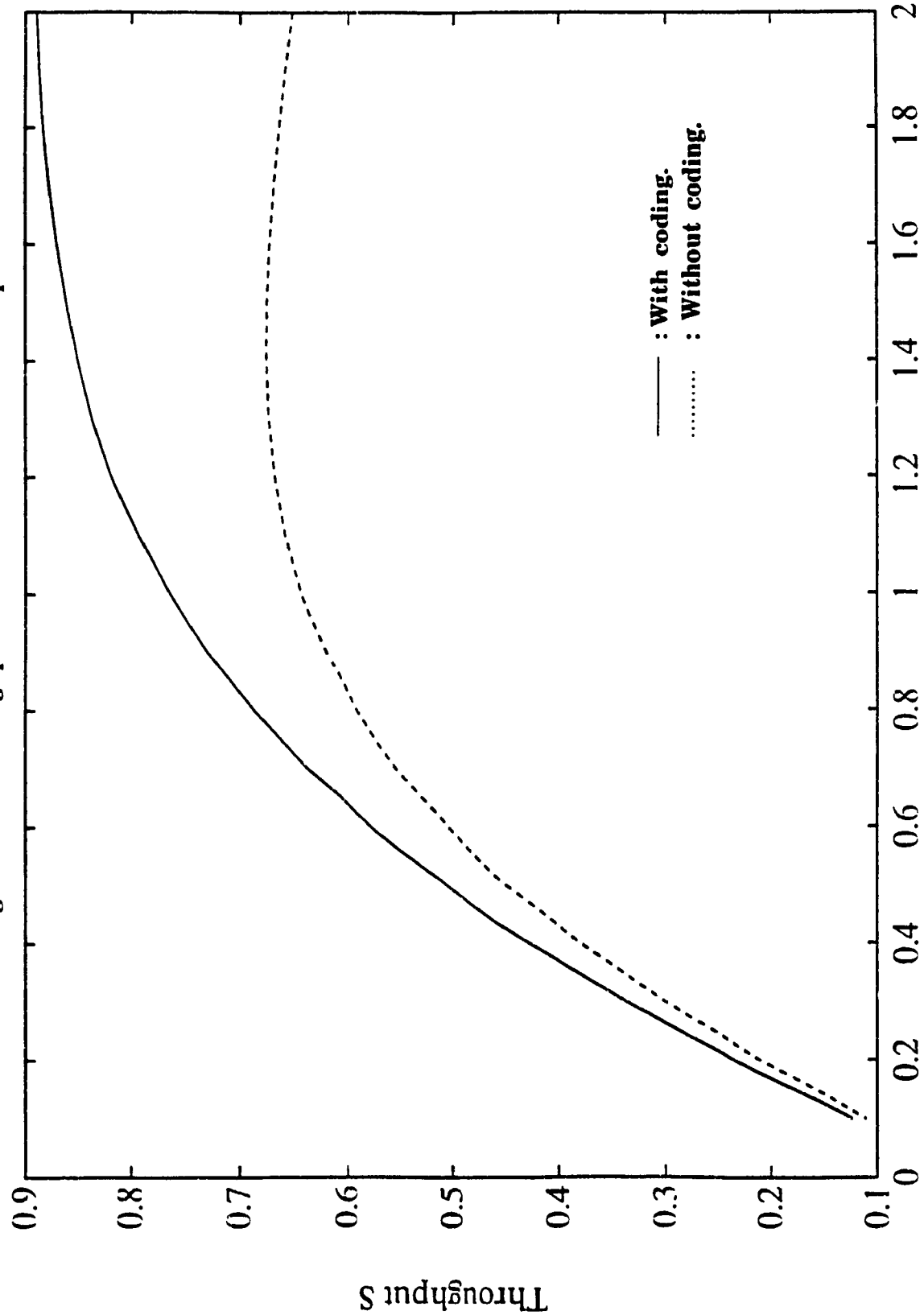


Figure 3.6 : Throughput Performance for the setup # 2

Figure 3.7 : Delay Performance for the setup # 1

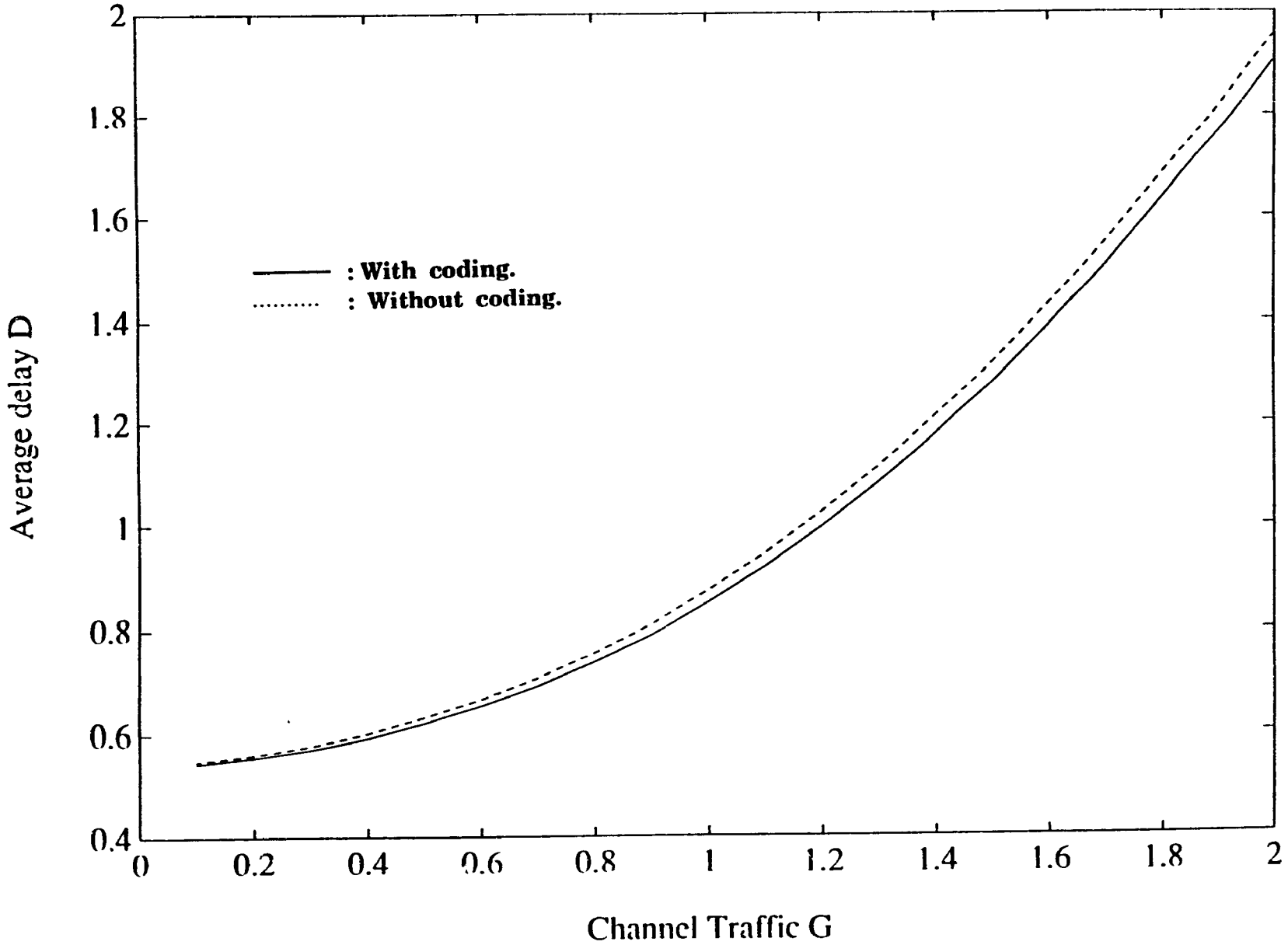
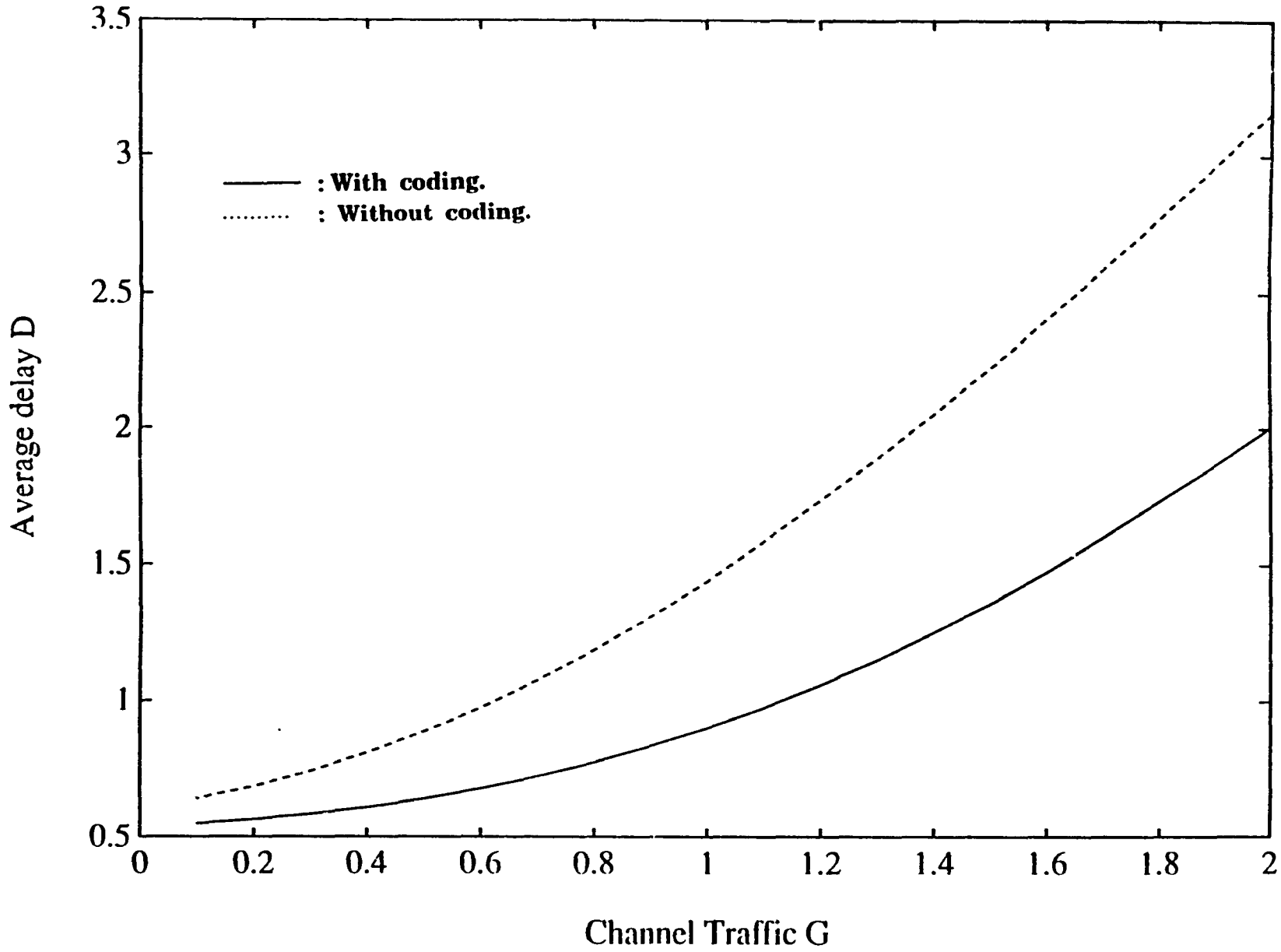


Figure 3.8 : Delay Performance for the setup # 2



Experimental Configuration	Without RS Codec	With RS Codec
Experimental Setup # 1	0.9927	0.9956
Experimental Setup # 2	0.9100	0.9889

Table 3.1 : Numerical values of Q .

CHAPTER 4

EXPERIMENTAL RESULTS ON SYSTEM PERFORMANCE

The objective of this chapter is to evaluate the system performance in real situations and to demonstrate the applicability of the proposed system. After describing the experimental set-up and its parameters, we explain the measurement techniques. Measured results are then compared to the analytical ones obtained in the previous chapter.

4.1. AN EXPERIMENTAL DATA COMMUNICATION SYSTEM

The previous chapter has provided the performance analysis of the CSMA/CD with ACK scheme and the analytical throughput and delay performance of the proposed system. An experimental data communications system based on the proposed system of Chapter two has been implemented in order to evaluate the wireless data communication system performance in a real indoor environment. The objectives are three fold: (i) to verify the analytical results, (ii) to demonstrate the applicability of the proposed system, (iii) to examine the design aspects.

4.1.1. The Experimental System

The experimental system consists of three personal computers (PC's) each equipped with the transceiver described in Chapter two. Data is transmitted in an Asynchronous, Character-Oriented mode at 1200 baud rate. The reason for using three computers is that we want to demonstrate a mesh topology. Recall that in a mesh topology every node in the network can communicate directly with every other one. The functional block diagram of each PC is illustrated in the Fig. 4.1. The Start-up delay is used to synchronize the execution of three PC's so that they can start running the test program at the same

time. The data packet is then generated by a packet generator. In the case of the coded packet, the newly generated data packet is encoded by a Reed-Solomon encoder (RK31KE) to form a sequence of output encoded data. Otherwise, the encoder will be bypassed. The final output data are sent out by a radio transmitter to the receiving station using the 1 persistent CSMA/CD with ACK scheme. At the receiver side, the receiving data are stored in the receiver buffer. Then, the decoder will decode the received data to form a receiving data packet if the coded scheme is used in the experiment, otherwise convert the receiving data directly to data packet. The header will be examined to determine whether the received packet is destinated to itself or not, and its type (Data or Ack). The packet receiver then verifies the correctness of the contents of the received packet. An appropriate response will be taken at the output of each block. This will be described in detail later on in this chapter. Finally, the output data will be stored into the output buffer and passed to the user.

4.1.2. Measurement Parameters

We define the measurement parameters of the experimental data communication system as follows:

Test Data Packet:

Two types of test data packet are used in this experiment. They are data packet and acknowledgment response packet.

(a) Data Packet

As shown in Figure 4.2, the data packet format consists of three parts. The first part contains two SYNC characters to indicate the beginning of the packet. The second part is the header. It consists of 6 fields containing the addresses of the sender and receiver, the type of the packet, the packet size, coding rate, and the Binary Cyclic Redundant Check-sum (BCC). The third part contains data packet. The size of the data packet is varied depending on the cases of uncoded or coded packet. In case of the uncoded packet, the packet size is equal to the sum of the header length and the information

length. In case of coded packet, because the header is encoded with an RS code (N,6), the packet size of the header is always equal to N, where N is the length of the code-word. The information, then, is fragmented into small segments. Each has K symbols and is encoded with an RS code (N,K) (Figure 4.3). Therefore, the packet size of the coded data packet is a multiple of N and always greater than the size of the uncoded one.

For simplicity of the experiment, the packet size of the uncoded data packet is set to be equal to the size of the coded one. The header in the uncoded data packet contains of N bytes consisted of six indicator bytes plus N - 6 arbitrary constant values. (The difference between a symbol and a byte is that a symbol consists of 5 bits and a byte consists of 8 bits). The packet size of uncoded information is also set to N (K bytes information, and (N-K) arbitrary constant values). The coded information will have K bytes. Therefore, the size of the coded data packet is equal to $2N$ which is exactly the same as the size of the uncoded data packet. Let T_i denote the transmission time of the data packet, then

$$T_i = \frac{N_c\, N_b}{f_b}. \tag{4.1}$$

where N_c is the number of character per packet, N_b is the total number of bits per character, and f_b is the baud rate (or number of bits per second) of the transceiver.

In the experiments, the values N and K are set to 31 and 20, respectively. Then the number of character per packet $N_{c/p}$ is equal to 62 (31 bytes for header and 31 bytes for information). As the data transmission is in an Asynchronous, Character-Oriented mode, each character contains seven information bits, one parity bit, one start bit and two stop bits or a total of eleven bits. The transmission bit rate is 1200 baud. This yields

$$T_i = (31*2)*11/1200 = 0.56833\, s.$$

(b) Acknowledgment Response Packet

As mentioned in Chapter 2, the acknowledgment response packet contains only the

SYNC characters and the header. The packet type in the header will indicate the type of the acknowledgment response. In the experiment, two types of acknowledgments are considered: Positive acknowledgment response (ACK) and Negative acknowledgment response (NAK). The positive acknowledgment response indicates that the transmitted packet is successfully received. Otherwise, the negative acknowledgment response indicates the error in the packet are detected by the receiver. For the simplicity in comparison, the acknowledgment packet is kept to be N character long. It consists of 6 indicator bytes (Fig. 4.2) and (N−6) dummy characters in case of uncoded packet. In case of coded packet, the (N−6) characters are for parity symbols in the RS (N,6) codeword. The transmission time of the acknowledgment response packet is

$$T_w = 31 * 11/1200 = 0.28427\ s.$$

Packet Transmission Time

Packet transmission time T is defined as the time interval from the transmission of the packet to the successfully receiving positive acknowledgment response (Figure 4.4). The time T, is the sum of the packet transmission time, T_i, the acknowledgment response time T_w, and the waiting time between the end of the transmitted packet and the beginning of the response packet. This waiting time is set to be 0.3553 s in the experiment. The packet transmission time T is therefore:

$$T = 0.56833+0.28427+0.3552=1.2078\ s.$$

Time-Out Delay

The sender expects to receive the acknowledgment response from the receiver within an interval T_d after the end of the packet transmission. If there is no response from the receiver within this time-out delay, the sender will assume the transmitted packet is lost and start retransmission. This value is set to be 0.6039 seconds in experiment.

Rate of Change

We assume that our traffic source consists of a finite number of users having an arrival rate of λ_s packet/second. In this experiment, due to the fact that we have only a

fixed number of users ($M=3$) in the system. Therefore, to vary the channel traffic source, we have to change the arrival rate λ_s. This value λ_s will be setup independently in each of the experimental test.

Measurement Time

Measurement time T_M is defined as a time required to carry out a whole experiment. It is set to 1798.85 seconds for all the experiments

Start-up Procedure

In order to avoid the channel capture phenomena when one computer starts before others, a startup procedure is introduced. In this startup procedure, we proceed the following steps.

Step 1: Synchronize the system clocks in all PCs.

Step 2: Setup a started timer in all PCs. The execution of the test program will be halted until the started timer is reached.

The startup procedure will synchronize the operation of all the PCs so that they can start running the test program at the same time.

4.1.3. Test Software and Its Operation:

The test software can be divided into five main modules: monitor, transceiver, timer, control-break, and RS codec.

Monitor Module

The operation of the monitor is shown by the flowchart in Figure 4.5. It initializes the communication port, presets the window timer, runs the startup procedure, and sets up the arrival rate λ_S (Fig. 4.7). It then generates the data packet with an arrival rate λ_s (box I in Fig. 4.5), randomly selects the destination address (box V in Fig. 4.5), senses the channel status (box II in Fig. 4.5), looks for the positive acknowledgment response (box III in Fig. 4.5), and finally counts the number of successful and unsuccessful packets over the entire experiment (box IV in Fig. 4.5). These numbers will be recorded in the output

data file before the program is terminated.

In order to create the interrupt routines for the transceiver and timer module, the interrupt vector table has been reinitialized such that interrupt I will be assigned to the module timer, interrupt III will be assigned to the transceiver module.

Transceiver Module

The flowchart of the transceiver module is shown in Figs. 4.8 through 4.11. It is composed of four main subroutines: transmission (Figure 4.8), character reception (Rx_char) (Figure 4.9), header reception (Rx_header) (Figure 4.10), and packet reception (Rx_packet) (Figure 4.11).

The main function of the transmission subroutine is to transmit the data to the receiver (box II in Figure 4.8). In order to differentiate between the noise and the start of the packet, two 'Start Of Text' (SOT) characters are sent out prior to the transmission of the test packet (box I in Figure 4.8).

The function of the Rx_char subroutine is to receive the data (box I in Figure 4.9) from the serial port, check for the start of text characters (box II in Figure 4.9), and call the subroutines Rx_header and Rx_packet to verify the correction of the received packet (box III in Figure 4.9). It also sets the status of the channel to busy and resets the window timer (box IV in Figure 4.9).

The function of the subroutine Rx_header is to check the correction of the header packet (box I in Figure 4.10), and the type of the packet (data packet or acknowledgment response packet) (box II in Figure 4.10). It also verifies that the received packet is actually destinated to its own station or not (box III in Figure 4.10).

The function of the Rx_packet subroutine is to check the correctness of the data packet (box I in Figure 4.11), obtain the packet, and prepare for the acknowledgment response (box II in Figure 4.11).

Timer Module

The flowchart for the timer module is shown in Figure 4.12. This module is used to keep track of the time-out delay T_a (box I in Figure 4.12), set the channel status to idle after the time-out T_d is expired (box II in Figure 4.12), and reset the window timer (box III in Figure 4.12). It also controls the timing for the whole operation (box IV in Figure 4.12).

Control-break Module

The control-break interrupt module (Figure 4.13) is used to terminate the execution of the program.

RS Codec Module

The flowchart of RS codec module is shown in Figure 4.14. Its function is to interface with the RS31KE/RS31KD codec board to encode the transmitted packets into a sequence of unsystematic code word (box I in Figure 4.14) and decode the receiving code words (box II in Figure 4.14). This RS31KD decoder chip can correct the ρ erasures and e errors in the packet according to the following formula,

$$\rho + 2e \leq N - K, \tag{4.2}$$

where N is the length of the code word, and K is the length of the information symbols. This RS codec module also has an interrupt routine. The interrupt IV has been assigned to this module. In conjunction with the hardware interrupt signal in the RS31KED codec board, a interrupt signal is generated every time a packet needs to be encoded or decoded. Two separate subroutines called Encoder and Decoder (Figures 4.15 and 4.16) are inserted into the monitor module (Figure 4.17) and the subroutines Rx_header (Figure 4.18) and Rx_packet (Figure 4.19) to enable and disable the interrupt of the RS codec module.

4.2. MEASUREMENT TECHNIQUES

In this section, we describe the method to collect the incoming data, the experimental results and channel parameters, and the experimental set-up configuration.

4.2.1. Data Collection

After receiving the incoming packets, the receiving station will perform the following steps (Figs. 4.1 and 4.5).

(i) Disregard the incoming data until two consecutive SYN characters are received.

(ii) Store the data into a buffer until the buffer length is equals to the size of the header (or = N in this test set-up). Then it will do the following.

* In case of coded packet, the incoming header data will be passed to the decoder.

* In case of uncoded packet, the decoder is bypassed.

The header data is then verified by a check-sum method (exclusive OR all the data). The receiving station will then examine the check-sum.

* If the check-sum is not equal to zero, the receiving packet will be discarded. Go back to (i).

* If the check-sum is equal to zero, the header data is considered correctly received.

After receiving the header correctly, the receiving station will compare the destination address of the receiving packet with its address.

* If the two addresses are not matched, the receiving packet will be rejected. Go back to (i).

* If two addresses are identical, the receiving station considers that the receiving packet is destinated to itself.

Next, the receiving station will check the type of the receiving packet.

*If it is a response type, the receiving station will either (a) schedule for the retransmission of the data packet if the response is negative (NAK), or (b) declare transaction is completed if the response is positive (ACK).

*If it is a data type, the receiving station will continue gathering data until the

buffer length is equals to the data packet size indicated in the header of the receiving packet. Go to step (iii).

(iii) The packet receiver then

* In case of coded packet, passes the incoming information data to the decoder.

* In case of uncoded packet, extracts the dummy bytes out of the information data.

The information data will then be compared with the predefined fixed information data. The receiving station will either (a) transmit a positive acknowledgment response packet back to the source user, if two data are identical, or (b) transmit a negative response packet otherwise.

After the process is completed, the information data will be stored into the buffer and passed to the user.

4.2.2. Derivation of Experimental Results

4.2.2.1. Number of Transmitted Packets:

The number of transmitted packets N_T is defined as the total number of packets transmitted to obtain a successful transaction. From Figure 4.5 we can see that the number of transmission N_T will be increased every time the data packet is ready for transmission. This transmission can be generated either by a newly arrival packet or by a retransmission packet. If the transaction is completed, the value N_T is stored into the output file and resetted at the beginning of each transmission of a newly arrival packet.

4.2.2.2. Performance Parameters

After carrying out the experimental test for about half an hour, the numbers of transmission are stored in the output data file. This file is then processed by another software program to yield the channel traffic, throughput and delay (Tables 4.2 to 4.5). The graph of throughput versus channel traffic and delay versus channel traffic for each of the computers are plotted in Figure 4.22 through 4.37. In the following, the

experimental channel traffic, throughput, and delay are calculated as follows:

Channel Traffic

If the packet transmission time lasts T seconds, the channel traffic G can be calculated as :

$$G = aT, \tag{4.3}$$

where a is the arrival rate and is calculated as :

$$a = \frac{N_{TT}}{T_M}, \tag{4.4}$$

where N_{TT} is the total number of packets transmitted during the test operation. It can be computed as:

$$N_{TT} = \sum_{i=1}^{3} N_T. \tag{4.5}$$

Channel Throughput

The formula of the experimental channel throughput can be calculated as follow:

$$S = \lambda_I T, \tag{4.6}$$

where

$$\lambda_I = \frac{N_{NT}}{T_M}, \tag{4.7}$$

and N_{NT} is the number of successful packets transmitted during the test operation and can be computed as

$$N_{NT} = \sum_{i=1}^{3} N_T(i). \tag{4.8}$$

Where $N_T(i)$ is the number of times in which the value N_T appeared in the output file.

Channel delay

If a transmission is successful, the transmitted packet is completely received after a transmission time T_i equation (4.1) equivalent to the packet length.

If a transmission is not successful due to the channel noise, the transmitter waits a time interval equal to T_i+T_d where T_d is the time-out delay. If a transmission is not successful due to the packet collision, the transmitter waits a time interval equal to T_i+T_c where T_c is the collision time. Therefore, the average delay is:

$$D = \frac{N_{NT}T_i+N_{Tr}T_i+N_{Tr}T_d+N_{Tc}T_i+N_{Tc}T_c}{N_{NT}}. \tag{4.9}$$

Where N_{Tr} is the number of unsuccessful packets due to the channel noise, and N_{Tc} is the number of unsuccessful packets due to the collision.

From the equations (4.5) and (4.8), we have:

$$N_{TT} = N_{NT}+N_{Tr}+N_{Tc}, \tag{4.10}$$

after some manipulation, we have:

$$D = T_i+(\frac{N_{Tr}}{N_{NT}})(T_i+T_d)+(\frac{N_{Tc}}{N_{NT}})(T_i+T_c). \tag{4.11}$$

Average Channel Throughput, Traffic and Delay

The average channel throughput, traffic and delay are defined as the average of all the experimental channel throughput, traffic and delay in each personal computer (PC), respectively.

System Throughput and Traffic

The System throughput and traffic of the system are calculated by adding all the average channel throughput and traffic of each PC, respectively.

Packet Delay

The packet delay of the system is obtained by averaging the number of the average channel delay in each PC.

4.2.3. Experimental Configurations

In order to investigate the effects of channel errors, two different sets of tests were performed: test configuration # 1 (test # 1), and test configuration # 2 (test # 2). In test # 1 (Figure 4.20), the three computers are placed in the same room (Concordia Hall Building H832-9). The distance between two PC's is 2 meters. As the distance is short, channel errors are low. In the test # 2 (Figure 4.21), the two computers are located in room H832-9, while the third one is located in room H832-5. The transmission is obstructed by the walls of these two different rooms, as illustrated in Figure 4.21. The distance is about 7 meters. The received signal is much weaker due to the walls. In addition, reflections and man-made interference introduce more channel errors. In each test configuration, we perform the test for both uncoded packet (uncoded test) and coded packet (coded test).

As shown in the Figure 4.1, the data packet is generated by a packet generator with an arrival rate λ_s. In test # 1, seven different values of λ_s are set up. They are 0.03046, 0.06072, 0.06648, 0.08318, 0.1111, 0.24956, and 0.9107. In test # 2, four different values of λ_s are used : 0.03046, 0.06072, 0.1656, and 0.9107.

A total of twenty two tests (7 uncoded test # 1, 7 of the coded test # 1, 4 of the uncoded test # 2, and 4 of the coded test # 2) have been carried out for all experiments.

The experimental results are summarized in Tables 4.2 to 4.5. Each table are divided into three parts, namely, description, content and result (Table 4.1).

The description part provides the information used to identifier the types of collective data. This includes:

(1) The media control method,

(2) The location in which the experiments took place, and

(3) The set-up configuration of the experimental system.

The content part shows the numerical data for the experimental setup. It includes:

(1) The arrival rate λs,

(2) The PC identifier, and

(3) The experimental data. It contains:

- The number of successful packets N_{NT} (column 1),
- The total number of transmission packets N_{TT} (column 2),
- The channel throughput S (column 3),
- The channel delay D (column 4),
- The channel traffic G (column 5),
- The average channel throughput (column 6),
- The average channel delay (column 7), and
- The average channel traffic (column 8).

Finally, the result part will show the total throughput, traffic, and the packet delay of the system.

4.3. MEASURED RESULTS AND DISCUSSIONS

Experimental results are statistically averaged to derive the values of the concerned parameters. Experiments were repeated for several runs. Obviously, as the number of runs approaches infinity, the obtained results averaged over the number of runs will converge to their statistical mean values. We are interested in determining the adequate number of runs to achieve experimental results of high confidence. To this end, the following table illustrates the variation in the measured result on the throughput averaged over the number of runs. Column 1 shows the number of runs. Column 2 indicates the throughput averaged for the number of runs. Column 3 displays the variation in two adjacent results.

Table 4.6: Illustrative results for Variations number of runs. (Test # 1; without Coding & arrival rate = 9.1074).

Number of Runs	Throughput	Variation
1	0.61719	----
2	0.64698	2.36%
3	0.72668	5.81%
4	0.77270	3.10%
5	0.79260	1.27%
6	0.80585	0.83%
7	0.79650	0.58%

It indicates that with more than 5 runs the obtained results is quite reliable within less than 1% variation.

The experimental results of the throughput and delay performance are summarized in Tables 4.2 and 4.5. and plotted in Figures 4.22 to 4.25.

In Figure 4.22, as the results are obtained in a good channel condition transmission, the experimental throughput performance for uncoded packet (plus dot line) and coded packet (cross dot line) is quite similar. The throughput improvement is less than 0.3%. On the other hand, from Figure 4.23, one can see that as the condition of the channel transmission gets worse, the coded packet (plus dot line) has better performance than the uncoded packet. The throughput improvement of the coded packet is about 21.5% higher than the throughput improvement of the uncoded packet. Also from the Figure 4.24, one can see that the average experimental packet delay for coded packet (plus dot line) and uncoded packet (cross dot line) are almost identical. But from the Figure 4.25, the average packet delay for coded packet is about 35% lower than the delay for uncoded packet. From the experiments, our channel error is nonstationary and unpredictable. This supports our conclusion given in previous Chapter 3 that the adaptive RS codec is a best technique to improve the performance of our system in a varying environment.

To evaluate the results of our experimental system, the comparison between the experimental and theoretical data are presented. From the Figs. 4.22 and 4.23, one can see that the experimental curves (dot lines) are very close to the theoretical ones (solid lines), the percent deviation of those curves are under 3%. This shows us that the theoretical model can be accurate in the case of throughput performance analysis.

In the case of delay performance analysis, from the Figures 4.24 and 4.25, one can see that in the low traffic region ($G < 0.6$), the experimental delay improvements (dot lines) are almost equal to the theoretical ones (solid lines). As the channel traffic increases, one sees that the experimental delay performances is worse than the theoretical one.

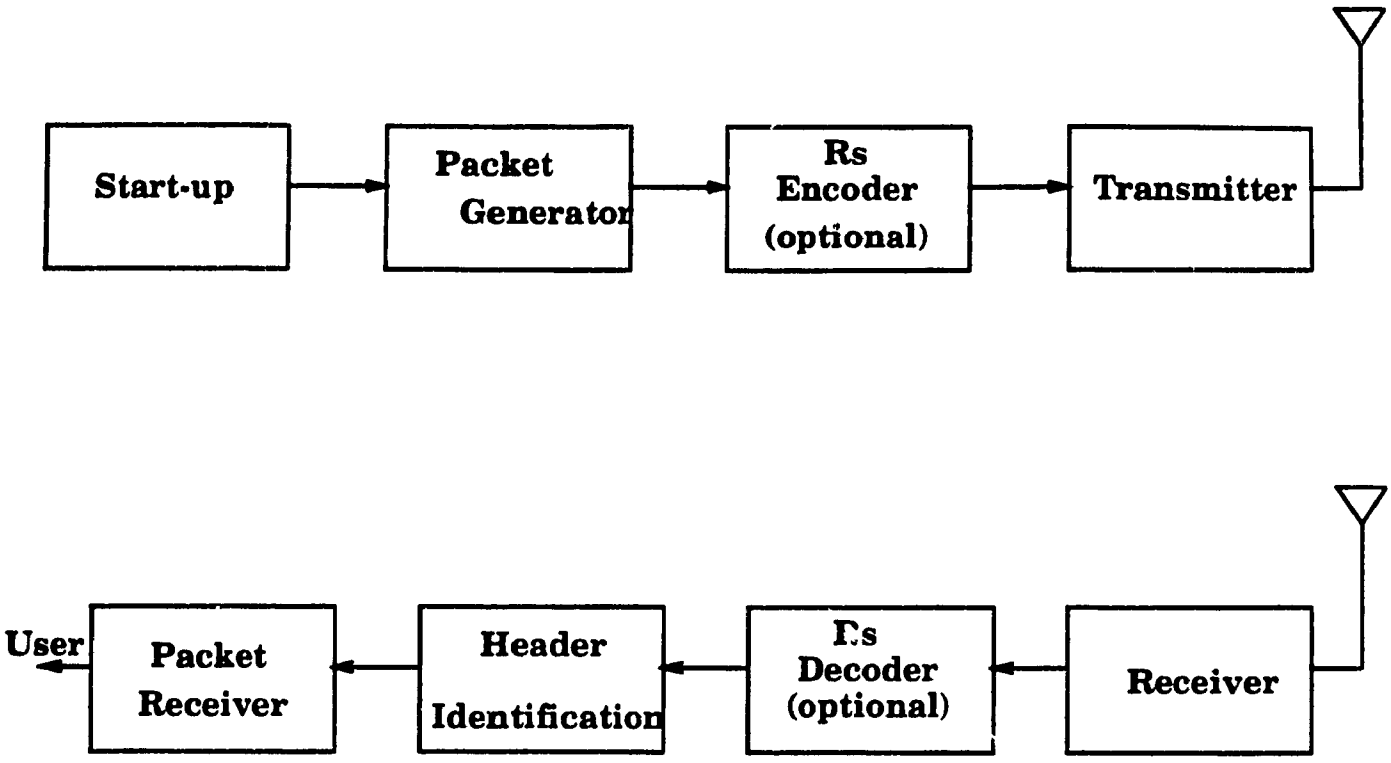


Figure 4.1 General Experimental System Block Diagram.

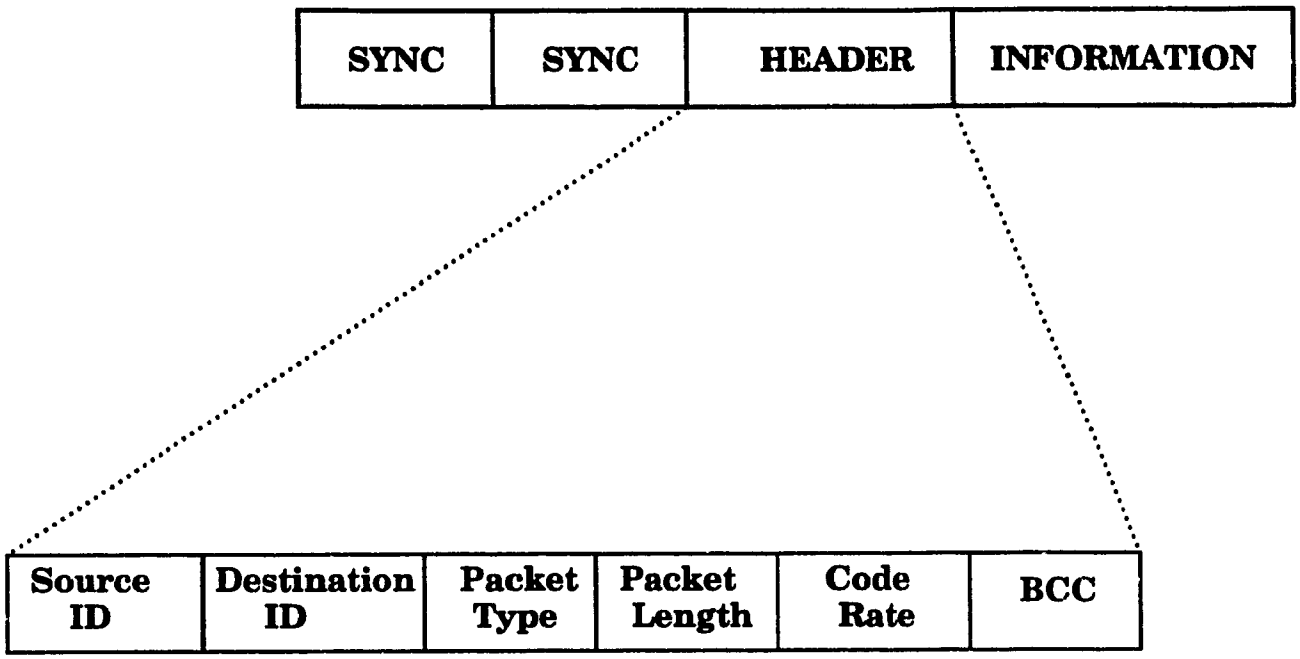


Figure 4.2 : General Packet Format.

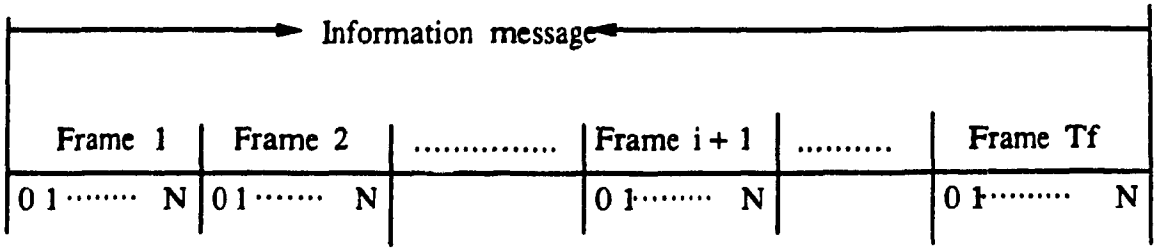


Figure 4.3 : The fragmentation of the information message

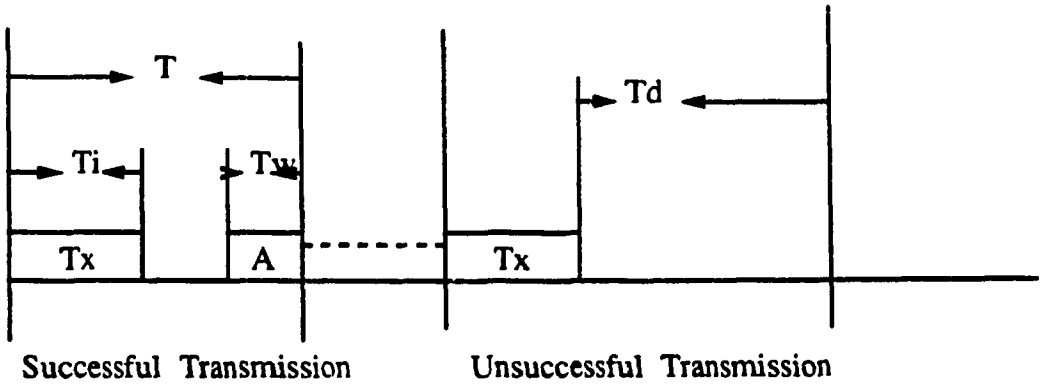


Figure 4.4 : The system timing description.

Tx : The Test data messages.

A : Acknowledgment response messages.

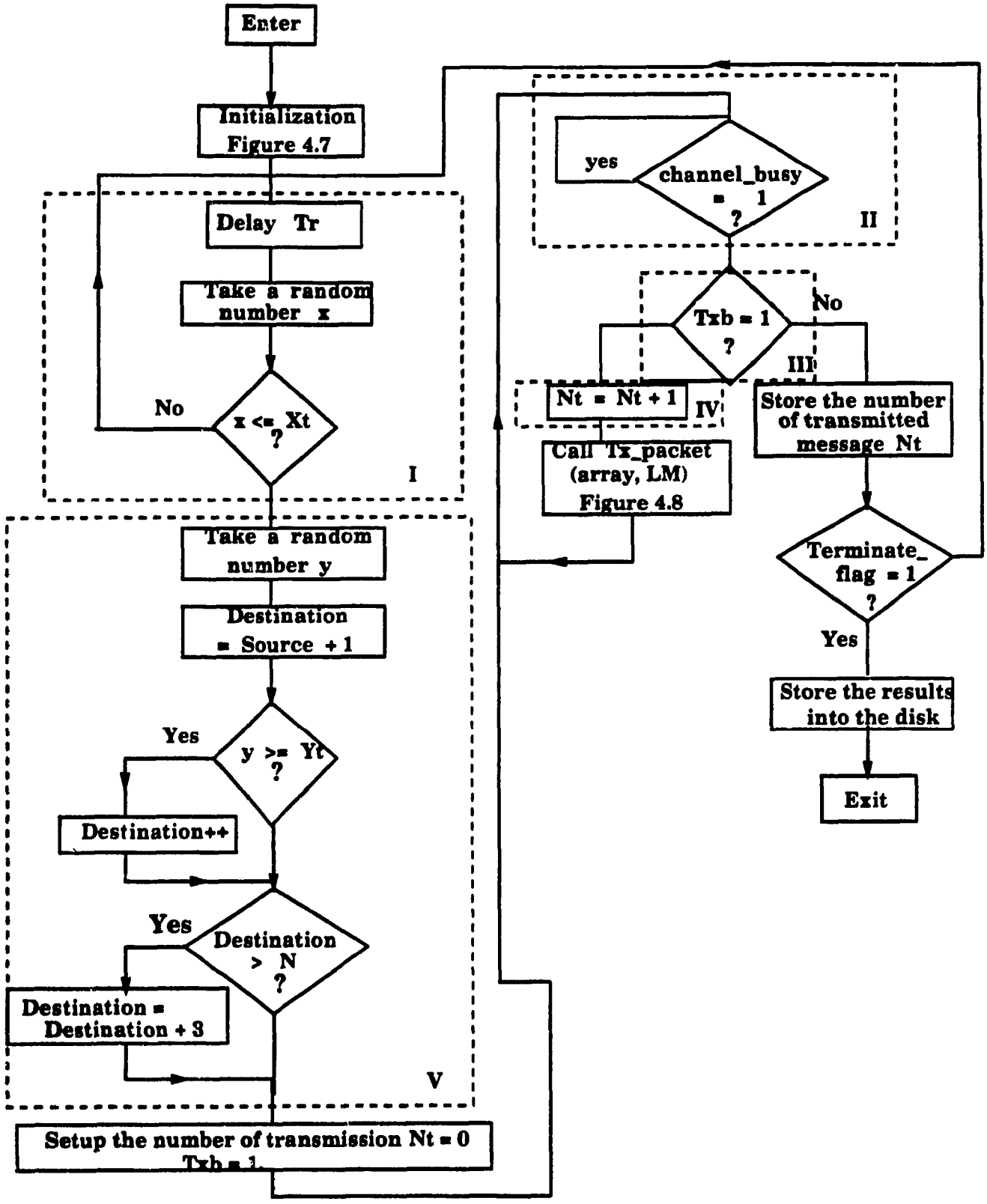


Figure 4.5 Monitor Module Uncoded Flow_chart

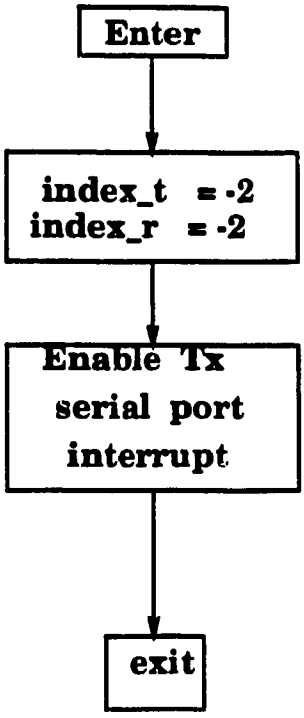


Figure 4.6 Procedure Tx_packet flow_chart

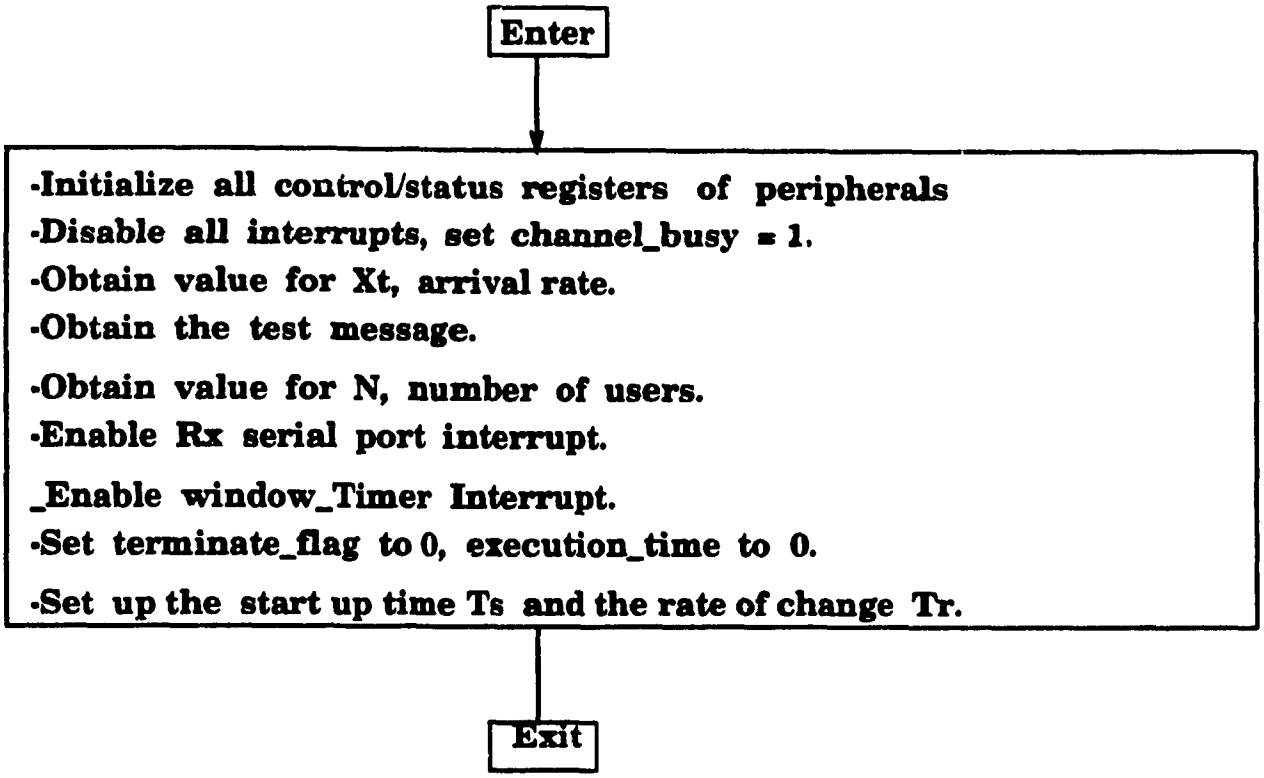


Figure 4.7 Procedure Initialization.

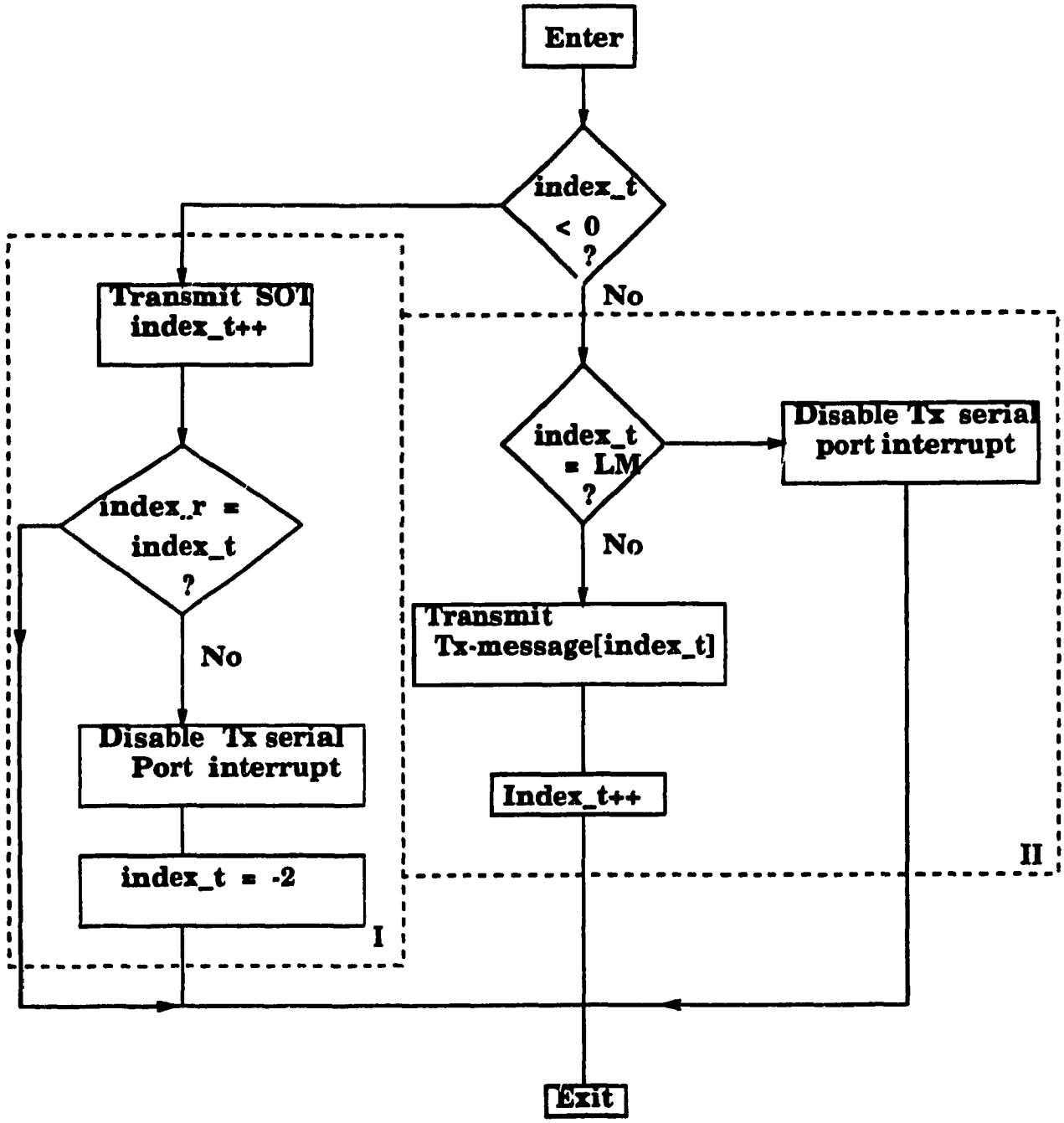


Figure 4.8 : Transmission Module flow_chart

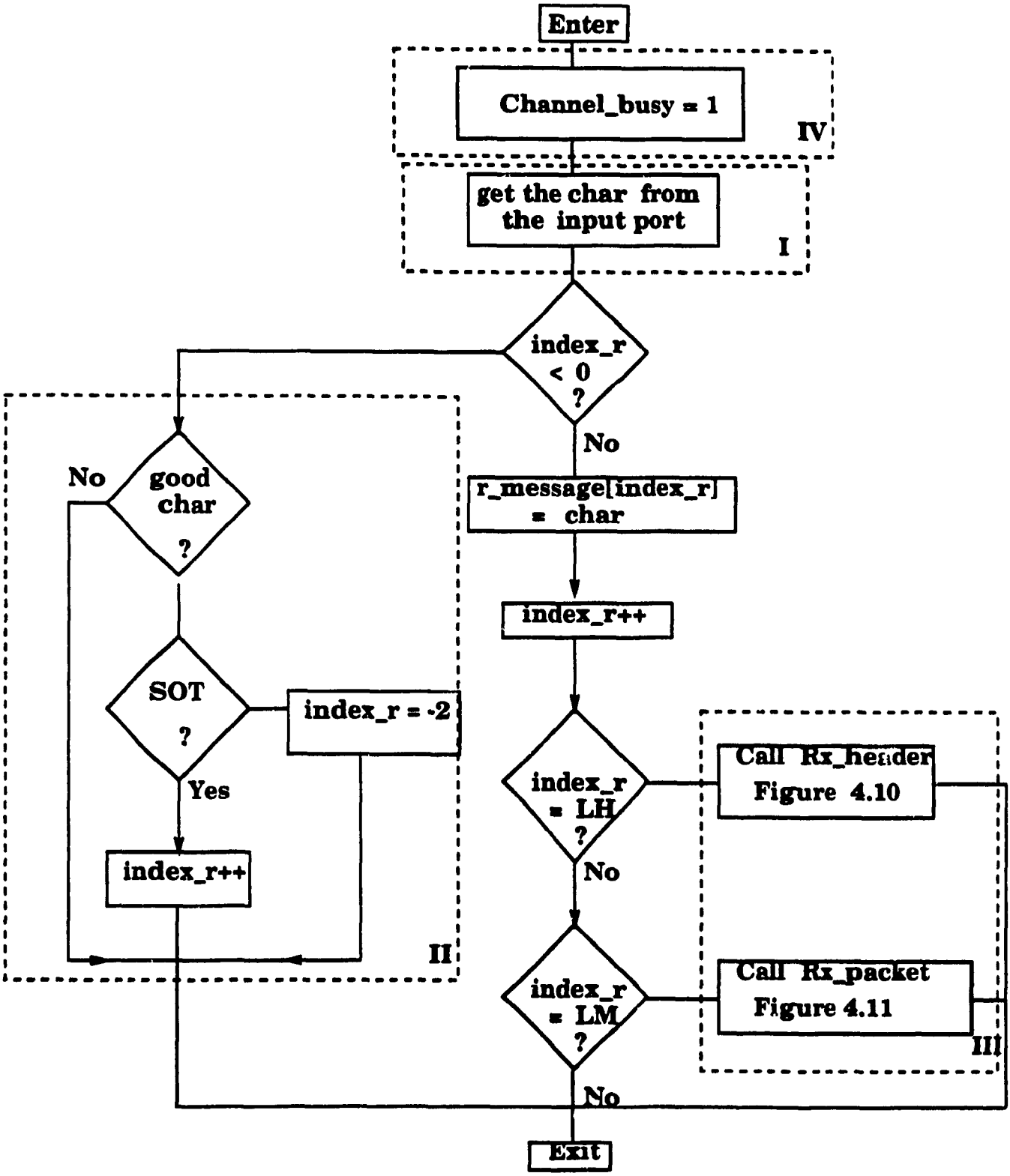


Figure 4.9 Rx_char Module flow_chart

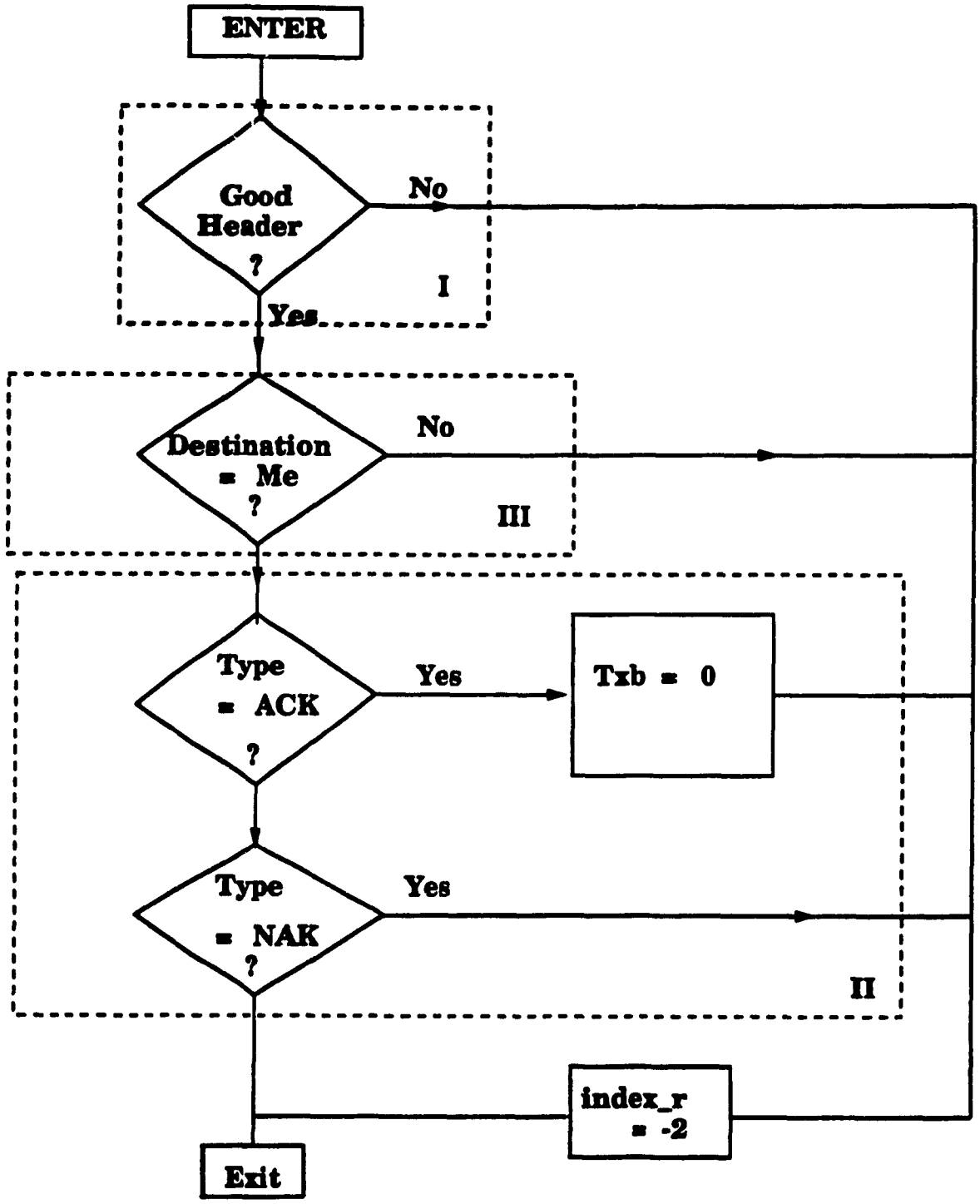


Figure 4.10 Procedure Rx_Header

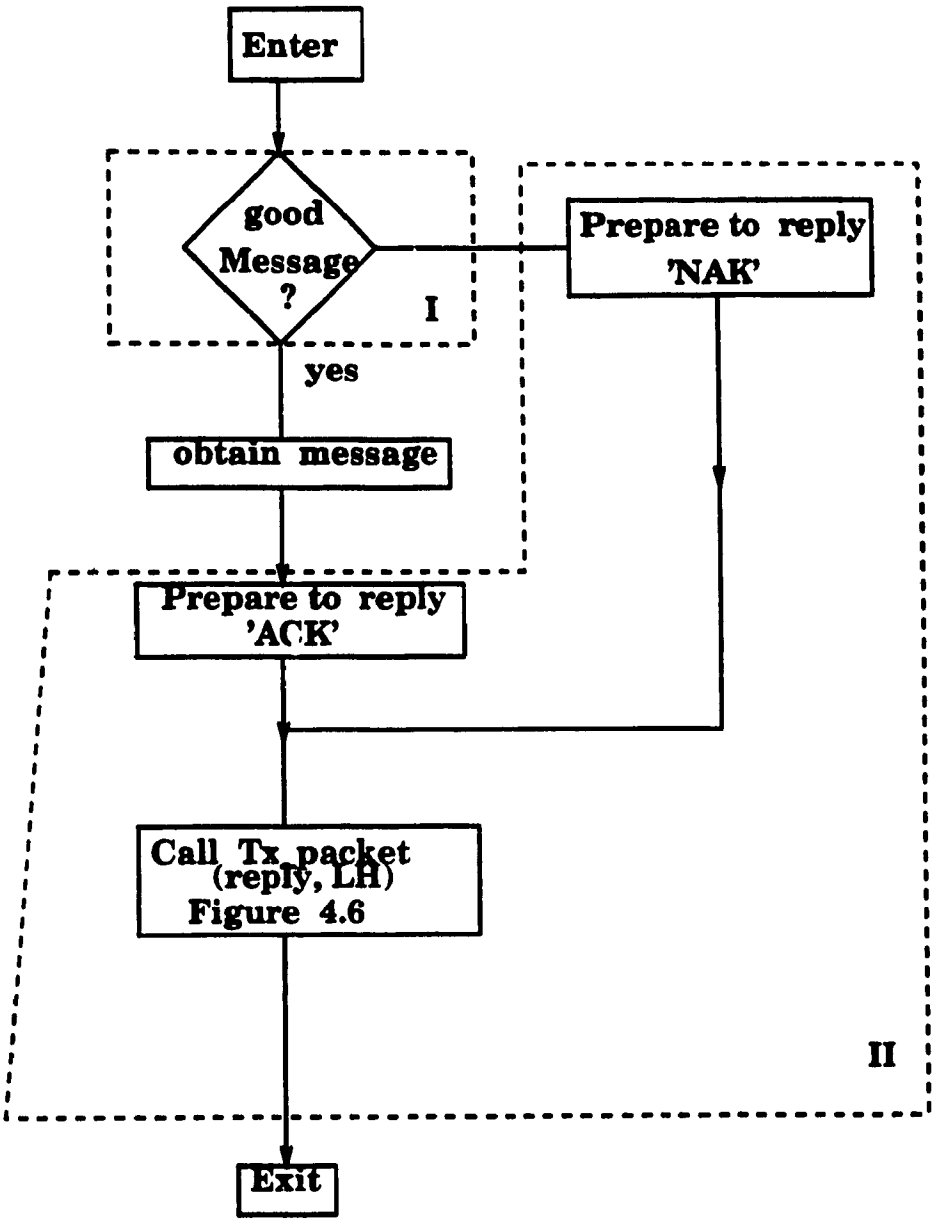


Figure 4.11 Procedure Rx_packet

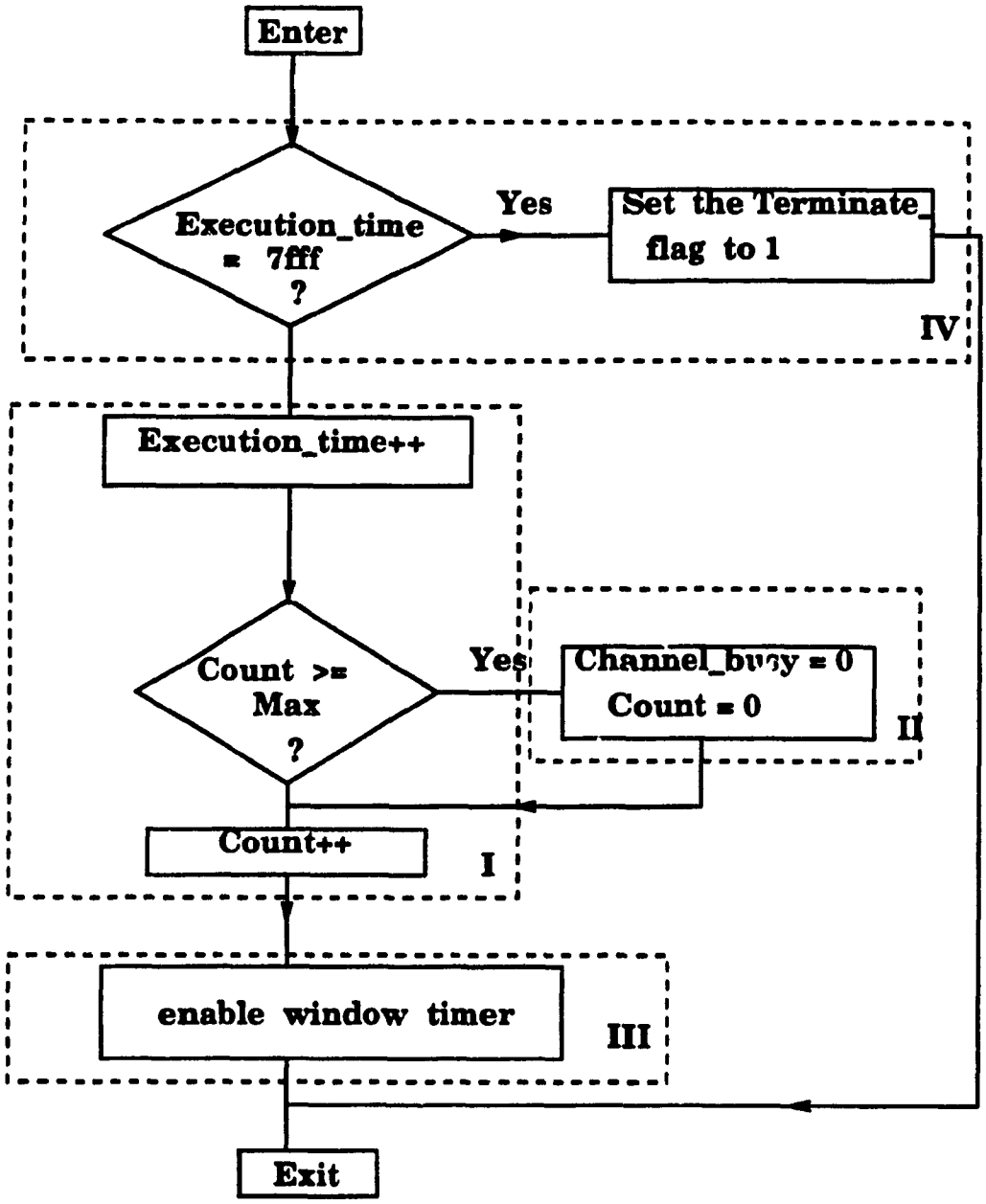


Figure 4.12 Timer Module

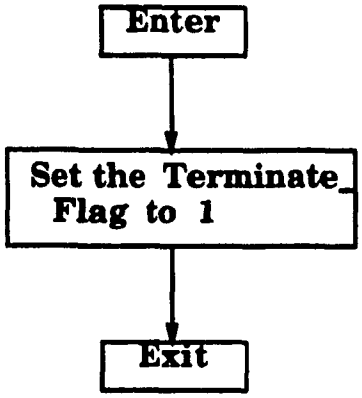


Figure 4.13 Control-break Interrupt Module

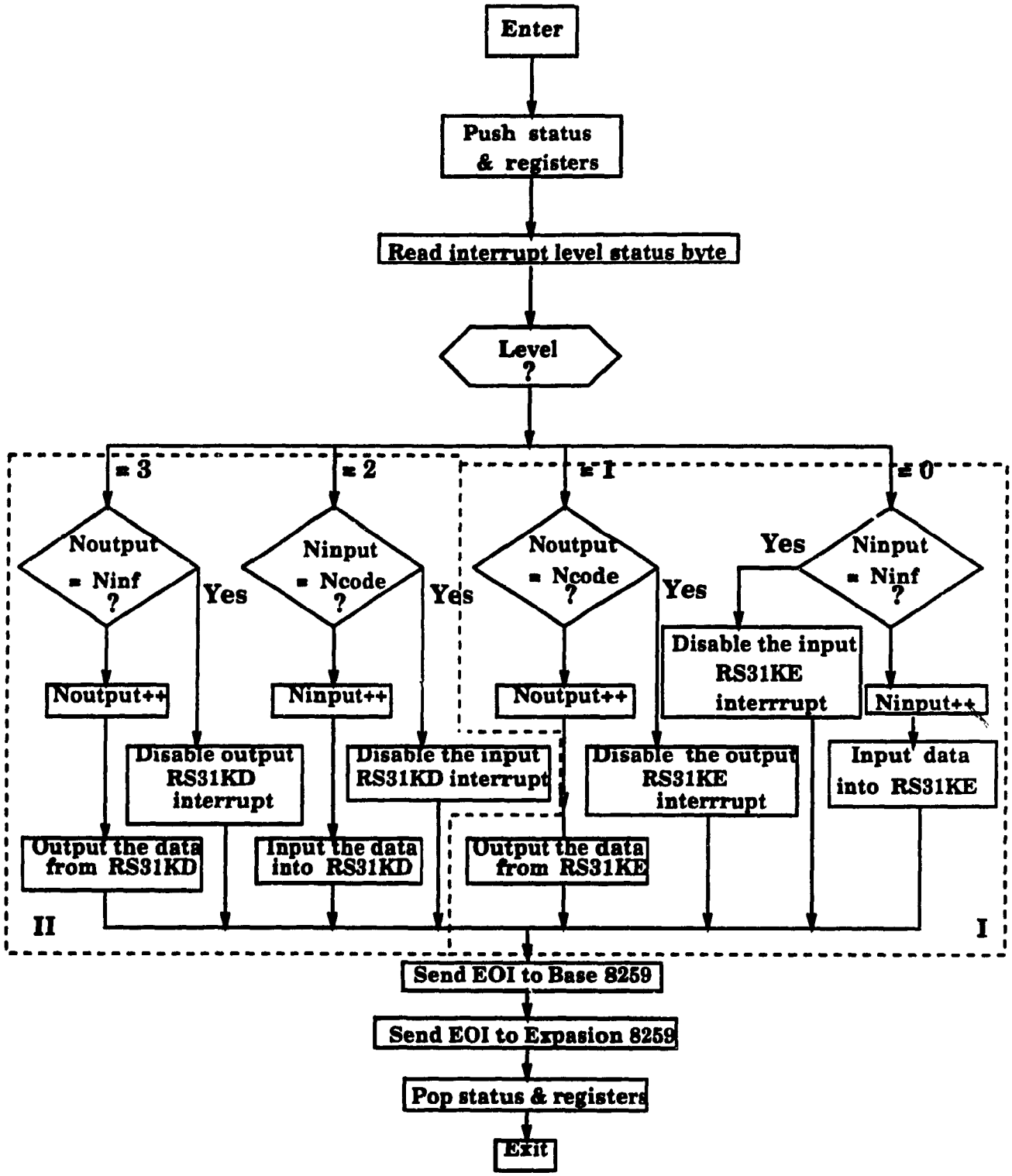


Figure 4.14 RS Codec Module

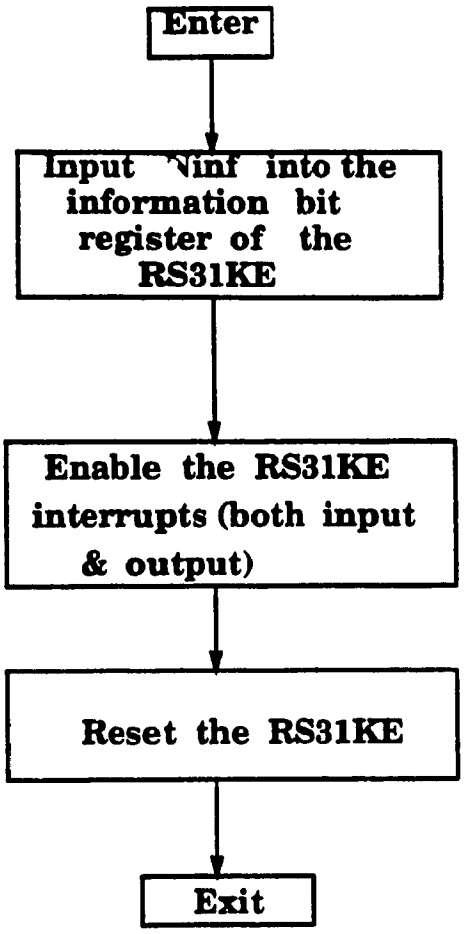


Figure 4.15 Subroutine Encoder

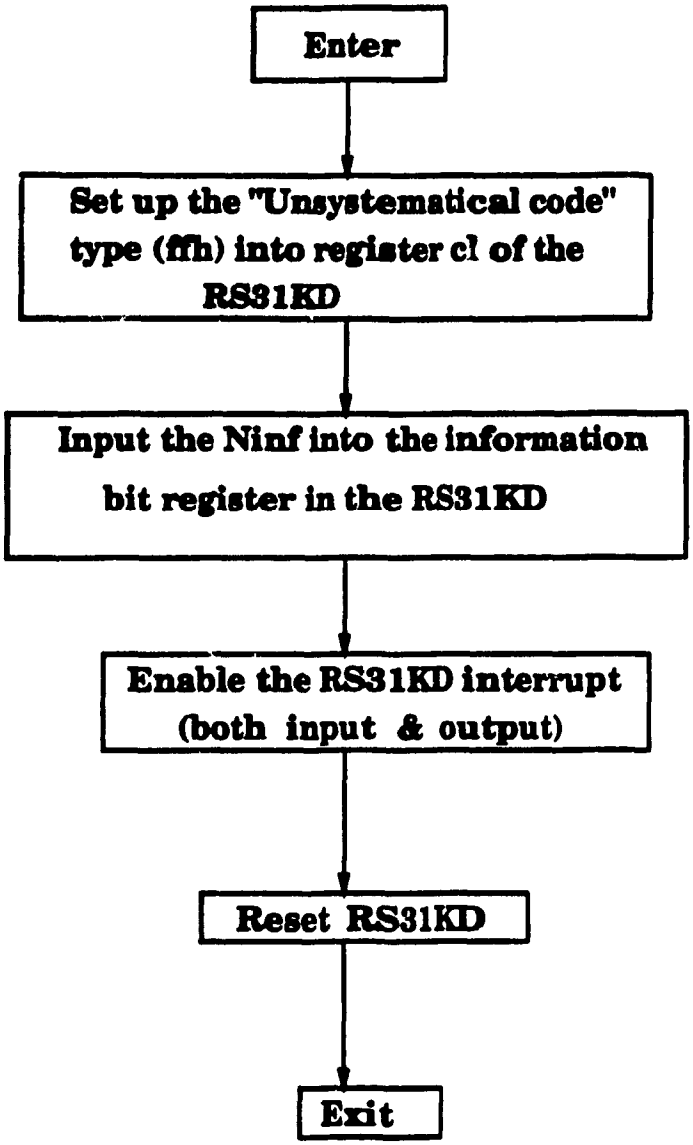


Figure 4.16 Subroutine Decoder

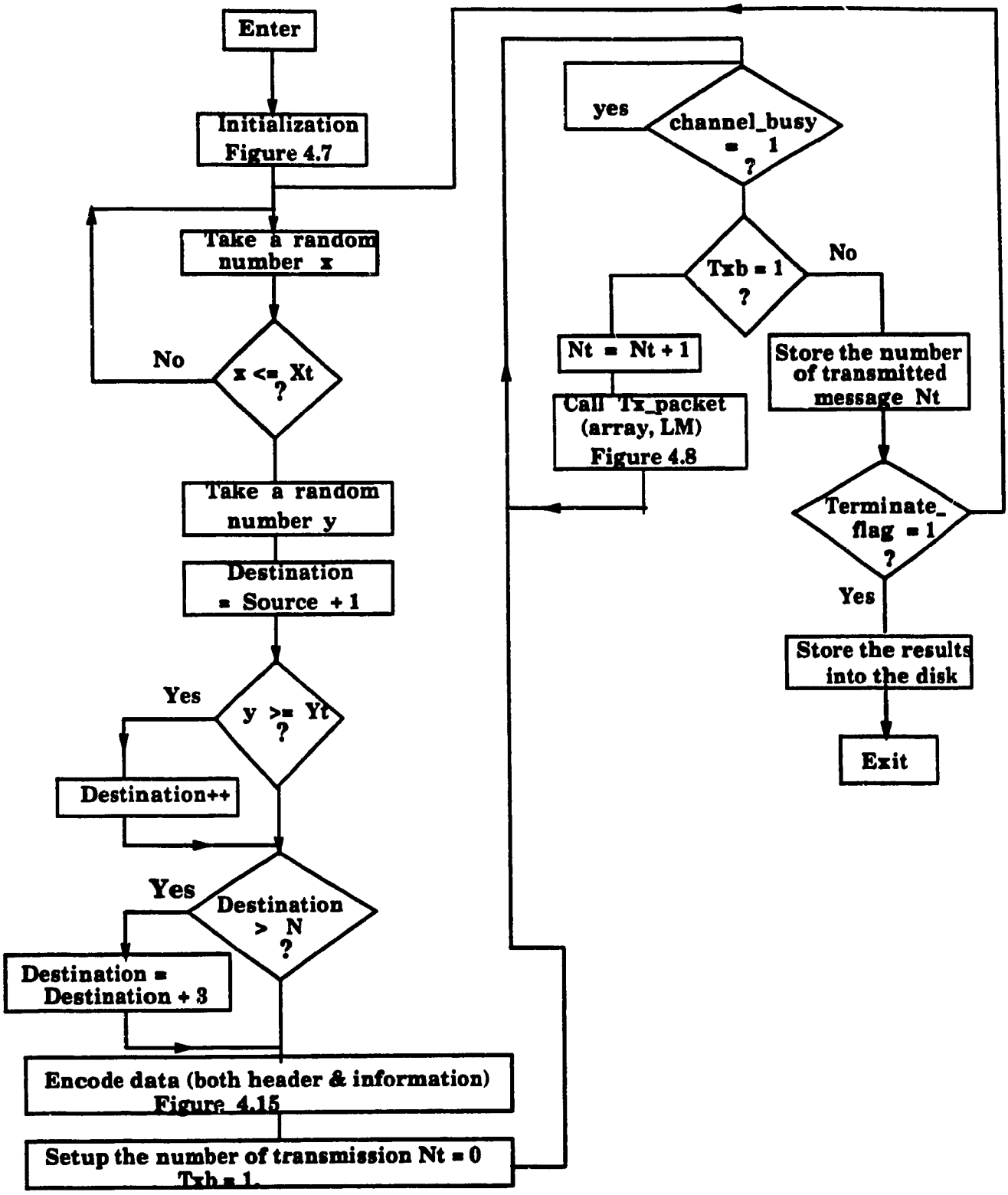


Figure 4.17 Monitor Module Flow_chart for RS Codec

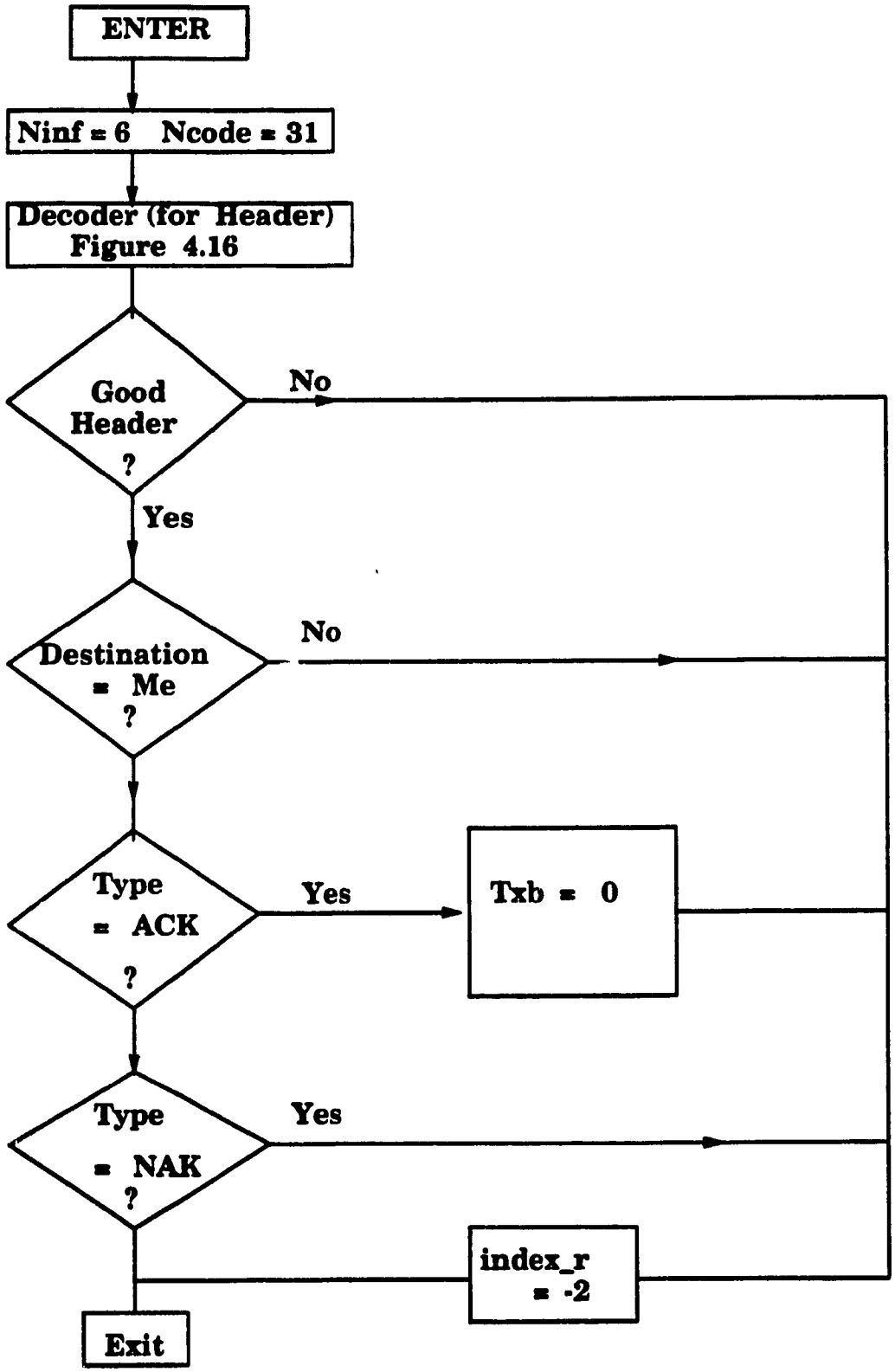


Figure 4.18 Procedure Coded Rx_Header for RS Codec

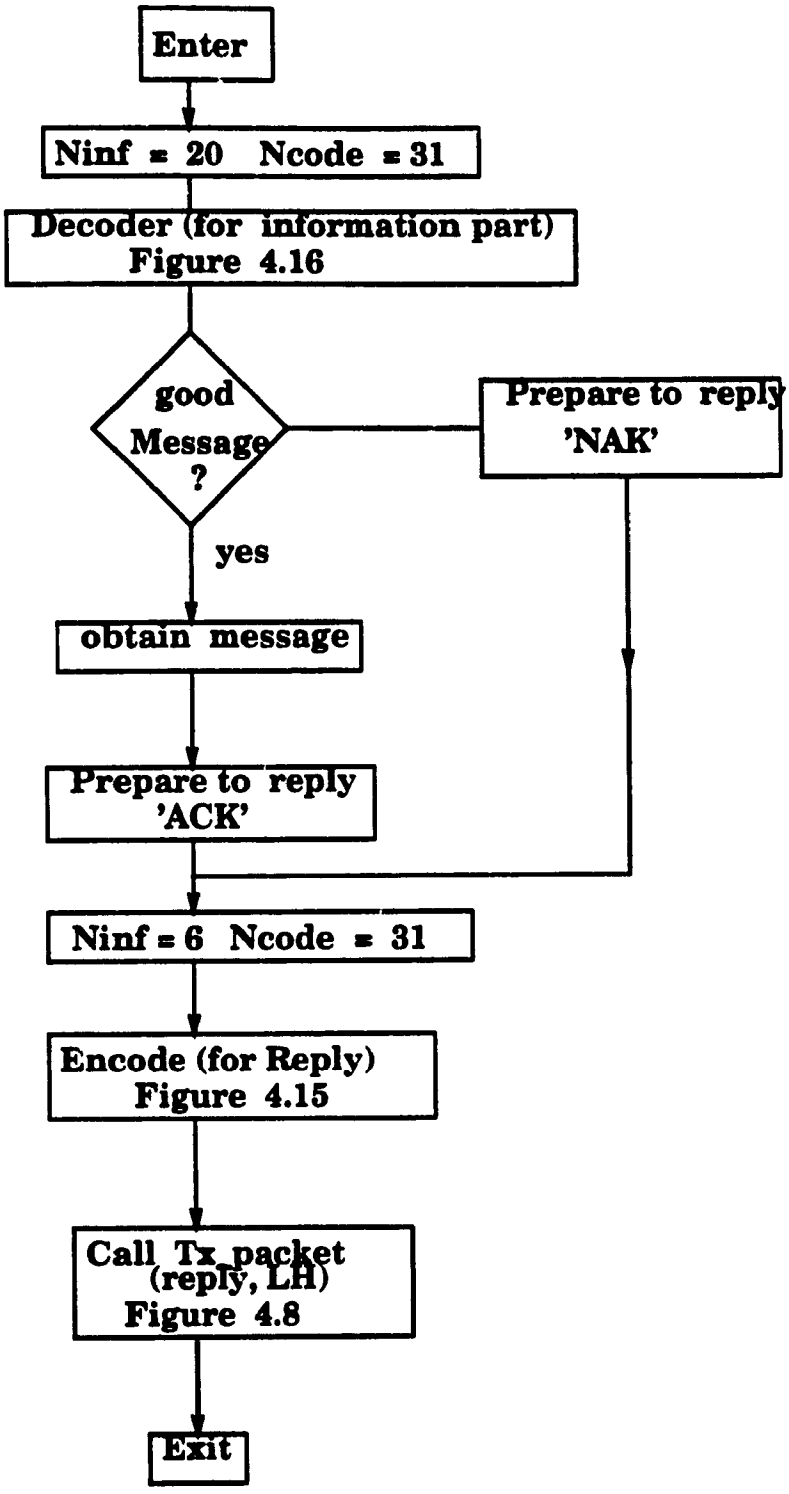


Figure 4.19 Procedure Rx_packet for RS Codec

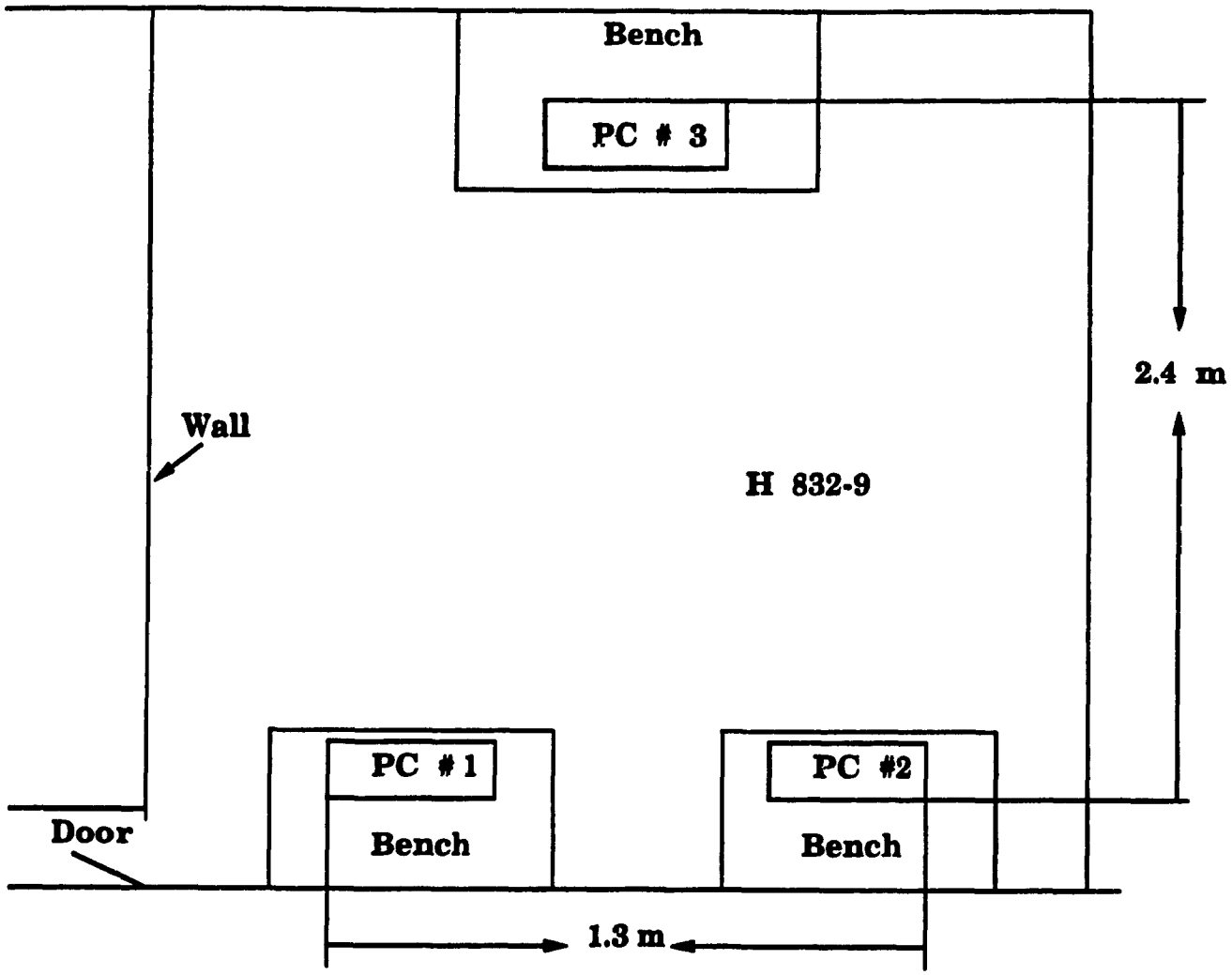


Figure 4.20 : Expriment Configuration # 1 : Transmission within one room.

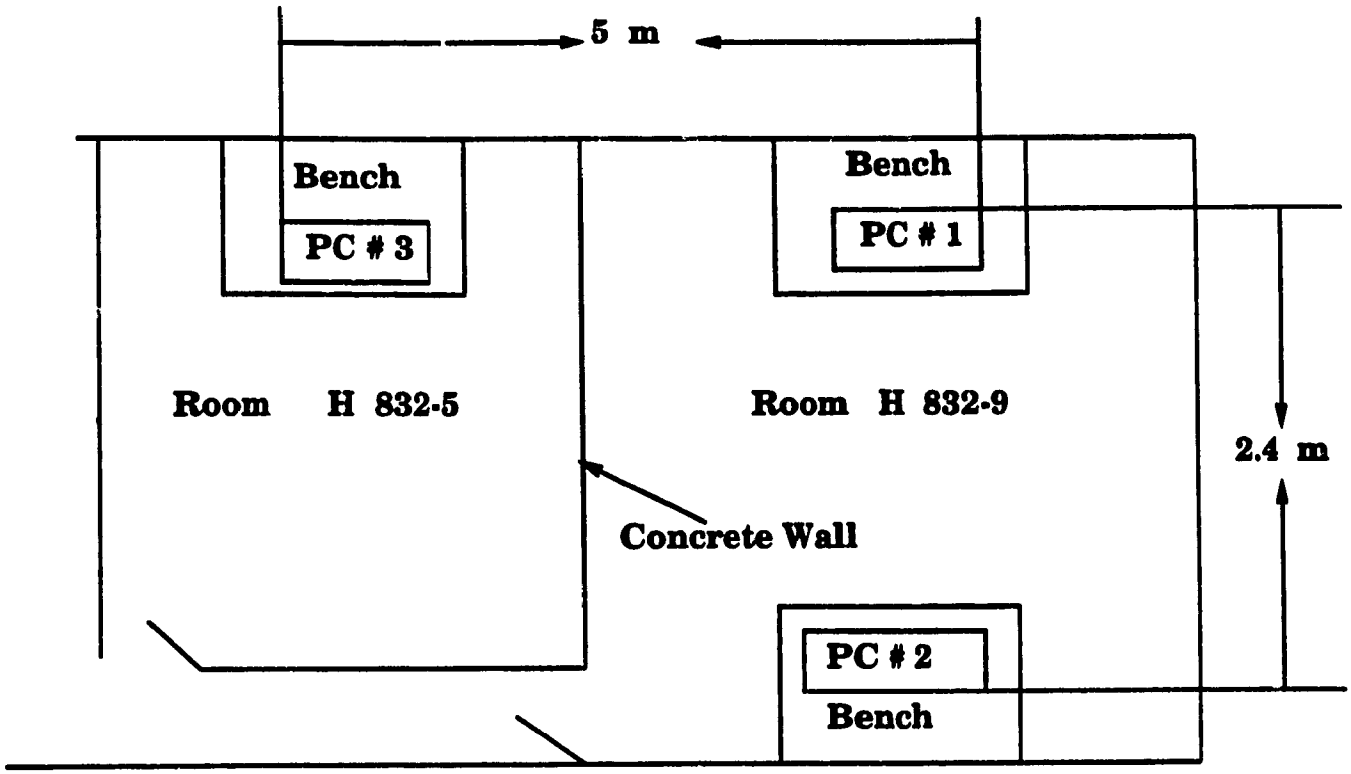


Figure 4.21 Experiment Configuration # 2 :
Transmission between two rooms seperated by walls.

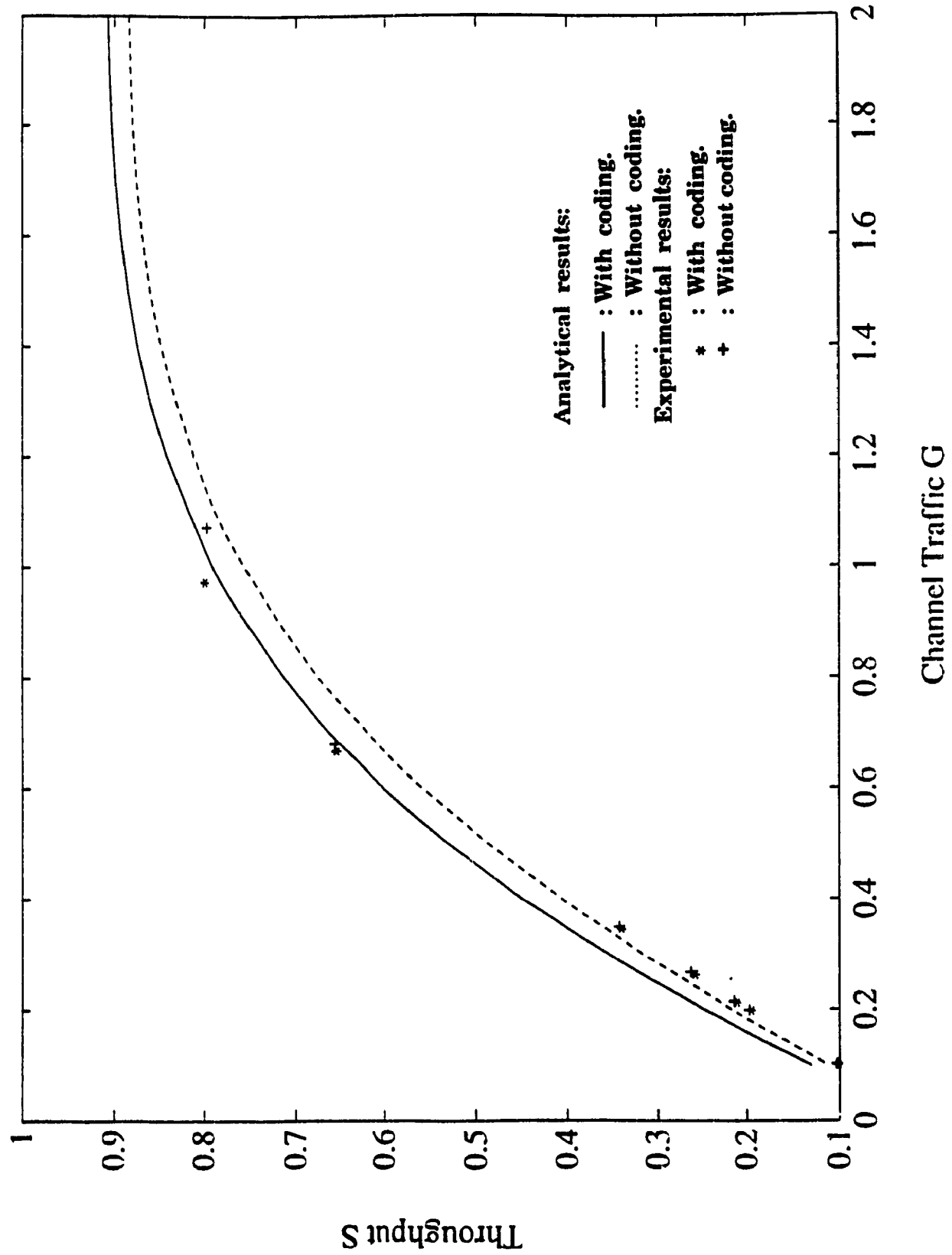


Figure 4.22 : Experimental and theoretical throughput Performance for the setup # 1

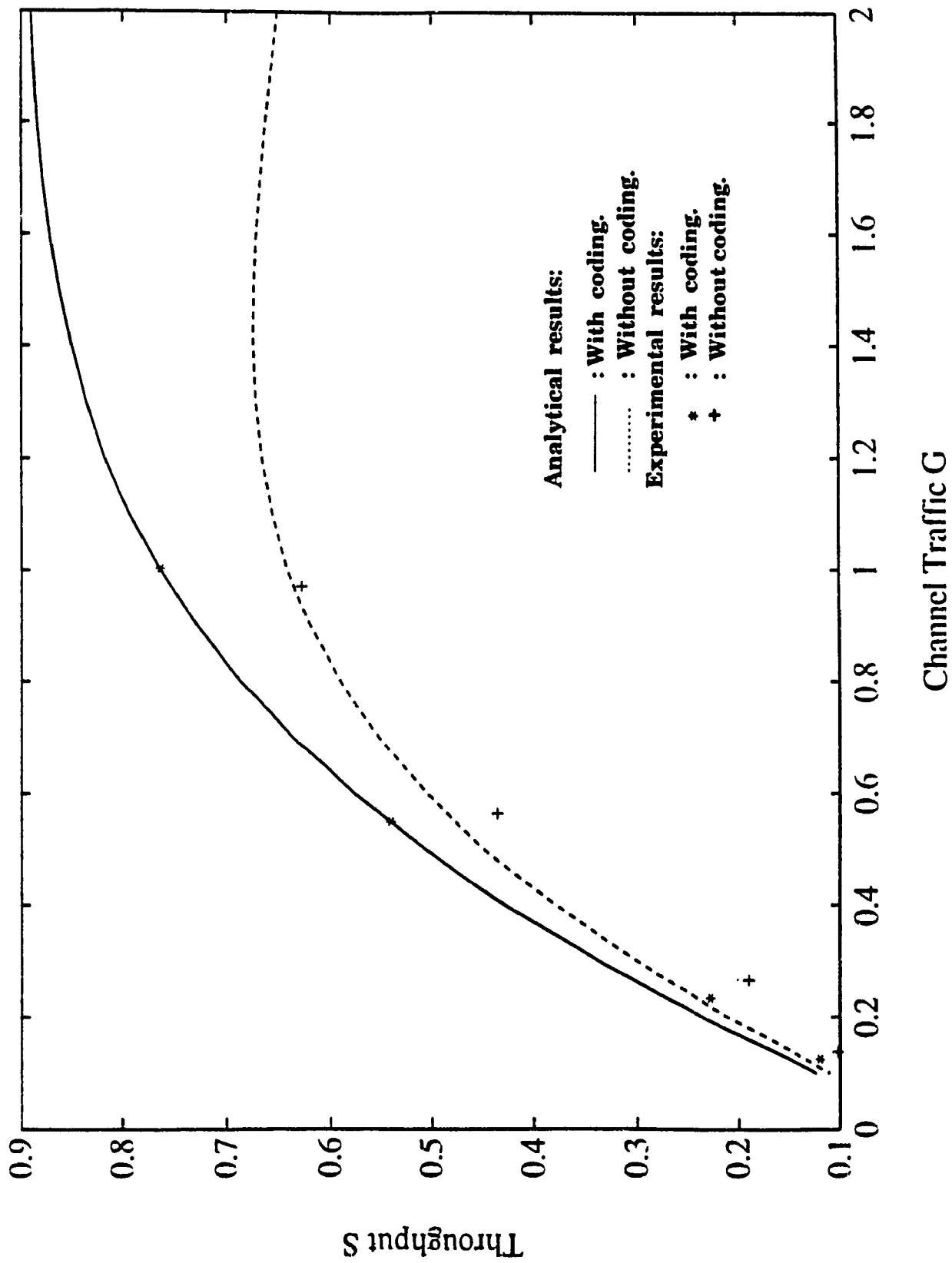


Figure 4.23 : Experimental and theoretical throughput Performance for the setup # 2

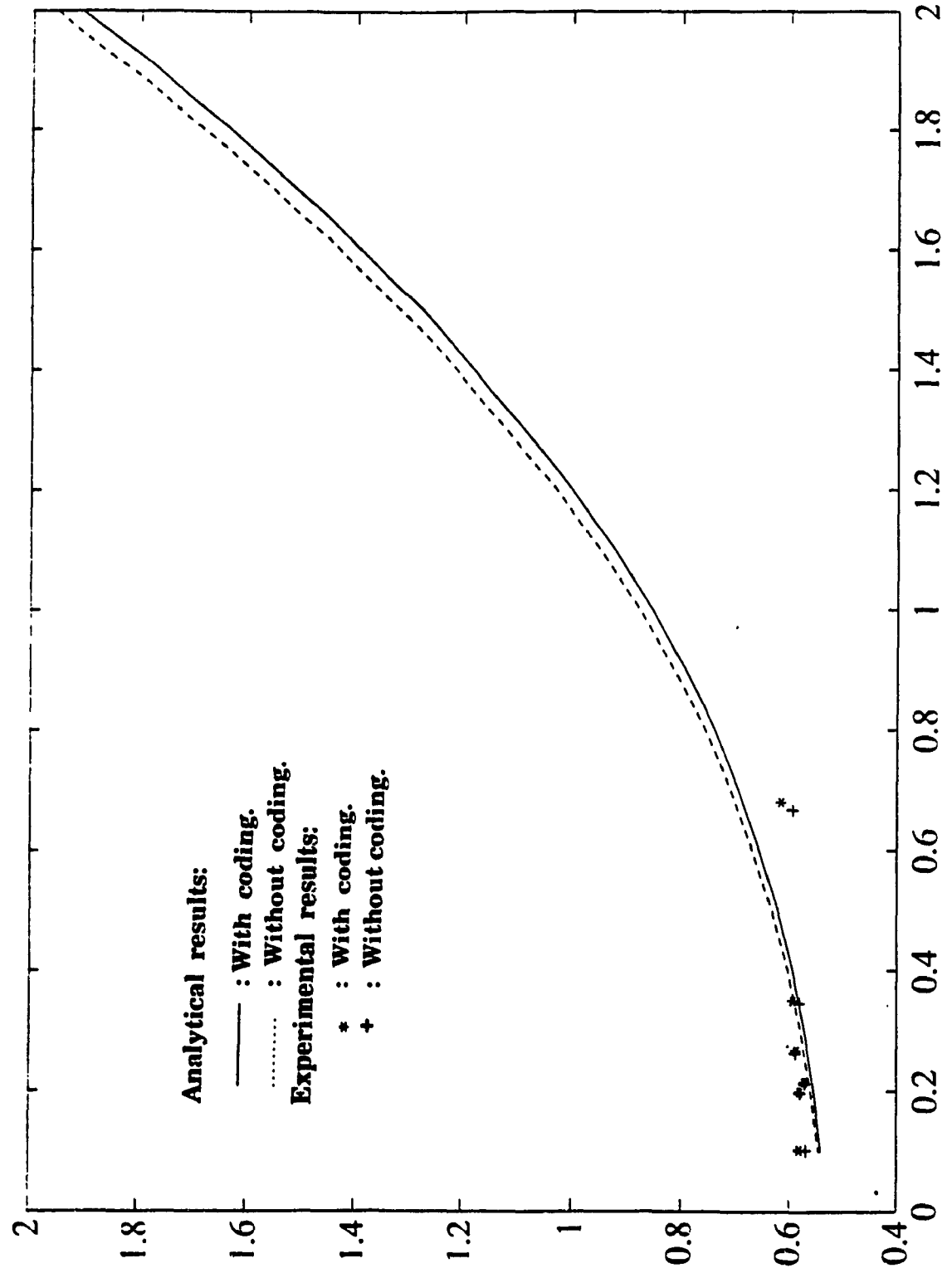


Figure 4.24 : Experimental and theoretical delay Performance for the setup # 1

Figure 4.25 : Experimental and theoretical delay Performance for the setup # 2

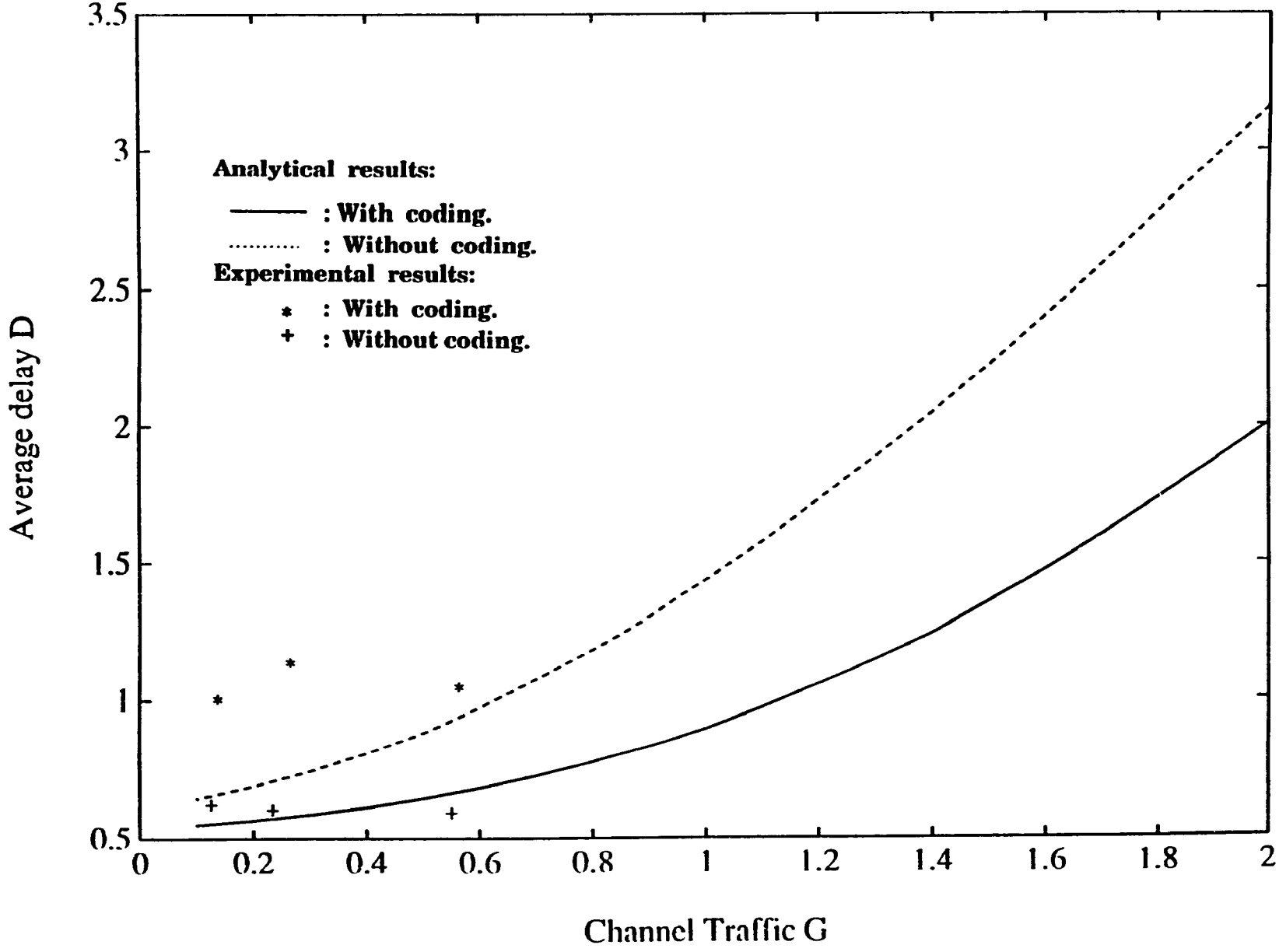


Table 4.1 : The structure of the Sample table for Experimental Channel Parameters.

Description Part

Scheme : Coded CSMA/CD with ACK.
Location : Room H 832-9
Configuration : All three PC's are running at the same time.

****

Arrival rate = 9.107 Sec
Machine # 1

No. of Succ. M	No. of Messages	Throughput	Delay	Traffic	Avg. S	Avg. D	Avg. G
1199	1295	0.768450	0.6131	0.829987	0.336	0.7955	0.423
391	571	0.25059	0.7587	0.365959			
507	573	0.324941	0.6379	0.367241			
0	201	0.0	1.1722	0.128823			

Machine # 2

⋮ ⋮ ⋮ ⋮ ⋮

Machine # 3

⋮ ⋮ ⋮ ⋮ ⋮

Content Part

Total throughput = 0.8005
Packet delay = 0.7580
Total traffic = 0.973

Result Part

****

Rate of change Tr : 4.0077

⋮

Table 4.2 : Table for the Configuration Test Set_up # 1 (without coding).

Scheme : Uncoded CSMA/CD with ACK.
Location : Room 832-29
Configuration : All 3 machines are running at the same time.
****
Arrival rate = 9.1074 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
316	435	0.202527	1.0097	0.278796	0.398	6.984	0.493
893	977	0.572332	0.6767	0.626169			
1204	1228	0.771655	0.5913	0.787039			
974	1075	0.624246	0.6665	0.688978			
858	1244	0.549900	1.0870	0.797291			
1	36	0.000641	40.919	0.023073			
98	385	0.062809	3.9446	0.246750			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
315	481	0.211500	1.1758	0.322957	0.202	2.095	0.282
159	185	0.106757	0.7568	0.124214			
153	159	0.102729	0.6135	0.106757			
113	209	0.075871	1.5477	0.140329			
147	460	0.098700	3.0231	0.308857			
1172	1172	0.786915	0.5683	0.786915			
41	269	0.027529	6.9795	0.180614			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
317	493	0.203168	1.2084	0.315968	0.197	1.29	0.297
323	411	0.207014	0.8824	0.263414			
57	93	0.036532	1.2964	0.059605			
269	324	0.172405	0.8250	0.207655			
137	434	0.087805	3.1490	0.278155			
132	132	0.084600	0.5683	0.084606			
915	1352	0.586432	1.1189	0.866510			

Total throughput = 0.7965
Average packet delay = 3.457
Total Traffic = 1.071

$$\text{Throughput} = (\text{No. of Succ. M})\,(T / T_M) \qquad \text{traffic} = (\text{No. of Messages})\,(T / T_M)$$

$$\text{Delay} = T_i + (\text{No. of Messages} - \text{No. of Succ. M})\,(T_i + T_d) / (\text{No. of Succ. M})$$

$T = 1.2078$ s $T_M = 1798.85$ s $T_i = 0.56833$ s $T_d = 0.6039$ s

****

Arrival rate = 0.2495 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
335	370	0.214705	0.6887	0.237137	0.215	0.656	0.25[illegible]
334	373	0.214064	0.7029	0.239059			
337	349	0.215986	0.6093	0.223677			
335	351	0.214705	0.6233	0.224959			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
335	335	0.224929	0.5683	0.224929	0.225	0.568	0.225
334	334	0.224257	0.5683	0.224257			
336	336	0.225600	0.5683	0.225600			
334	334	0.224257	0.5683	0.224257			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
335	361	0.214705	0.6570	0.231368	0.215	0.607	0.225
335	335	0.214705	0.5683	0.214705			
337	356	0.215986	0.6333	0.228164			
335	355	0.214705	0.5683	0.227523			

Total throughput = 0.65473
Packet delay = 0.610
Total Traffic = 0.68145
****

Arrival rate = 0.11106 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
176	181	0.112800	0.6010	0.116005	0.113	0.606	0.116
176	181	0.112800	0.6010	0.116005			
175	182	0.112159	0.6144	0.116646			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
175	175	0.117500	0.5683	0.117500	0.118	0.570	0.118
175	175	0.117500	0.5683	0.117500			
175	176	0.117500	0.5749	0.118172			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
176	179	0.112800	0.5879	0.114723	0.112	0.608	0.116
175	180	0.112159	0.6012	0.115364			
175	185	0.112159	0.6342	0.118568			

Total throughput = 0.34246
Packet delay = 0.5946
Total Traffic = 0.3502
****

Arrival rate = 0.08317 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
137	137	0.087805	0.5683	0.087805	0.088	0.599	0.089
137	137	0.087805	0.5683	0.087805			
136	147	0.087164	0.6615	0.094214			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
137	137	0.087805	0.5683	0.087805	0.088	0.596	0.090
137	137	0.087805	0.5683	0.087805			
137	147	0.087805	0.6524	0.094214			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
137	137	0.087805	0.5683	0.087805	0.088	0.568	0.088
137	137	0.087805	0.5683	0.087805			
137	137	0.087805	0.5683	0.087805			

Total throughput = 0.26311
Packet delay = 0.5879
Total Traffic = 0.2676
****

Arrival rate = 0.06647 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
112	112	0.071782	0.5683	0.071782	0.072	0.568	0.072
111	111	0.071141	0.5683	0.071141			
112	112	0.071782	0.5683	0.071782			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
111	112	0.071141	0.5788	0.071782	0.071	0.579	0.072
111	111	0.071141	0.5683	0.071141			
111	113	0.071141	0.5891	0.072423			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
111	111	0.071141	0.5683	0.071141	0.071	0.575	0.072
111	113	0.071141	0.5891	0.072423			
112	112	0.071782	0.5683	0.071782			

Total throughput = 0.21406
Packet delay = 0.5740
Total Traffic = 0.2151
****

Arrival rate = 0.06071 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
103	103	0.066014	0.5683	0.066014	0.066	0.587	0.067
102	107	0.065373	0.6248	0.068577			
102	102	0.065373	0.5683	0.065373			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
103	103	0.066014	0.5683	0.066014	0.066	0.58	0.066
102	106	0.065373	0.6135	0.067936			
102	102	0.065373	0.5683	0.065373			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
102	102	0.065373	0.5683	0.065373	0.066	0.576	0.066
103	103	0.066014	0.5683	0.066014			
102	104	0.065373	0.5909	0.066655			

Total throughput = 0.1964
Packet delay = 0.582
Total Traffic = 0.1989

****

Arrival rate = 0.03045 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
53	55	0.033968	0.6118	0.035250	0.034	0.583	0.034
53	53	0.033968	0.5683	0.033968			
53	53	0.033968	0.5683	0.033968			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
53	53	0.033968	0.5683	0.033968	0.034	0.597	0.035
53	54	0.033968	0.5900	0.034609			
53	56	0.033968	0.6335	0.035891			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
53	53	0.033968	0.5683	0.033968	0.034	0.568	0.034
53	53	0.033968	0.5683	0.033968			
53	53	0.033968	0.5683	0.033968			

Total throughput = 0.1019
Packet delay = 0.5827
Total Traffic = 0.103068
*

Table 4.2 : Table for the Configuration Test Set_up # 2 (without coding).

Scheme : Uncoded CSMA/CD with ACK.
Location : Rooms H832-9 and H832-5.
Configuration : All 3 machines are running at the same time.
****
Arrival rate = 9.107 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
661	698.00	0.443815	0.6328	0.468657	0.178	2.182	0.2698
218	316.00	0.146372	1.0866	0.212172			
123	337.00	0.082586	2.5718	0.226272			
59	257.00	0.039614	4.4373	0.172557			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
550	618.00	0.352500	0.7108	0.396082	0.265	6.173	0.340
965	1103.0	0.618478	0.7332	0.706923			
11	217.00	0.007050	22.159	0.139077			
128	186.00	0.082036	1.0907	0.119209			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
33	129.00	0.021150	3.9222	0.082677	0.184	3.709	0.361
45	154.00	0.028841	3.3609	0.098700			
942	1164.0	0.603737	0.8400	0.746019			
127	804.00	0.081396	6.7140	0.515291			

Total throughput = 0.627
Packet delay = 4.0216Total Traffic = 0.9708

****
Arrival rate = 0.1656 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
169	405.00	0.113472	2.1780	0.271929	0.149	1.111	0.202
247	251.00	0.165843	0.5867	0.168529			
248	248.00	0.166514	0.5682	0.166514			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
174	341.00	0.111518	1.6748	0.218550	0.143	0.937	0.178
246	246.00	0.157664	0.5683	0.157664			
248	248.00	0.158946	0.5683	0.158946			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
173	366.00	0.110877	1.8545	0.234573	0.143	0.997	0.184
247	247.00	0.158305	0.5683	0.158305			
247	247.00	0.158305	0.5683	0.158305			

Total throughput = 0.434
Packet delay = 1.0129
Total Traffic = 0.5644

****

Arrival rate = 0.0607 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
102	102.00	0.068486	0.5683	0.068486	0.064	1.212	0.093
102	102.00	0.068486	0.5683	0.068486			
78	269.00	0.049991	3.3914	0.172405			
101	111.00	0.067814	0.6824	0.074529			
99	123.00	0.066471	0.8478	0.082586			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
102	102.00	0.065373	0.5683	0.065373	0.062	1.350	0.095
102	102.00	0.065373	0.5683	0.065373			
73	314.00	0.049014	4.3744	0.210829			
101	110.00	0.064732	0.6710	0.070500			
100	100.00	0.064091	0.5683	0.064091			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
103	103.00	0.066014	0.5683	0.066014	0.064	0.864	0.079
103	103.00	0.066014	0.5683	0.066014			
90	176.00	0.057682	1.6699	0.112800			
103	103.00	0.066014	0.5683	0.066014			
98	130.00	0.062809	0.9447	0.083318			

Total throughput = 0.19
Packet delay = 1.1418
Total Traffic = 0.267

****

Arrival rate = 0.03046 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
50	104.00	0.033571	1.8134	0.069829	0.035	0.983	0.047
53	53.000	0.035586	0.5683	0.035586			
53	53.000	0.035586	0.5683	0.035586			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
49	89.00	0.031405	1.5094	0.057041	0.033	0.882	0.042
53	53.00	0.033968	0.5683	0.033968			
53	53.00	0.033968	0.5683	0.033968			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
49	124.00	0.031405	2.3329	0.079473	0.033	1.156	0.049
53	53.000	0.033968	0.5683	0.033968			
53	54.000	0.033968	0.5897	0.034609			

Total throughput = 0.10143
Packet delay = 1.0072
Total Traffic = 0.138
*

Table 4.4 : Table for the Configuration Test Set_up # 1 (with coding).

Scheme : Coded CSMA/CD with ACK.
Location : Room 832-9.
Configuration : All 3 machines are running at the same time.
****
Arrival rate = 9.1074 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
1199	1295.000	0.768450	0.6606	0.829978	0.337	7.7145	0.423
391	571.0000	0.250596	1.0990	0.365959			
507	573.0000	0.324941	0.7184	0.367241			
8	201.0000	0.005210	28.380	0.128823			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
81	81.000	0.051914	0.5683	0.051914	0.0385	1.9145	0.0802
49	92.000	0.031405	1.5800	0.058964			
49	92.000	0.031405	1.5800	0.058964			
60	235.00	0.038455	3.9300	0.150614			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
33	50.000	0.021150	1.1622	0.032045	0.426	0.7939	0.4696
777	825 00	0.497987	0.6395	0.528750			
729	782.00	0.467223	0.6521	0.501191			
1124	1274.0	0.720382	0.7221	0.816519			

Total throughput = 0.8006
Packet delay = 3.233
Total Traffic = 0.973
****

Arrival rate = 0.2495 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
335	338	0.214705	0.5786	0.216627	0.2149	0.5715	0.2155
336	336	0.215346	0.5683	0.215346			
335	335	0.214705	0.5683	0.214705			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
335	335	0.224929	0.5683	0.224929	0.2249	0.5683	0.2249
335	335	0.224929	0.5683	0.224929			
335	335	0.224929	0.5683	0.224929			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
336	354	0.215346	0.6300	0.226882	0.215	0.6400	0.229
336	357	0.215346	0.6400	0.228805			
335	359	0.214705	0.6500	0.230086			

Total throughput = 0.655
Packet delay = 0.5932
Total Traffic = 0.669

****

Arrival rate = 0.11106 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
174	175	0.116829	0.5740	0.117500	0.117	0.5740	0.1179
175	176	0.117500	0.5740	0.118172			
175	176	0.117500	0.5740	0.118172			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
175	175	0.112159	0.5683	0.112159	0.1120	0.5727	0.1124
174	176	0.111518	0.5815	0.112800			
175	175	0.112159	0.5683	0.112159			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
175	181	0.112159	0.6078	0.116005	0.1122	0.6034	0.1156
175	179	0.112159	0.5946	0.114723			
175	181	0.112159	0.6078	0.116005			

Total throughput = 0.341379
Packet delay = 0.5833
Total Traffic = 0.34587

****

Arrival rate = 0.08317 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
135	135	0.086523	0.5683	0.086523	.0865	0.5683	.0865
135	135	0.086523	0.5683	0.086523			
135	135	0.086523	0.5683	0.086523			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
135	135	0.086523	0.5683	0.086523	.0865	0.5796	.0865
135	135	0.086523	0.5683	0.086523			
135	139	0.086523	0.6024	0.089086			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
136	136	0.087164	0.5683	0.087164	.08674	0.6138	0.0901
135	143	0.086523	0.6366	0.091650			
135	143	0.086523	0.6366	0.091650			

Total throughput = 0.25978
Packet delay = 0.5872
Total Traffic = 0.263

****

Arrival rate = 0.06647 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
111	112	0.071141	0.5787	0.071782	0.0707	0.5717	0.0709
110	110	0.070500	0.5683	0.070500			
110	110	0.070500	0.5683	0.070500			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
110	110	0.070500	0.5683	0.070500	0.0705	0.5719	0.0709
110	111	0.070500	0.5737	0.071141			
110	111	0.070500	0.5737	0.071141			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
111	111	0.071141	0.5683	0.071141	0.0709	0.5683	0.0709
110	110	0.070500	0.5683	0.070500			
111	111	0.071141	0.5683	0.071141			

Total throughput = 0.21213
Packet delay = 0.5773
Total Traffic = 0.2127

****

Arrival rate = 0.06071 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
102	103	0.065373	0.5796	0.066014	0.0654	0.5758	0.0658
102	103	0.065373	0.5796	0.066014			
102	102	0.065373	0.5683	0.065373			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
101	101	0.064732	0.5683	0.064732	0.065	0.5720	0.065
101	101	0.064732	0.5683	0.064732			
102	103	0.065373	0.5796	0.066014			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
101	101	0.064732	0.5683	0.064732	0.065	0.5987	0.0666
102	102	0.065373	0.5683	0.065373			
101	109	0.064732	0.6596	0.069859			

Total throughput = 0.195263
Packet delay = 0.5821
Total Traffic = 0.1976

****

Arrival rate = 0.03045 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
53	53	0.033968	0.5683	0.033968	0.034	0.568	0.034
53	53	0.033968	0.5683	0.033968			
53	53	0.033968	0.5683	0.033968			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
53	53	0.033968	0.5683	0.033968	0.034	0.568	0.034
53	53	0.033968	0.5683	0.033968			
53	53	0.033968	0.5683	0.033968			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
53	53	0.033968	0.5683	0.033968	0.034	0.568	0.034
53	53	0.033968	0.5683	0.033968			
53	53	0.033968	0.5683	0.033968			

Total throughput = 0.1019
Packet delay = 0.5683
Total Traffic = 0.1019

Table 4.5 : Table for the Configuration Test Set_up # 2 (with coding).

Scheme : Coded CSMA/CD with ACK.
Location : rooms H832-9 and H832-5
Configuration : All 3 machines are running at the same time.
****
Arrival rate = 9.1074 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
469	639.00	0.386468	0.9862	0.526553	0.378	0.956	0.460
232	459.00	0.191174	1.6964	0.378228			
930	934.00	0.766345	0.5732	0.769641			
203	203.00	0.167277	0.5683	0.167277			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
417	645.00	0.311791	1.1986	0.482266	0.226	0.917	0.316
417	645.00	0.311791	1.1986	0.482266			
188	199.00	0.140568	0.6357	0.148792			
188	199.00	0.140568	0.6357	0.148792			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
16	194.00	0.010420	13.390	0.136178	0.164	9.157	0.229
10	190.00	0.006510	21.320	0.133370			
3	5.00	0.002106	1.3369	0.003510			
905	915.00	0.635263	0.5810	0.642283			

Total throughput = 0.7680
Packet delay = 3.6767
Total Traffic = 1.004
****

Arrival rate = 0.1656 Sec
Machine # 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
239	241	0.196942	0.5779	0.198590	0.196	0.575	0.198
239	241	0.196942	0.5779	0.198590			
237	237	0.195294	0.5683	0.195294			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
237	237	0.177205	0.5683	0.177205	0.178	0.582	0.180
238	243	0.177953	0.5925	0.181691			
239	243	0.178700	0.5876	0.181691			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
238	247	0.167064	0.6119	0.173381	0.167	0.609	0.173
239	245	0.167766	0.5972	0.171977			
238	248	0.167064	0.6167	0.174083			

Total throughput = 0.541638
Packet delay = 0.5900
Total Traffic = 0.550833
****

Arrival rate = 0.0607 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
101	103	0.083227	0.5911	0.084875	0.083	0.579	0.084
100	101	0.082403	0.5798	0.083227			
101	101	0.083227	0.5683	0.083227			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
100	106	0.074770	0.6375	0.079256	0.075	0.641	0.080
100	113	0.074770	0.7182	0.084490			
101	101	0.075518	0.5683	0.075518			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
100	100	0.070195	0.5683	0.070195	0.071	0.584	0.071
101	101	0.070897	0.5683	0.070897			
100	104	0.070195	0.6144	0.073003			

Total throughput = 0.2283
Packet delay = 0.6012
Total Traffic = 0.234855

****

Arrival rate = 0.03046 Sec
Machine # : 1

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
53	56	0.043673	0.6335	0.046145	0.043	0.708	0.049
52	68	0.042849	0.9230	0.056034			
53	53	0.043673	0.5683	0.043673			

Machine # : 2

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
53	54	0.039628	0.5900	0.040376	0.039	0.583	0.040
53	54	0.039628	0.5900	0.040376			
53	53	0.039628	0.5683	0.039628			

Machine # : 3

No. of Succ. M	No. of Messages	Throughput	Delay	traffic	Avg. S	Avg. D	Avg. G
53	53	0.037203	0.5683	0.037203	0.037	0.568	0.037
53	53	0.037203	0.5683	0.037203			
53	53	0.037203	0.5683	0.037203			

Total throughput = 0.120221
Packet delay = 0.6190
Total Traffic = 0.1259
*

CHAPTER 5

CONCLUSIONS

The analytical performances of the portable data communication system using CSMA/CD with ACK scheme was proposed and evaluated. Both analytical and experimental results were provided. The experimental results were used to verify the analytical ones and to demonstrate the applicability of the proposed wireless data communication system.

In Chapter 2, the transmission characteristics of an indoor wireless communications environment were examined by experiments for various configurations. It is noticed for indoor wireless communication environments that errors often occur in bursts [6]. This indicates that the transmission performance is mostly affected by relatively unpredictable man-made interference and/or reflections rather than thermal noise. In this varying environment, an adaptive burst-error-correction Reed-Solomon (RS) coding scheme can be an appropriate one to obtain high system throughput by changing the error correction capability according to the level of error occurrence over the channel.

By including the probability of successful transmission due to the channel characteristics in the analysis, the system throughput and average delay were analytically evaluated in Chapter 3. It is indicated that the use of RS codes significantly improves the system performance when errors occur more often over the channel.

The analytical results were then verified by experiments. It is shown that both analytical and experimental results are in a good agreement as shown in Chapter 4. The experiments reported in Chapter 4 were also used to demonstrate the feasibility and applicability of the proposed system. For this, both software and hardware aspects of the

proposed system were described in detail.

In summary, a low-cost portable data communications system using CSMA/CD with ACK has been introduced, analytically evaluated and experimentally demonstrated. The proposed system shows its feasibility and applicability in applications with low-capacity traffic requirements.

Suggestions for further studies include:

. Adaptive RS Coding Schemes.

. Combined Coding/Diversity Techniques.

. Extension to High-Capacity Portable Data Systems.

REFERENCES

[1] Shyamala Rddy, "The Missing Links: Wireless Networks Provide Ease and Portability." *The Local Area Network Magazine, Miller Freeman Publications Inc., San Francisco, Calif.*, Vol. 5, No. 9, September, 1990, pp. 100-107.

[2] John T. Mulqueen, "Mobile Data Network Debuts," *Data Communications, McGraw-Hill Inc., New York, New York*, Vol. 20, No. 1, January, 1991, pp. 78-79.

[3] F.A. Tobagi, V.B. Hunt, "Performance Analysis of Carrier-Sense-Multiple-Access with Collision Detection", Chapter 20 of *Advances in Local-Area-Networks* edited by K. Kummerle, F.A.Tobagi, J.O.Limb, IEEE Press, 1987-, pp. 245-255.

[4] A. Hazeltine, D. Richman, and B.D. Loughlin, "Superregenerative Design," *Electronics*, Vol. 2, Sept. 1948, p. 99.

[5] T. Le-Ngoc et al., "A gate-array based programmable Reed-Solomon codec: structure-implementation-applications", *MILCOM'90*, Monterey, CA, Sept. 30-Oct. 3, 1990, pp 121-130.

[6] W. Murray, P. Dennis, and K. Le, *"Data Transmission using a Low-Cost ASK Transmitter Super-regenerative Receiver,"* Elect 481 Project, May 1990, Concordia University, pp. 22-29.

[7] A. Papoulis, *Probability, Random Variables, and Stochastic Process:* McGraw-Hill, 1984, pp. 30-35.

[8] Y. Takahashi, Y. Matsumoto, and T. Hasegawa, "Probability Distributions of Delay and Interdeparture time in Non-slotted CSMA/CD," *Local Communication Systems: Lan and PBX, IFIP 1987*, pp. 423-435, Toulouse, France.

APPENDIX A

C SOURCE CODE

```
/* Module name: main.c
   Created:  Jan. 1, 1992
   By     :  Victor C. Phan
   Version:  1
*/
/*************************************************************
 *
 *  main.c - c program containing main and other functions
 *
 *************************************************************/


#include <sys\stat.h>
#include <stdlib.h>
#include <time.h>
#include <dos.h>
#include <bios.h>
#include <stdio.h>
#include "extern.h"
#define  TRUE  1
#define  FALSE 0

/* baud rate */

#define B600    0x60    /* 600 bits/second */
#define B1200   0x80    /* 1200 bits/second */
#define B2400   0xA0    /* 2400 bits/second */
#define B4800   0xC0    /* 4800 bits/second */
#define B9600   0xE0    /* 9600 bits/second */

/* Parity */

#define  PNONE  0       /* no parity */
#define  PODD   8       /* odd parity */
#define  PEVEN  0x18    /* even parity */

/* stop bits */

#define  SB1     0        /* 1 stop bit */
#define  SB2     4        /* 2 stop bits */

/* data bits */

#define DB7     2       /* 7 data bits */
#define DB8     3       /* 8 data bits */
```

```
/* control key */

#define  LF      0x0a
#define  CR      0x0d
#define SOH      1

/* constant */
#define forever 1


/* global variable */
FILE   *handle;          /* file handle */
int length, tx_time_rate = 500; /* number of back_off time */
char  outarray[70];     /* The encoder output */
char   message_length, txx[70];
/* input message length and its content */
int  rate_of_change, max1_index;  /*arrival rate, block
                                    number*/
int   in1[1800],  ntt[1800];     /* number of transmission */

/* global function */
void   Te_message();    /*  information encoding */
void   testmode();      /* preparing the message */
void   Te1_message();   /* header encoding */
void   dis_rcv();    /* disable the receiver */
void    ena_rcv();   /* enable the receiver */
void  Initialization();   /* initialization */

/*************************************************************
**/
/*  main()
     Function : get the information packet and send it out
                if channel is free. Otherwise, wait until
                channel is idle, then send the packet. If
                the packet is successfully transmitted,
                (i.e. the positive acknowledgment response
                is correctly received) the transaction
                is done and the packet will be regenerated
                with the arrival rate (1/ count_rate).
    Algorithm : Carrier Sense Multiple Access/ Collision
                Detection with ACK (CSMA/CD with ACK)
                scheme. (For more details, refer the
                chapter 4.)

   Variable definition :
       type_message : the packet type. This can be:
             1) Information packet (type_message = 1),
             2) Positive acknowledgment response packet
                (type_message = 6), or
             3) Negative acknowledgment response packet
                (type_message = 2).
      count_rate  : the reciprocal of the arrival rate.
```

```
      nt   : number of transmission (low significant part).
      incre1 : number of transmission (high significant part).
      txb1 , txb : flag to indicate that the transaction is
                   completed.
      channel_busyb : status of the channel. If
                      a) channel is busy, channel_busyb = 1,
                      b) channel is free, channel_busyb = 0.
      max2_index : the number of information block length.
      out_exit : the process terminate status.
*/

main()
{
int  x;
char loopback;
int    incre1, nt;

      /* initialization */
      handle = fopen("lanrd.dat","w");
      Initialization();
      type_message = 1;  /* set up the packet type */
      dis_rcv();     /* disable the receiving interrupt. This
                        is used to prevent the deadlock
                       situation occured */
      testmode();      /* prepare to encode the information
                       data*/
      Te1_message();  /* encode the data */
      ena_rcv();    /* enable back the receiving interrupt */
      while ( forever ) {
      /*  package  generator */
           count_rate = 0;
           while ( count_rate < rate_of_change )
                 if ( out_exit) goto end_exit;
           do {
              x = rand() ;
           }while ( x > 32700 );
      /* randomly set up the destination address */
        x = rand();
          Destination = Source + 1 ;
        if(x > 18360)
          ++Destination;
        if ( Destination > 3 ) Destination = Destination - 3;
        /* prepare and encode the header */
        type_message = 0x01;   /* set up the packet type */
        dis_rcv();      /* disable the receiving interrupt */
       testmode();        /* prepare to encode the header */
       Te_message();      /* encode the header */
       ena_rcv();        /* enable back the interrupt */
      /* set up the number of transmission, Txb */
        nt = 0; incre1  = 0;
        txb1 = txb = 1;      /* set up the responding flag */
        loopback = 1;     /* transaction is begun */
```

```
          while ( loopback ) {
            /* verify the channel status */
              if ( channel_busyb == 1) {
               }  /* channel is busy */
               else{  /* channel is free */
                if ( DIF == 1 ) {
                  if ( txb1 ) {
                  /* reply is not the positive acknowledgment
                     response packet */
                     ++nt;   /* increase the number
                                of transmission  */
                     if ( nt > 32700 ) {
                         ++incre1;  /* in case the the counter
                                       is out of ranger, increase
                                       the high significant
                                       counter */
                         nt = 0;    /* reset the lower
                                      significant counter */
                     }
                   /* verify the number of back-off time. This
                     will be used for future study */
                     if ( nt > tx_time_rate )  goto outloop ;
                   /* copy the output data into the encoder input
                     buffer */
                    max2_index = 62; /* 62 */
                     for( x = 0; x < 62; x++)
                             E_bufferout[x] = outarray[x];
                  /* reset the tramission index */
                     index_r = index_t = -2;
                     Tx_packet(); /* enable the transmiter */
                  }  /* txb = 1 */
                  else {
                 /* reply is an positive acknowledgment
                             response packet */
outloop:
                  /* store the packet counter */
                     ntt[length] = nt;
                     in1[length] = incre1;
                     ++length;
                 /* check the terminate flag */
                     if (out_exit) goto end_exit;
                /* transaction is done */
                     loopback = 0;
                  }  /* txb = 0 */
                 } /* DIF */
                } /* channel free */
            } /* while loopback */
       } /* while forever */
        /*The execution is over */
end_exit:         /* store the last packet counter */
                  ntt[length] = nt;
                  in1[length] = incre1;
```

```
                ++length;
                /* reinit the interrupt vector */
                UN_Install();
                /* store the results into the disk */
                for ( x = 0; x < length; x++)
                   fprintf(handle,"x%i %i\n",ntt[x],
                       in1[x]);
                fclose(handle);
 }



/*************************************************************/

/* Te_message()
      Function : encode the header.
      Variable definition :
           mask_E : control register for the 8259 expansion
                    interrupt.
           block_length, no_block: The number of encoder
                                   block length.
           E_bufferout : buffer contains the encoder output
                         data.
           outarray : buffer contains the backup encoder
                      output data.
*/

void   Te_message()
{
int i;

en1_again:
        mask_E = 0xfc;  /* set up value for interrupt */
        block_length = no_block = 6;  /* header code */
        infor_E_block();  /*  send K */
        max_index = 31; /* set up the encoder output length */
        Change();   /* encode the header */
        /* This section is used in case that the encoder is
            fail. The procedure will be repeated */
        if ( code_end ) {
                goto en1_again;
        }
        /*store the encoder output data into the buffer

          (outarray) */
        for ( i = 0; i < max_index; i++)
                outarray[i] = E_bufferout[i];
        /* set the index for the encoder output */
        max_index += 31;
```

```
}

/*************************************************************/

/*  testmode
        Function : prepare the information packet.
        Variable definition:
            bufferin : buffer contains the information data.
            max_index1 : number of block length.
*/
void testmode()
{
int i;
char lrc;

        max1_index  = 1;
        /* prepare the packet */
        if ( type_message & 0x01 )
                for ( i = 0; i < 50; i++) bufferin[i] = 0;
                /* reset the buffer */

        /* store the information into the buffer */
        bufferin[0] = Destination ;  /* setup the destination
                                        address */
        bufferin[1] = Source;        /* setup the Source
                                     address */
        bufferin[2] = type_message;/* setup the packet type */
        bufferin[3] = 1;    /* default for the block length */
        /* check for the packet type */
        if ( type_message & 0x01 ) { /* it is information
                                     packet */
           /* restore the actual message */
           for ( i = 0; i < len_message; i++) {
              bufferin[i + 6] = tx[i];
                 if ( !(i % block_new) ) {
                /* set up the start bit for evey blocklength */
                          bufferin[i + 6] |= 0x20 ;
                  /* update the block length */
                          ++max1_index;
                   }
            }  /* restore */
            bufferin[3] = max1_index; /* store the block
                                      length */
        }
        bufferin[4] = block_new; /* setup the information
                                 length */
        /* calculate the BCC */
        for ( lrc = FALSE, i =0; i  < 5 ; i++)
               lrc ^= bufferin[i];
        bufferin[5] = lrc;
        bufferin[0] = bufferin[0] | 0x20; /* setup the start
                                        bit */
```

```
}

/*************************************************************/
/* Initialization
       Function : request for the input data.
       Variable definition :
           DIF   : flag to indicate the transmitted interrupt
                   status.
           outexit : terminate flag.
           blocknew : number of information data.
           tx : buffer contains the input message.
           len_message: number of input data.
*/

void  Initialization()
{
char    c;
int     jndex, i;
long    now, then;

        /* setup the flags and counter */
        DIF = 1;       /* transmitter flag */
        length = 0;    /* pointer to the input data */
        out_exit = 0; /* terminate flag */
        /* set up the mode of operation */
        printf("\n Source: ");  /* input the source address */
        scanf("%d",&i);
        Source = (char)i;
        /* input the arrival rate */
        printf("\n The rate of change in second : " );
        scanf("%d", &rate_of_change);
        /* set up the information length */
        block_new = 20;
        /* get the input message */
        printf("\nInput the message: ");/*input the test
                                          message */
        scanf("%s",txx);
        /* convert the 8 bits input data into 5 bits Galios
              Field data. */
        shift(tx, txx, 5, &jndex);   /* convert 8 bits to 5
                                     bits */
        tx[jndex] = NULL;
        i = 0;
        /* setup the startup time */
        now = biostime(i, then);
        printf("\n current time : %lu  enter new time : ",
             now);
        scanf("%lu",&now);
        while( biostime(i, then) < now);
        /* set up the control register flag */
        mask_E = 0xff;
        /* get the input message length */
```

```
        len_message = strlen(tx);
        /* initialize the control/status register, interrupts
         and timer */
        Install( B1200 + DB7 + SB1 + PODD);
}

/*************************************************************/
/* shift
       Function:  Convert 8 bits per character into
                  "bits" bits per character.
       Variable definition :
             new : buffer contains "bits" bits data packet.
             frame: buffer contains 8 bits data packet.
             bits : Galois Field bit length.
             jndex: The length of the "bits" bits data packet.
             otherbit: the offset of the shifting value.
             char1, char2: 8 bits data.
             value: the offset complement.
    Algorithm: The operations of this procedure can be
               described as follow (Figure A.1).
               The first five bits of the first element of the
               8 bits data (frame) will be copied to the
               first element of the 5 bits data (new). The
               next three bits of the first element of the 8
               bits data (frame) will be combined with
               the two bits of the second element of the 8
               bits data (frame) and then copied to the
               second element of the 5 bits data (new). The
               five bits of the second element of the 8 bits
               data (frame) will be continuous copied to the
               third element of the 5 bits data (new). This
               procedure will repeat until all the 8 bits
               data (frame) are converted into the 5 bits data
               (new).
*/

shift(char new[], char frame[], char bits, int *jndex)
{
unsigned char otherbits, i, j, char1, char2, new1;
unsigned int value, index;
        /* initialize the variable */
        otherbits = 0;
        index = strlen(frame) - 1 ;
        frame[index+1] = 0;  /* set the last index to NULL */

        for ( j = i = 0; i <= index; j++) {
                value = 0xff;  /*set the offset complement */
                if (( otherbits + bits) < 8 ) {
                        /* copy five bits of the 8 bits data
                           (frame) into the 5 bits data
                         (new) */
                        /* adjust the offset complement */
```

```
                        value = value >> otherbits;
                        /* get the data from the 8 bits data
                           (frame) */
                        char1 = (unsigned)frame[i] & 0xff;
                        /* set up the shifting bits */
                        otherbits += bits;
                        /* copied the value to the 5 bits
                           data (new) */
                        new[j] = (char1 & (unsigned
                                 char)value)
                                 >> ( 8 - otherbits);
                    }
                    else {
                        /* combined the # bits from the previous
                           8 bits data (frame) with the # bits
                           in the present 8 bits data (frame)
                           and copied to the 5 bits data (new) */
                           /* get the previous 8 bits data */
                           char1 = frame[i]&0xff;
                           /* get the present 8 bits data */
                           char2 = frame[++i]&0xff;
                           /* get the present bits */
                           new1 = char2 >> ( 16 - otherbits -
                                  bits);
                           /* adjust the offset complement */
                           value = value << ( 8 - otherbits);
                           value = ~ value;
                           value = value & 0xff;
                           /* copied the data into 5 bits data */
                           new[j] = (( char1 & (unsigned
                                     char)value)
                                << ( otherbits + bits - 8 )) +
                                    new1;
                           /* set the new shifting offset */
                           otherbits = otherbits + bits - 8;
                    }
            }
            /* update the output length */
            *jndex = j;
 }

/*************************************************************/
/* Tel_message()
      Function : encode the information.
      Variable definition :
           mask_E : control register for the 8259 expansion
                    interrupt.
           block_length, no_block: The number of encoder
                                   block length.
           E_bufferout : buffer contains the encoder output
                         data.
           outarray : buffer contains the backup encoder
```

```
                        output data.
*/
void
Tel_message()
{
int i;

en1_again:
        block_length = block_new; /* reset the block length */
        no_block = block_length * ( max1_index - 1);
        mask_E = 0xfc;    /* set up value for interrupt */
        infor_E_block();
        /* calculate number of block length */
        max_index = 31 * ( max1_index - 1) ;
        for ( i =0; i < no_block; i++)
            bufferin[i] = bufferin[i+6];
        change();  /* encode the data */
        if ( code_end ) {
            goto en1_again;  /* in case of the encoder
                        failure, repeat the algorithm */
        }
        /* store the encoder output into the outarray for
         later use  */
        for ( i = 0; i < 31; i++)
             outarray[i + 31] = E_bufferout[i];
}

/*************************************************************/
/* dis_rcv
/*    These are the inline assembly codes. This procedure
      is used to disable the interrupts to avoid the deadlock
      situation occurs.
*/
void
 dis_rcv()
{
asm     push  ax            /* save the entry registes */
asm     push  dx
asm     mov   al, 00h       /* disable the transceiver
                               interrupts */
asm     mov   dx, 03f9h
asm     out   dx, al
asm     pop dx             /* restore the entry registers */
asm     pop  ax
}

/*************************************************************/
/* ena_rcv
/*    These are the inline assembly codes. This procedure
      is used to enable the receiving interrupts to avoid the

      deadlock situation occurs.
```

```
*/

void
ena_rcv()
{

asm     push  ax            /* save the entry registers */
asm     push  dx
asm     mov   al, 01h     /* enable the receiving interrupt */
asm     mov   dx, 03f9h
asm     out   dx, al
asm     pop   dx          /* restore the entry registers */
asm     pop   ax

}
```

APPENDIX B

ASSEMBLY SOURCE CODE

```
;Module name: int_han.asm
;Creater    : Victor C. Phan
;Date       : Jan. 1, 1992
;Version    : 1
;*************************************************************
;
;                  int_han.asm - 80C86 Assembly code
;
;*************************************************************
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                  Define MARCO
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;

OSCall  MACRO   DosFunct            ; calls the Operating
System
        push    ax
        mov     ah, DosFunct
        int     21h
        pop     ax
        ENDM
        Write   MACRO   Char        ; displays one character
        push    dx
        mov     dl, Char            ; destroys DL
        OSCall  2
        pop     dx
        ENDM
        Wr_str  MACRO   Str_pnt     ; uses DOS call to display
string
        push    dx
        mov     dx, OFFSET Str_pnt ; DS:DX point to string
        OSCall  09h                 ; Print String
        pop     dx
        ENDM
        OPort   MACRO  Port,Val     ; Put the character (Val)
        push   dx                  ; to the output port (Port)

        mov    dx,Port
        mov    al,Val
        out    dx, al
```

```
        pop     dx
        ENDM
        IPort   MACRO Port          ; input the character from
        push    dx                    ; the input port (Port).
        mov     dx, Port              ; The character will be
        in      al,dx                 ; returned through the
        pop     dx                    ; MACRO OSCall IPort.
        ENDM
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;
;
;       Definition of symbolic names for the registers
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;
;
;
;
LF          EQU   0ah           ; ASCII control character key.
                                ; Line feed.
CR          EQU   0dh           ; Carriage return.
ACK         EQU   06h           ; Positive acknowledgment.
SOT         EQU   02h           ; Start of Text.
NAK         EQU   02h           ; Negative acknowledgment.
End_st      EQU   '$'           ; DOS end of string
ic_mas      EQU   21h           ; port address, 8259 mask register
Pic_eoi     EQU   20h           ; port address, 8259 eoi instr.
                                ; control reg
Int_mask    EQU   18h           ; Mask for 8259 all the interrupts
Buf_len     EQU   70            ; buffer length.
ICW1E       EQU   13h          ; Internal Command Word register 1.
ICW2E       EQU   00h          ; Internal Command Word register 2.
ICW4E       EQU   00h          ; Internal Command Word register 3.
max_count   EQU   0009h         ; timing window.
control     EQU   20h           ; Control register.
OCW3E       EQU   00001100b     ; set the polling status to 1 all
                                ; the rest to zeros

;
;
;
; switching address to activate the chip selector
; bit in the latch.
intadd2           EQU     0000001110100010b ; latch 2
intadd3           EQU     0000001110100011b ; latch 3

intadd4           EQU     0000001110100100b ; latch 4
intadd5           EQU     0000001110100101b ; latch 5
intadd6           EQU     0000001110100110b ; latch 6
intadd7           EQU     0000001110100111b ; latch 7
```

```
;
; Port constants
Base_port       EQU     03F8h
Port_A          EQU     Base_port + 0     ; com_data
Port_B          EQU     Base_port + 1     ; com_ier 9
Port_C          EQU     Base_port + 2     ; IIR
Port_D          EQU     Base_port + 3     ; b
Port_E          EQU     Base_port + 4     ; com_mcr c
Port_F          EQU     Base_port + 5     ; com_sts d
Port_G          EQU     Base_port + 6     ; e
; set modem control register DTR and OUT2 bits
Modem           EQU     00001011b
; input interrupt enable for serial port controller
Int_A           EQU     00000001b
; determine when output buffer is available
Data_out        EQU     00100000b
RCV_Err         EQU     10011110b  ; test for receiver error
;                       76543210
DGROUP      GROUP _DATA
;
_TEXT SEGMENT byte public 'CODE'
        ASSUME cs:_TEXT,ds:DGROUP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;
;                   Procedure  Install (configuration)
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;
; Procedure places the interrupt vectors with a custom
; interrupt handler and enables interrupts from the 8259.
;
;  On entry :
;       Configuration : The configuration of the I/O port.
;  On exit  :
;       Nothing.
;
PUBLIC _Install
_Install PROC    NEAR
    push bp             ; preserve bp in order to use the bp
    mov  bp,sp          ; to access parameters on the stack
    push cx             ; save the entry registers
    push bx             ;          "
    push ax             ;          "
    push dx             ;          "
    push ds             ;          "
;
; intialize the 8259 expansion controller
;
    mov  al, ICW1E      ; Initialize the ICW 1 register
```

```
                        ;       00010011
                        ;          || ||--> ICW4 needed
                        ;          || |---> only one.
                        ;          ||---> Edge trigger.
                        ;          |----> Init mode.
    mov  dx, intadd2    ; Write Hex 13 to ICW 1.
    out  dx, al
    mov  al, ICW2E      ; Initialize the ICW 1 register.
                        ; Set all the interrupt addresses
                        ; to zero.
    mov  dx, intadd3    ;  Write 00 to the ICW 2.
    out  dx, al
    mov  al, ICW4E      ; Initialize the ICW 4 register
                        ;       00000000
                        ;          |||||--> MCS 80/85.
                        ;          ||||--> Normal EOT.
                        ;          |||---> Non buffer
                        ;          ||----> mode.
                        ;          |---> not special fulling
                        ;                nest mode.
    mov  dx, intadd3    ; Write 00 to ICW 4
    out  dx, al
    mov  dx, intadd3    ; set the mask for the expansion
                        ; register.
    mov  al, _mask_E    ;
    out  dx, al
    mov  al, 20h        ; Write hex 20 to the mask register
                        ; to init the expansion board.
    mov  dx, intadd2
    out  dx, al         ; send EOI to 8259 expansion
    mov  dx, intadd2
    mov  al, OCW3E      ; write the OCW 3 to set the
                        ; polling status.
    out  dx, al
;
; get the vectors for the following interrupt requests
; and store them into the array (com_int) :
;      - Timing,
;      - Transceiver,
;      - RS31KED,
;      - Control-break.
;
    xor  bx, bx             ; clear the register.
    mov  com_int[bx], 0ch   ; transceiver vector.
    inc  bx
    mov  com_int[bx], 1bh   ; control-break.
    inc  bx
    mov  com_int[bx], 08h   ; Timing.
    inc  bx
    mov  com_int[bx], 0bh   ; RS31KED.
;
; get the offset for each of the above interrupt and
```

```
; store them into the array (intoff).
;
    xor  bx, bx
    mov  dx, offset intTR_han ; address of transceiver intrp.

                              ; handler
    mov  intoff[bx], dx
    add  bx, 2
    mov  dx, offset intCR_han ; address of control break
                              ; intrp. handler
    mov  intoff[bx], dx
    add  bx,2
    mov  dx, offset intT_han  ; address of timer intrp.
                              ; handler
    mov  intoff[bx], dx
    add  bx,2
    mov  dx, offset intED_han ; address of encode/decode
                              ; intrp. handler

    mov  intoff[bx], dx



;
; set up the communication port
;
    mov  dx,0             ; set up the communication port
    mov  al, [bp+ 4]      ; get the configuration request.
    mov  ah, 0            ; call ROM BIOS to init the
    int  14h              ; the communication port.
;
; reset the index
;
    xor  bx, bx           ; set the index to zero
    xor  cx, cx
;
; replace the old interrupt vectors by the new ones.
;
again:
    mov  dx, bx
    mov  bx, cx
    mov  al, [com_int+bx] ; get interrupt handler address
    mov  temp2, al
                          ; swap the location for the purpose of


                          ; passing parameters
    OSCall  35h           ; ES:BX = address
    mov  ax, bx           ; swap the index register and
    mov  bx, dx           ; index array
    mov  intc_seg[bx],es  ; save segment
    mov  intc_offs[bx],ax ; save offset
```

```
    mov  dx, intoff[bx]
    mov  al, temp2
    push cs              ; set the code segment to
    pop  ds              ; data segment.
    OSCall  25h          ; set address of the new custom
                         ; interrupt handler routine for
                         ; specified machine
                         ; intrp
    pop  ds              ; restore the data segment.
    push ds              ; push back to stack for later use.
    add  bx, 2           ; get the next interrupt vector.
    inc  cx
    cmp  cx, 3           ; Is it all done ?
    jle  again


;
; initialize the control registers and enable interrupts
;
    OPort   Port_E, Modem
    OPort   Port_B, Int_A ; 00000001
                          ;        |-> enable the receving
                          ;            interrupt in the
                          ;            transceiver and
                          ;           disable all the rests.

    in   al,Pic_mask     ; read current 8259 intrp. mask
    and  al,not Int_mask ; reset mask for this Com port
    out  Pic_mask,al     ; write back 8259 intrp. mask


;
; end of Intall
;
    pop  ds              ; restore the entry registers
    pop  dx              ;
    pop  ax              ;
    pop  bx              ;
    pop  cx              ;
    pop  bp              ; retrieve the base pointer
    ret                  ; end of procedure.
_Install ENDP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;               Procedure Change
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Procedure enables the RS31KED interrupt. The global
;   value (mask_E) contains the enabling interrupt
;   vector.
```

```
;
;       On entry :
;           mask_E : enabling interrupt vector.
;       On exit  :
;           set up the waiting flag (code_end) to indicate
;           that the RS31KED interrupt is enable.
;
PUBLIC _Change
_Change PROC   NEAR
     push  bp              ; preserve the base pointer.
     mov   bp, sp
     push  dx              ; save the entry registers.
     push  ax
     mov   _code_end, 1    ; set up the waiting flag
     mov   dx, intadd3     ;
     mov   al, _mask_E     ; enable the requesting interrupt.
     out   dx, al          ;
     pop   ax              ; retrieve the entry register.
     pop   dx              ;
     pop   bp              ; restore the base pointer.
     ret                   ; end of the procedure.
_Change ENDP


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;             Procedure Tx_packet
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Procedure enables the transmitter interrupt in the
;   communication port.
;
;       On entry : nothing.
;       On exit : set up the flag (DIF) to indicate that
;                 the transmitter interrupt is enable.
;
PUBLIC _Tx_packet
_Tx_packet PROC   NEAR
     push  bp              ; preserve the base pointer.
     mov   bp, sp
     push  dx              ; save the entry registers.
     push  ax
     mov   _DIF, 0         ; set up the flag for transmitted
     mov   al, 02h
     mov   dx, 03f9h       ; get the transmitter int. vector.
     out   dx, al          ; enable the transmitter interrupt.
no_tx:
     pop   ax              ; retrieve the entry registers
     pop   dx              ;
     pop   bp              ; restore the base pointer.
```

```
     ret                    ; end of the procedure.
_Tx_packet ENDP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                       Procedure Encoder.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Procedure presets the encoder chip.
;
;       On entry :
;           block_length : number of block to be encoded.
;       On exit  :
;             nothing.
;
PUBLIC    _infor_E_block
_infor_E_block  PROC   NEAR
     push bp                    ; preserve the base pointer.
     mov  bp, sp
     push dx                    ; save the entry registers
     push ax
     mov  dx, intadd6
     mov  al, _block_length     ; input the block length
     out  dx, al
;
;   delay
;
     mov  al, 7fh
next3:
     nop
     dec  al
     jnz  next3
;
; reset the encoder
;
     mov  dx, intadd4           ; reset the encoder
     mov  al, 80h
     out  dx, al
;
; end
;
     pop  ax                    ; retrieve all the data
     pop  dx
     pop  bp
     ret
_infor_E_block ENDP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                       Procedure Decoder.
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Procedure presets the decoder chip.
;
;     On entry :
;           block_length : number of block to be decoded.
;     On exit  :
;           nothing.
;
PUBLIC    _infor_D_block
_infor_D_block  PROC   NEAR
     push bp                    ; preserve the data
     mov  bp, sp
     push dx
     push ax
;
; setup the type as well as the cl to indicate the
; length of dropped information system in a code.
;
     mov  dx, intadd7
     mov  al, 0ffh              ; 11111111
                                ;   ||||||--> code length = 5
                                ;   |--> transform code.
     out  dx, al
;
; delay
;
     mov  al, 7fh
next6:
     nop
     dec  al
     jnz  next6
;
; set up the block length for the decoder.
;
     mov  dx, intadd6
     mov  al, _block_length  ; input the block length
     out  dx, al
;
; delay
;
     mov  al, 7fh
next2:
     nop
     dec  al
     jnz  next2
;
; reset the decoder
;
     mov  dx, intadd5           ; reset the decoder
     mov  al, 80h
     out  dx, al
```

```
;
; end.
;
     pop  ax                  ; retrieve all the data
     pop  dx
     pop  bp
     ret
_infor_D_block ENDP


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;              Procedure UN_Install
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Procedure restores the previous IRQ5 interrupt vector
; and enables interrupts from the 8259.
;
;
;
PUBLIC _UN_Install
_UN_Install PROC    NEAR
        push   bp               ; preserve the base pointer.
        mov    bp,sp
        push   ax               ; save the entry registers.
        push   dx
        push   bx
        push   cx
;
; clear the registers.
;
        xor    cx, cx
        xor    bx, bx
;
; mask for all interrupts
;
        in     al,Pic_mask     ; read current 8259 intrp. mask
        or     al,Int_mask      ; set mask for this Com port
        out    Pic_mask,al      ; write back 8259 intrp. mask
;
; get the old interrupt vectors from the com_int and interrupt
; offsets from the intc_off and put them back to the original
; places.
;
again1:
        push   ds               ; save the data segment for
                                ; later use.
        mov    dx, bx
        mov    bx, cx
        mov    al,com_int[bx] ; get interrupt vector
        mov    bx, dx
```

```
        mov     dx,intc_offs[bx]     ; saved offset
        mov     ds,intc_seg[bx]      ; saved segment
        OSCall  25h                  ; restore original addr. of
                                     ; interrupt handler
        pop     ds                   ; restore the data segment.
        add     bx, 2                ; get the next interrupt
                                     ; pointer.
        inc     cx
        cmp     cx, 3                ; Is it all done ?
        jle     again1
;
; end
;
        pop     cx                  ; retrieve all the entry
                                     ; registers.
        pop     bx
        pop     dx
        pop     ax
        pop     bp                   ; restore the base pointer.
        ret
_UN_Install ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                            RS Codec Module
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Procedure intED_han is a custom interrupt handle for RS
; Codec.
;
;     On entry :
;        buffer_in : buffer contains the
;                    encoder input data.
;        r_message : buffer contains the decoder input
;                    data.
;        max_index : max. number of encoder output data or
;                    max. number of decoder input data.
;        no_block  : max. number of encoder input data or
;                    max. number of decoder output data.
;     On exit  :
;        E_bufferout : buffer contains the encoded data.
;        D_bufferout : buffer contains the decoded data.
;
intED_han PROC   NEAR
     sti                          ; turn interrupts back on
     push bp                     ; save all the neccessary
                                  ; registers.
     mov  bp, sp
     push ax
     push ds
     push bx
     push dx
```

```
     mov  ax,  seg DGROUP  ; put the address of the data
                           ; segment into
     mov  ds, ax           ; the data segment register
     cli                   ; disable all the interrupts
;
; get the polling status
;
     mov  dx, intadd2
     mov  al, OCW3E        ; get the polling status
     out  dx, al
     mov  dx, intadd2
     in   al, dx           ; read the expansion 8259A
                           ; interrupt level to status byte.
     and  al, 07h
;
;  branch to active level service routines
;
     cmp  al, 1            ; is it encoder output ?.
     je   RDYOE
     cmp  al, 2            ; is it decoder output ?.
     je   RDYID
     cmp  al, 3            ; is it decoder input ?.
     je   RDYOD
     cmp  al, 0            ; is it encoder input ?.
     je   RDYIE
     jmp  end1
;
; encoder input routine
;
RDYIE:
     mov  bx, asc_in       ; get the output pointer
     cmp  bx, _no_block    ; Is it equal to max. length of
                           ; the encoder input data.
     jl   end7
     jmp  end3
end7:
     mov  al, [_bufferin + bx] ; get the data from the
                               ; bufferin
     mov  temp7, al
     mov  temp4, 07ffh         ; set the delay value. This
                               ; value needs to be changed
                               ; according to the speed of
                               ; CPU.
     mov  dx, intadd4
     out  dx, al           ; output the data to encoder
     inc  bx               ; increase the buffer pointer
     mov  asc_in, bx
     jmp  end1
;
; encoder output routine
;
RDYOE:
```

```
        mov   bx, asc_out       ; get the bufferout pointer
        cmp   bx, _max_index    ; Is it equal to max out index
        jge   end2
        mov   dx, intadd4
        in    al, dx            ; get the data from encoder
        mov   temp4, 07ffh      ; set the delay value. This value
                                ; needs to be changed according
                                ; to the speed of CPU.
        mov   [_E_bufferout + bx], al   ; store data into buffer
                                        ; out
        mov   temp7, al
        inc   bx                ; increase the buffer out pointer
        mov   asc_out, bx
        jmp   end1
;
; decoder input routine.
;
RDYID:
        mov   bx, asc_in        ;get the output pointer
        cmp   bx, _max_index    ; Is it equal to max in index
        jge   end5
        mov   al, [r_message + bx] ; get the data from the
                                   ; bufferin
        mov   temp4, 01fh       ; set the delay value. This value
                                ; needs to be changed according
                                ; to the speed of CPU.
        mov   temp7, al
        mov   dx, intadd5
        out   dx, al            ; output the data to encoder
        inc   bx                ; increase the buffer pointer
        mov   asc_in, bx
        jmp   end1
;
; decoder output routine.
;
RDYOD:
        mov   bx, asc_out       ; get the bufferout pointer
        cmp   bx, _no_block     ; Is it equal to max out index
        jge   end2
        mov   dx, intadd5
        in    al, dx            ; get the data from encoder
        mov   temp4, 001fh      ; set the delay value. This value
                                ; needs to be changed according
                                ; to the speed of CPU.
        mov   [_D_bufferout + bx], al   ; store data into buffer
        mov   temp7, al         ; out
        inc   bx                ; increase the buffer out pointer
        mov   asc_out, bx
        jmp   end1
;
; end of the decoder input.
;
```

```
end5:
     mov  al, 0f7h          ; disable the decoder input intrp.
                            ; but still enable the decoder
                            ; output interrupt.
                            ;   11110111
                            ;       ||--> disable the decoder
                            ;       |     input interrupt.
                            ;       |--> enable the decoder
                            ;            output interrupt.
     mov  dx, intadd3
     out  dx, al
     jmp  end1
;
; end of the encoder input.
;
end3:
     mov  al, 0fdh          ; disable the encoder input
                            ; interrupt.
                            ; 11111101
                            ;       ||--> disable the encoder
                            ;       ||    input interrupt.
                            ;       |--> enable the encoder
                            ;            output interrupt.
     mov  dx, intadd3
     out  dx, al
     jmp  end1
;
; end of the encoder output or decoder output.
;
end2:
     mov  _code_end, 0      ; reset the waiting flag to
                            ; indicate
                            ; that the RS31KED interrupt is
                            ; disable.
     xor  bx, bx
     mov  asc_in, bx        ; reset the buffer pointer.
     mov  asc_out, bx
     mov  al, 0ffh          ; disable the encoder/decoder
                            ; interrupt.
     mov  dx, intadd3
     out  dx, al
;
; continuing executing the interrupt routine.
;
end1:
     mov  ax, temp4         ; get the delay time
;
; delay
;
next1:
     nop
     dec  ax
```

```
     jnz  next1
;
; end
;
     sti                    ; enable the in board 8259
                            ; interrupt
     mov  temp4, 2          ; reset the delay time.
     mov  al, 20h           ; send EOI to 8259
     out  Pic_eoi, al
     mov  dx, intadd2
     out  dx, al            ; send EOI to 8259 expansion
     pop  dx               ; retrieve all the entry registers
     pop  bx
     pop  ds
     pop  ax
     pop  bp                ; restore the base pointer.
     iret                   ; end of interrupt routine.
intED_han   ENDP


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                           Timer Module
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Procedure intT_han is a custom interrupt handler for the
; timer
; Its function is to verify the clock tick timer, check the
; channel status, and calculate the execution time.
;
;       On entry  :
;            count_rate : number of clock tick.
;            txb   :  the flag to indicate the type of ;  ;
;                     message.
;                     It can either be
;                     1) information or negative
;                     acknowledgment
;                     response (txb = 0). or
;                     2) positive acknowledgment
;                     response (txb=1).
;       On exit :
;             nothing.
;
intT_han PROC              NEAR
        push    bp                ; preserve the base pointer.
        mov     bp, sp
        push    ax                  ; save the entry registers
        push    dx
        push    ds
        mov     ax,  SEG DGROUP   ; put the address of the
                                  ; data segment
```

```
        mov     ds, ax              ; into the data segment
                                    ; register
;
; get the clock tick timer
;
        mov     ax, _count_rate     ; increase the counter rate
        cmp     ax, 600             ; Is it equal to the preset
        jge     nocount             ; timing clock_tick.
                                    ; This is used to set up the
                                    ; case where only 1 machine
                                    ; is running, the other two
                                    ; are listenning.
        inc     ax
        mov     _count_rate, ax
;
; chech the response status
;
nocount:
        mov     al, _txb            ; check for the ack reply
        cmp     al, 1               ; Is it ACK?
        je      notxb
        mov     _txb, 1             ; set the flag in case of
        mov     _txb1, 0           ; receiving the ACK message
        jmp     busy
;
; check the channel status
;
notxb:
        mov     ax, count
        cmp     ax, max_count       ; Is channel busy ?
        jge     busy
        inc     ax                  ;  channel busy
        jmp     outtime
busy:
        mov     _channel_busyb, 0 ; channel free
        mov     ax, 0
outtime:
        mov     count, ax
;
; calculate the execution time
;
        mov     ax, base2           ; check the execution time
        dec     ax
        cmp     ax, 0
        jne     go_on2
        mov     _out_exit, 1       ; set up the terminate flag
        mov     ax, 7ff0h           ; reset the counter
go_on2:
        mov     base2, ax
;
; end
;
```

```
        mov     al, 20h          ; send EOI to 8259
        out     Pic_eoi, al
        pop     ds              ; retrieve all the entry
                                 ; registers
        pop     dx
        pop     ax               ;
        pop     bp              ; restore the base pointer.
        iret
intT_han ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                   Control-break Interrupt Module
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Procedure intCR_han is a custom interrupt handler for the
; control-break
;
;       On entry    :  control break key is hit.
;       On exit     :  the terminate flag is set.
;
intCR_han PROC              NEAR
        sti                         ; turn the interrupt back on
        push    bp
        mov     bp, sp              ; preserve the base pointer
        push    ax                   ; save the entry registers
        push    bx
        push    cx
        push    ds
        mov     ax,  SEG DGROUP     ; put the address of the
                                     ; data segment
        mov     ds, ax               ; into the data segment
                                     ; register
        cli                          ; turn the interrupt off
;
; set the terminate flag
;
        mov     _out_exit, 1
        sti                          ; turn interrupt back on
;
; end
;
        mov     al, 20h              ; send EOI to 8259
        out     Pic_eoi, al
        pop     ds                  ; restrieve all the entry
                                     ; registers
        pop     cx
        pop     bx
        pop     ax
        pop     bp                  ; restore the base pointer.
        iret
intCR_han ENDP
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                       Transmission Module
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  Procedure to transmit the character to communication port
;
;       On entry  :
;            E_bufferout : buffer contains the output data.
;       On exit   :
;            Reset the transmitted flag (DIF).
;
TX_char  PROC  NEAR
;
; Send out the Start Of Text (SOT) characters.
;
        mov       ax, 0
        cmp       _index_t, ax      ; Is it any SOT to be
                                    ; transmitted
        jge       start_message
        mov       al, SOT
        OPort   Port_A, al          ; transmit SOT
        inc       _index_t
;
; delay
;
        mov       bx, 7f00h
loop_tx:
        dec       bx
        cmp       bx, 0
        jle       dis1_out
;
; check the collision status.
;
        IPort   Port_F              ; wait for the echo
        and       al, 01h
        cmp       al, 0
        je        loop_tx           ; is it any character at
                                    ; receiver register?
        call      Rx_char           ; receive the character
        mov       ax, _index_r
        cmp       ax, _index_t      ; collision ?
        je        dis2_out
        mov        ax, -2           ; if collision, reset the
        mov       _index_t, ax      ; index_t
;
; branching instruction is needed here for a far jump.
;
dis1_out:
        jmp       dis_out
dis2_out:
```

```
        jmp     out_tx
;
; start sending out the character
;
start_message:
        mov     ax, _index_t
        cmp     ax, _max2_index  ; Is the end of message ?
        je      dis_tx
        inc     ax               ; check for the last index
        cmp     ax, _max2_index  ; due to the hardware
                                 ; defection
                                 ; we need to set up a long
                                 ; delay following a
                                 ; terminate character (hex
                                 ; ff) at the end of the
                                 ; packet transmission.
        jne     no_time_out
        call    delay_in         ; set the delay for last
                                 ; character
        call    delay_in
        call    delay_in
        mov     al, 0ffh         ; transmit the last
                                 ; character.
        OPort   Port_A, al
        jmp     last_entry
;
; send the data.
;
no_time_out:
        mov     bx, _index_t
        mov     al, [_E_bufferout + bx]
        OPort   Port_A, al       ; transmit the character
last_entry:
        inc     _index_t
        jmp     out_tx
;
; send the terminate character.
;
dis_out:
        call    delay_in
        call    delay_in
        call    delay_in
        mov     al, 0ffh
        OPort   Port_A,al        ; transmit the last
                                 ; character
;
; disable the transmitter
;
dis_tx:
        mov     _max2_index, 62
        OPort   Port_B, Int_A    ; disable the transmitter
                                 ; interrupt
```

```
        mov     count, 0
        mov     _channel_busyb, 1 ; set the channel status to
                                  ; busy.
        mov     _index_r, -2      ; reset the receive index
        mov     ax, _count_rate
        mov     _DIF, 1           ; turn off the transmitted
                                  ; flag
out_tx:
        ret
TX_char  ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                  delay procedure
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   short delay time (1fff)
;
delay_out PROC NEAR
        push    bx
        mov     bx, 1fffh
loop_r:
        dec     bx
        cmp     bx, 0
        jg      loop_r
        pop     bx
        ret
delay_out ENDP

;
; long delay time (7fff)
;
delay_in PROC NEAR
        push    bx
        mov     bx, 7fffh
loop_t:
        dec     bx
        cmp     bx, 0
        jg      loop_t
        pop     bx
        ret
delay_in ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;              Procedure tmessage
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  procedure prepares to encode the acknowledgment response
;  packet
;
```

```
;         On entry   :
;               nothing.
;         On exit :
;             E_bufferin : buffer contains the encoded data.
;
tmessage PROC NEAR
;
; prepare the acknowledgment packet and store into
; the buffer (bufferin).
;
nextse:
        xor     bx, bx              ; clear the index
        mov     BCC, 0              ; set up the BCC
; store the destination address
        mov     al, _Destination
        mov     [_bufferin + bx], al
        xor     BCC, al             ; calculate the check sum
        inc     bx                  ; increase the index
; store the source address
        mov     al, _Source
        mov     [_bufferin + bx], al
        xor     BCC, al             ; calculate the check sum.
        inc     bx
; store the packet type
        mov     al, _type_message
        mov     [_bufferin + bx], al
        xor     BCC, al
        inc     bx
; store the block length
        mov     al, 1
        mov     [_bufferin + bx], al
        xor     BCC, al
        inc     bx
; store the information data length.
        mov     al, 20
        mov     [_bufferin + bx], al
        xor     BCC, al
        inc     bx
; store the checksum
        mov     al, BCC
        mov     [_bufferin + bx], al
; set the start bit
        mov     al, _Destination
        or      al, 20h
        xor      bx, bx
        mov     [_bufferin + bx], al
; prepare to encode the data.
        mov     _block_length, 6    ; set up the block length
        mov     _mask_E, 0fch       ; set up the encoder mask
        call    _infor_E_block
        mov     _no_block, 6        ; set up the encoder input
                                    ; block length.
```

```
        mov     _max_index, 31   ; set up the encoder output
                                  ; block length
        call    _Change
        cmp     _code_end, 1      ; wait for the output
        jne     nextsl
        mov     al, 0ffh         ; disable the RDYO interrupt
        mov     dx, intadd3       ; this step is used in case
        out     dx, al           ; that the encoded is failed
                                  ; to encode the incoming
                                  ; data. The encoded
        jmp     nextse            ; procedure will be
                                  ; repeated.
;
; set up the index.
;
nextsl:
        mov     _max2_index, 31  ; save the transmitted data
                                  ; length
        mov     _index_t, -2     ; reset the transmitted data

                                  ; index
        mov     _index_r, -2      ; reset the received data
                                  ; index.
        ret
tmessage ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                  Procedure  intTR_han
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Procedure int_han is a custom interrupt handler for the
; serial port interrupt which places received characters into
; a buffer or transmit the character over serial port.
;
;
intTR_han PROC             NEAR
        sti                          ; turn the interrupt back on
        push    bp
        mov     bp, sp               ; preserve the base pointer
        push    ax                    ; save the entry registers
        push    bx
        push    cx
        push    dx
        push    ds
        mov     ax,  SEG DGROUP      ; put the address of the
                                     ; data segment
        mov     ds, ax               ; into the data segment
                                     ; register
        cli                          ; clear the interrupts for
                                     ; the pointer manipulation
;
; get the transceiver status
```

```
;
        IPort   Port_C
        cmp      al, 2
        je       TX              ; Is it the transmitter
                                 ; interrupt ?
        cmp      al, 4
        je       RCV               ; Is it the receiver
                                   ; interrupt ?
        jmp      next
;
; send the character
;
TX:
        call     TX_char           ; transmit the character
        jmp      next
;
; prepare to send the acknowledgment response packet.
;
next76:
        call     delay_in          ; time our delay
        call     delay_in
        call     delay_in
        call     tmessage        ; prepare the acknowledgment

                                   ; packet
        mov      al, temp2         ; restore the destination
        mov      _Destination, al
        call     _Tx_packet        ; send the packet
        jmp      next26
;
; receive the character
;
RCV:
        call     Rx_char            ; receiver the character
;
; check for the EOI requirement.
;
        mov      ax, temp
        cmp      ax, 1
        je       next26            ; Is it EOI already sent ?
;
; check the message status.
;
        cmp      ax, 2
        je       next76            ; Is the message correctly
                                        ; received ?
next:
;
; end
;
        sti
        mov      al, 20h            ; send EOI to 8259
```

```
        out     Pic_eoi, al
        mov     temp, 0             ; reset the index
        jmp     nexit
next26:
        mov     temp, 0             ; reset the index
nexit:
        pop     ds                  ; restore the entry
registers.
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        pop     bp          ; retrieve the base pointer.
        iret
intTR_han ENDP
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                       Rx_header Procedure
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Procedure verifies the header of the receiving packet.
;
;       On entry :
;          r_message : buffer contains the receiving message.
;       On exit  :
;           txb  : indicate the receiving packet type.
;
Rx_header  PROC NEAR
        sti                             ; turn on the interrupts
        mov      al, 20h                ; send EOI to 8259
        out      Pic_eoi, al            ;
        mov     temp, 1                ; set the flag to indicate
                                        ; that we do not need send
                                        ; EOI at the end of the
                                        ; interrupt.
;
; prepare decoding the header.
;
        mov      ah, 0
        mov      bx, 0
        mov     al, 20h                ; set up the start bit for

                                        ; encoding data.
        or       [r_message + bx], al
loopy:
        mov     _block_length, 6    ; set up the ouput block
                                        ; length.
        mov      _mask_E, 0f3h          ; mask for enabling the
                                        ; decoder
```

```
        call    _infor_D_block      ; set up the information
                                    ; block length
        mov     _max_index, 31      ; set up the input block
                                    ; length
        inc     ah
        mov     _no_block, 6
        call    _Change             ; enable the decoder
        mov     al, _code_end
        cmp     al, 1               ; wait for the output
        jne     loopy1
        cmp     ah, 30              ; This is used for the
                                    ; case where the decoder
                                    ; is failed to decode the
                                    ; The decoding procedure
                                    ; will then be repeated
        jne     loopy
        jmp     reset1
loopy1:
;
; calculate the checksum
;
        mov     BCC, 0              ; set check sum to false
        xor     bx, bx              ; clear the index
loopBCC:
        mov     al,  [_D_bufferout + bx]
        and     al, 1fh
        xor     BCC, al
        inc     bx                  ; calculate the checksum
        cmp     bx, 6
        jne     loopBCC             ; Is it all done ?
;
; verify the checksum
;
        mov     al, BCC
        cmp     al, 0               ; check the check sum
        jne     reset1
;
; verify the source address
;
        xor     bx, bx              ; reset the index
        mov     al, [_D_bufferout + bx]
        and     al, 1fh
        cmp     al, _Source         ; Is it me?
        jne     reset1
; verify the packet type
;
        mov     bx, 2
        mov     al, [_D_bufferout + bx]
        cmp     al, ACK             ; Is it Positive
                                    ; acknowledgment
                                    ; response (ACK) ?
        je      reset2
```

```
        cmp     al, NAK             ; Is it Negative
                                    ; acknowledgment
                                     ; response (NAK) ?
        je      reset1
        jmp     reset3
;
; end of the Rx_header in case of wrong header.
;
resett:
        jmp     reset1
;
; end of the Rx_header in case of positive acknowledgment
;   response.
;
reset2:
        mov     _txb, 0             ; set the flag to indicate

                                     ; that
                                     ; this is a ACK response.
reset1:
        mov     _index_r, -2        ; reset the receiver index
        ret
;
; calculate the information data length.
;
reset3:
        mov     bx, 3
        mov     ax, 31              ; set the packet pointer
                                    ; to the last index.
        mov    dl, [_D_bufferout + bx] ; get the information
                                        ; block length.
reloop:
;
; use "additive" algorithm to calculate the length.
;
        dec     dl
        cmp     dl, 0
        je      retro
        add     ax, ax
        jmp     reloop
retro:
;
;  this is for safety only.
;
        cmp     ax, 65
        jl      pass_ok
        mov     ax, 62
pass_ok:
;
; store the Source address for later use.
;
        mov     lmessage, ax        ; store the length of
```

```
                                        ; message
        mov     bx, 1                   ; get the index
        mov     al, [_D_bufferout + bx] ; get the address
        mov     Desttemp, al            ; store for later use.
        ret
Rx_header  ENDP
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;             Rx_packet  procedure
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Procedure verifies the correctiveness of the receiving
; data.
;
;        On entry :
;            r_message : contains the receiving information
;                        data.
;        On exit :
;            temp  : indicate that the packet is correctly
;                     received (temp = 2).
;            type_message : packet type.
;            temp1 : flag to indicate that we do not need
;                    to send EOI at the end of the interrupt.
;                     (temp = 1).
;        Desttemp : the setup Destination address.
;
Rx_packet  PROC NEAR
        sti                             ; turn the interrupt back
                                          ; on
        mov     al, 20h                   ; send EOI to 8259
        out     Pic_eoi, al               ;
        mov     temp, 1                   ; set up the flag to
                                          ; indicate we do not need
                                          ; to send EOI.
;
; Prepare to decode the receiving data
;
        mov     ah, 0
        mov     bx, 31                  ; get the last pointer for
                                          ; the decoder output
                                          ; buffer
        mov     al, 20h                   ; get the start bit.
        or      [r_message + bx], al
loopRx:
        mov     _mask_E, 0f3h             ; mask for enabling the
                                          ; decoder
        mov     al, _block_new          ; get the input block
                                          ; length
        mov     _block_length, al
        call    _infor_D_block          ; set up the information
```

```
                                        ; block length
        mov     ax, 62                  ; get the input block
                                        ; length
        mov     _max_index, ax          ; set up the index length
        inc     ah
        mov     _no_block, 20           ; get the output block
                                        ; length
        mov     ax, 31
        mov     asc_in, ax              ; set up the starting
                                        ; point for output.
        call    _Change                 ; enable the decoder
        mov     al, _code_end
        cmp     al, 1                   ; wait for the output
        jne     loopR1
        cmp     ah, 30                  ; in case of the decoder
                                        ; failure, reset the
                                        ; decoder and repeat th.
                                        ; procedure.
        jne     loopRx
        jmp     M1_reset
loopR1:
;
; calculate the checksum.
;
        mov     BCC, 0                  ; set check sum to false
        mov     al, 0                   ; set flag to check for
                                        ; good message
        xor     bx, bx
loopRCC:
;
; verify the receiving information
;
        mov     al, [_D_bufferout + bx] ; get the receiving
                                        ; data
        mov     ah, [_tx + bx]          ; get the previous known
                                        ; input data.
        and     al, 1fh                 ; get rid of the start
        and     ah, 1fh                 ; bits.
        cmp     ah, al                  ; check every character
        je      ok_M
        mov     BCC , 1                 ; bad message
ok_M:
;
; verify the checksum
;
        inc     bx                      ; calculate the checksum
        cmp     bx, _len_message
        jne     loopRCC
        mov     al, BCC
        cmp     al, 0                   ; check the check sum and
                                        ; good message
        jne     M1_reset
```

```
;
; set up the packet type to indicate that the acknowledgment
; response should be positive.
;
        mov     _type_message, 06h ; reply ACK
M_reset:
;
; Replace the setup destination address by receiving Source
; address and keep the
        mov     al, _Destination   ; get the setup
                                   ; destination
        mov     temp2, al
        mov     al, Desttemp       ; replace it by the
                                   ; receiving
        mov     _Destination, al   ; Source address.
        mov     temp, 2            ; flag to indicate the
                                   ; packet
                                   ; is correctly received.
        ret
M1_reset:
;
; set up the packet type to indicate that the acknowledgment
; response should be negative.
;

        mov     _type_message, 02h ; reply NAK
        jmp     M_reset
Rx_packet  ENDP
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                  Rx_char procedure
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  procedure to receive the character from serial port
;
Rx_char  PROC  NEAR
;
; set up the timing window and channel status.
;
        mov     count, 0           ; set the window timing.
        mov     _channel_busyb, 1 ; set the channel to busy.
;
; get the character from serial port
;
        mov     dx, Port_A         ; get the character
        in      al, dx             ;
        mov     temp1, al
        mov     temp, 0            ; flag to indicate that
                                   ; we need to send EOI
                                   ; at the end of interrupt.
```

```
;
; check for the Start Of Text (SOT) character
;
        mov     bx, _index_r
        cmp     bx, 0               ;  SOT ?
        jl      prefix
;
; get only the data and start bi information
;
        mov     al, temp1
        and     al, 1fh
        mov     [r_message + bx], al ; store the receiving
                                     ; data.
        inc     bx                  ; increase the receiving
                                    ; data index.
        mov     _index_r, bx
        cmp     bx, lhead
        je      out1_range          ; call Rx_header
        cmp     bx, lmessage
        je      out2_range          ; call Rx_packet
        ret
out1_range:
        jmp     Rx_header
out2_range:
        jmp     Rx_packet
prefix:
;
; check for the SOT characters
;
        mov     al, temp1
        cmp     al, SOT             ;  SOT ?
        jne     good2
        inc     _index_r            ; increase the receiving
                                    ; data index
retint:
        ret
good2:
        mov     _index_r, -2        ; reset the data index.
        ret
Rx_char  ENDP
_TEXT ENDS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Storage reservation for program variable and pointer
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
_DATA SEGMENT word public 'DATA'
      PUBLIC _count_rate
_count_rate   DW 0          ; Counter to control the arrival
```

```
                                ; rate
        PUBLIC _Destination
_Destination    DB 0            ; Destination address
        PUBLIC _Source
_Source    DB 0                 ; Source address
        PUBLIC _DIF
_DIF    DB 0                    ; flag to indicate that the
                                ; transmitter is enable
        PUBLIC _mask_E
_mask_E    DB    0              ; RS Codec control interrupts
                                ; address
        PUBLIC _index_r
_index_r    DW    -2            ; receiver buffer index
        PUBLIC _index_t
_index_t    DW    -2            ; transmitter buffer index
        PUBLIC _channel_busyb
_channel_busyb    DB    0       ; channel status flag
      PUBLIC _block_length
_block_length   DB  0           ; RS Codec information length
        PUBLIC _txb
_txb    DB  0                   ; Flag indicates the type of the
                                ; packet
        PUBLIC _txb1
_txb1    DB  0                  ;
        PUBLIC _len_message
_len_message DW 0               ; length of the input data
        PUBLIC  _no_block
_no_block       DW  0           ; RS Codec block length
        PUBLIC _code_end
_code_end  DB   0               ; waiting flag
        PUBLIC _block_new
_block_new  DB   0              ; index of the RS Codec input data
        PUBLIC _max_index
_max_index  DW  0               ; index of the RS Codec output
                                ; data
        PUBLIC _type_message
_type_message   DB  0           ; type of the packet
        PUBLIC _out_exit
_out_exit  DB   0               ; flag indicates the terminate
                                ; status
        PUBLIC _max2_index
_max2_index  DW  0              ; index for the transmitter data
com_int    DB     4 DUP(?)      ; interrupt no. for IRQ3
asc_in     DW     0             ; input pointer to ring buffer
asc_out    DW     0             ; output pointer to ring buffer
tx_ascout DW      0             ; output pointer to transmited
buffer
intc_offs DW      4 DUP(?)      ; original content of IRQ5
intc_seg  DW      4 DUP(?)      ; service vector
intoff    DW      4 DUP(?)      ; addresses of the interrupt
                                ; handler
r_message DB    100 DUP(?)      ; temporary receiver data
```

```
BCC         DB    0              ; checksum
bufflen     DB    0              ; buffer length
temp        DW    0             ; temp. storage for buffer length
lhead       DW    31             ; length of header
lmessage    DW    62             ; length of packet
Desttemp    DB    0              ; temp. destination address
temp1       DB    0              ; temporary value
temp2       DB    0              ;      "
temp4       DW    0
temp7       DB    0
count       DW    0
base2       DW    7fffh          ; counter for the execution time
base1       DW    1              ;          "
        PUBLIC _bufferin
_bufferin    DB Buf_len DUP(?)      ; RS Codec input data
        PUBLIC _E_bufferout
_E_bufferout  DB Buf_len DUP(?)    ; RS Encoder output data
        PUBLIC _D_bufferout
_D_bufferout  DB Buf_len DUP(?)    ; RS Decoder output data
        PUBLIC _tx
_tx           DB Buf_len DUP(?)    ; RS232 receiver data
_DATA    ENDS


END
```

APPENDIX C

HEADER FILE

```
/*********************************************************
 *
 *              extern.h - header file
 *
*********************************************************/



/*********************************************************
 *
 *  This header file is used to declaring the external
 *  variables as well as procedures between the C
 *  and the Assembly programs
 *
*********************************************************/

/* external procedures */
extern void Install(char);
extern void UN_Install(void);
extern void Change(void);
extern void infor_E_block(void);
extern void infor_D_block(void);
extern void tx_enable(void);

/* external variables */
extern char bufferin[70], E_bufferout[70], D_bufferout[70],
            tx[70];
extern char mask_E, block_length, code_end, channel_busyb,
            txb, txb1, type_message;
extern int  no_block, max_index, index_t, index_r, count_rate,
            len_message, max2_index;
extern char Destination, Source, out_exit, DIF, block_new;
```