P. 214

PG [redacted]

[redacted] WORK 1994-95

[redacted]

Department of Electrical and Electronics Engineering

Kumaraguru College of Technology

Chinnavedampatty, Coimbatore - 641 006.

P-214

Certificate

This is to certify that the project report entitled

"PG COMMUNICATION USING TELEPHONE LINES"

has been submitted by

Mr._____ Roll No._____

in partial fulfilment of the award of
Bachelor of Engineering
in the Electrical and Electronics Engineering
Branch of the Bharathiar University, Coimbatore
during the academic year 1994-'95.

Dr. K. A. PALANISWAMY, M.E., Ph.D.
Professor and Head
Department of Electrical and Electronics Engineering,
Kumaraguru College of Technology,
Coimbatore-641 006.

_____
Guide

_____
Head of the Department

Certified that the candidate with University Register No. _____

was examined in Project Work Viva - Voce on _____

_____
Internal Examiner

_____
External Examiner

ACKNOWLEDGEMENT

We are greatly thankful to our guide **Mr.S.KUMAR, M.E.,** for his excellent guidance and encouragement in completing this project.

We wish to express our sincere gratitude to **Dr.K.A.PALANISWAMY, B.E., M.Sc (Engg.), Ph.D., M.I.S.T.E., C.Eng (I), F.I.E.,** Professor and Head of the Department of Electrical and Electronics Engineering for his encouragement at all times.

We are immensely thankful to **Dr. S. SUBRAMANIAN, M.Sc (Engg)., Ph.D., SM.IEEE.,** Principal, for his benevolence and blessings throughout the project work.

We also thank all the members of the staff of the Electrical and Electronics Engineering Department and Computer Science and Engineering Department for all their help and encouragement.

SYNOPSIS

The objective of this project is to demonstrate communication between two MS-DOS based IBM PC/XT compatibles over the public telephone network, using a Modem. A single chip FSK Modem AM7910 is used.

The computer acts as a data terminal. Data is taken out through asynchronous communication port COM1 in the serial form. The only external device required to connect the AM7910 to the phone line is a data coupler. Either a direct connect data compler (also known as a data access arrangement) or an acoustic coupler may be used.

Asynchronous mode of transmission and reception has been used. The modulation technique employed is frequency shift keying. Speed of transmission may go upto 1200 bits/second. The parity option can be chosed using software. Four wire full duplex mode or two wire half duplex mode can be used.

The software developed, provides facilities for direct keying in of data and tranfer of files from one computer to another.

The project aims at design and fabrication of the hardware circuit using Modem and development of the software for the communication, using [illegible] language.

CONTENT

III MODEM DESIGN

CHAPTER I

INTRODUCTION

Communication systems that employ already existing channel networks are always preferred. Therefore, data communication via the telephone line is becoming more than just a convenient option. This project work is aimed at developing an efficient communication system which can be used in computer networking.

Most communications will permit signals to be transmitted in a limited band of frequencies. However, any information bearing signal must be such that most, if not all, of its power spectral density lies within this band.

1.1. MODEM

In order to use the telephone networks for data transmission, we generally need to convert our digital signals into a form that will go smoothly along the telephone line and will, at the other end, convert that signal back into digital form for use in a computer. The modem solely serves this purpose.

Data communication is the common factor to distributed data processing on line systems, teleprocessing and terminal based systems.

Modems are generally characteristised by speed and modulation techniques. Low speed modems are implemented using frequency shift keying (FSK) modulation. Medium speed modem (1200 to 4800 bps) are implemented using phase shift keying or Quadrature Amplitude modulation (QAM). High speed modems (9600 bps) are also implemented using Quadrature Amplitude Modulation (QAM).

1.2. SCOPE OF THE PROJECT

In order to transmit any data, it is important to encode it to facilitate easy and better transmission. This is done in the form of 1) Amplitude Shift Keying

2) Frequency shift keying

3) Phase shift keying and

4) Quadrature Amplitude Modulation.

These are described in detail in the chapter 2.

Chapter 3 deals with the details of the design of the MODEM circuit.

The software for the communication has been developed for this project and has been explained in detailed in the fourth chapter.

The hardware and software part of the project have been integrated together. Its installation and testing are described in chapter 5. Conclusion drawn out of this project are given in chapter 6.

CHAPTER II

SYSTEM DESCRIPTION

2.1 Introduction

Normally, a local computer terminal is used to transmit and receive data from a remote terminal using telephone lines as the transmission path. A UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART) can be used to control the digital data flow between the terminal and the modem. After the digital data from the computer is conditioned by the communications controller, the modem operates on the conversion of the digital data to an analog carrier. The demodulation process is also controlled by the modem. The equipment connected to the telephone network must be protected from the external environment.

In addition, the network must be protected from improper connection of equipment. These tasks are assigned to a device known as a Data Access Arrangement (DAA). The system as a whole is shown in Fig 2.1

2.2. Modes of Encoding

The modes of information modulation are discussed here

2.2.1. Amplitude Shift Keying (ASK)

The amplitude of the carrier signal changes with the logical level. Zero amplitude may correspond to logical "0" and a high amplitude may correspond to a logical "1". This is shown in the Fig 2.2.

2.2.2 Frequency Shift Keying (FSK)

Frequency shift keying is a modulsition technique which encodes one bit of the serial data stream per baud. A logical "one" in the bit stream places a mark frequency (FM) on the phone line. A logic "zero" places a space frequency (FS) on the line. As the bit stream switches between logical one and zero, the analog signal on the line modulates between FM and FS as shown in the figure. The modulation process generates energy over a broad spectrum. This spectrum depends on the sequence of bits in the serial data stream.

Since it encodes only one bit per baud, the FSK uses approximately 1 Hz of bandwidth for each bit per second of data. Though it is the least efficient method of

typically requires the least amount of hardware for its implementation.

2.2.3 Phase Shift Keying (PSK)

Phase shift keying is a modulation technique which encodes more than one bit of the serial data stream into a modulation symbol. Sequential bits in the data stream are grouped into pairs or triplets. When grouped into pairs, for example, a two bit code (dibit) is formed which selects one of four phase shifts to be applied to a carrier on the phone line as shown in fig 2.4. A three bit code (tribit) selects one of the eight phase shifts. Since spectrum usage is determined by the symbol or baud rate, encoding more bits per symbol allows a higher bit rate for a given band width.

The demodulation of the PSK from the phase shifted carrier into two or three bit codes requires more sophisticated hardware than that required by the FSK demodulation.

2.2.4. Quadrature Amplitude Modulation (QAM)

The technique of quadrature amplitude modulation encodes multiple data bits into a modulation symbol by assigning

phase shifting the carrier on the line, QAM modulates the amplitude. Typically, four sequential bits are encoded into a constellation or group as shown. With this type of encoding, a single channel at 9600 bps can be provided on the line. This type of modem is the most complex and also the costliest of all the modems to implement.

2.3 Asynchronous and Synchronous Transmission.

The serial data stream enters and exits a modem either synchronously or asynchronously. An asynchronous data stream may have a symbol rate or baud or bit rate that is from zero to the maximum permitted by the modulation technique. The bit rate is determined by the user and there is no separate clock signal to qualify the bits in the data stream. **FSK modems are asynchronous.**

A synchronous data stream has a fixed bit rate determined by the modulation technique. Timing clocks are required to qualify the transmitted and received data streams. PSK and QAM modems are synchronous modems.

2.4. Telephone Network Bandwidth Usage

Simplex transmission is the transmission of data in only one direction during a given call. This is typically limited to 4 wire lines.

Half duplex transmission is said to be used when only one modem transmits and the other receives at any instant. Most of the available bandwidth can be used by this single transmission band. The process of reverting the direction of transmission is called " line turn around" This is required if data must be transferred in both directions during a call. Half duplex over two wire lines is used at bit rates from 1200 bps to 9600 bps.

Full duplex transmission is simultaneous, bidirectional data transmission. Each modem can transmit and receive at the same time. Available transmission bandwidth is divided into 2 channels via frequency division multiplexing (FDM) or the entire bandwidth is used for each channel via echo cancellation. Low to medium data rates are possible over two wire lines. **An advantage of full duplex operation is that no line turn around is required.**

2.5. MODEM CIRCUIT DESCRIPTION

Chip AM7910 is designed complete with transmitter, receiver and interface control and timing logic. The chip operates at 300, 600 or 1200 bps asynchronous data rates. Five pin-programmable mode control lines (MC0 to MC4 select the desired modem configuration

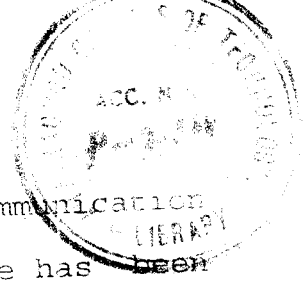
The digital signal processing [illegible] help the chip perform its major [illegible] demodulation and filtering. Based [illegible] chip furnishes TTL-level RS-232 [illegible] signal.

Attaching external parts, the AM7910 [illegible] into a complete stand-alone modem as shown in [illegible] OPAMP and associated resistors form a diplexer [illegible] directly connecting the chip to an external [illegible] interface. Level-conversion circuits change [illegible] voltage levels to RS-232C levels. [illegible] capacitors and switches permit [illegible] onboard functions to suit application needs.

2.6. COMMUNICATION SOFTWARE

The communication software required [illegible] in MS-DOS environment suited to the IBM PC/XT [illegible] Since the IBM PC/XT uses 8088 processor, the [illegible] been developed in the 8088 assembly language.

The IBM PC/XT has been provided with [illegible] communication adapters called as the serial port [illegible] named as COM1 and COM2. This adapter contains [illegible] Universal Asynchronous Receiver Transmitter [illegible] responsible for transmission [illegible]



data from/to the computer through the serial communication line interface RS-232. The communication software has been designed to program the serial port directly. The communication software can work both in terminal emulator mode and file transfer mode. In the terminal emulator mode, computer acts as a dumb CRT terminal. Characters typed on the keyboard are sent out through the output port to the modem and characters coming to the PC on the input port are displayed on CRT. In file transfer mode, a file can be transmitted from one PC to another.

SPECIFICATIONS OF THE SYSTEM

Modem chip used	: AM 7910 world chip FSK modem
Speed of transmission	: 1200 bps
Type of modulation	: FSK
Parity	: Can be none, odd or even
Number of channels and mode	: 4 wire full-duplex or 2 wire half duplex
Number of data bits	: 7 or 8
Number of stop bits	: 1 or 2
Bandwidth	: Phone system bandwidth (3100Hz)

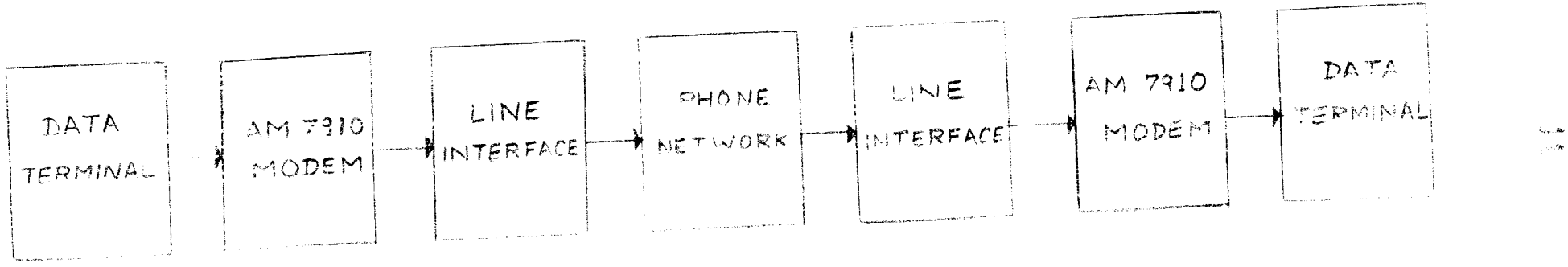
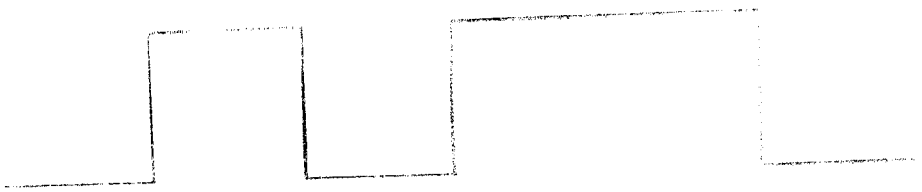


FIG 3-1 SYSTEM BLOCK DIAGRAM



BINARY DATA

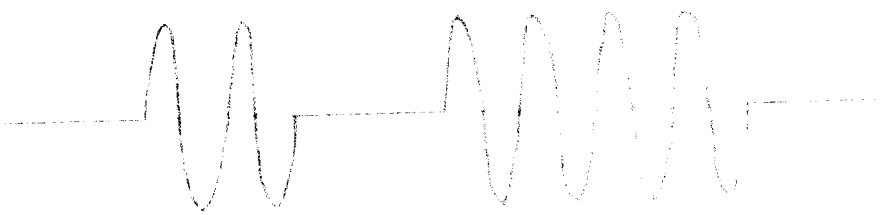


FIG 2·2 AMPLITUDE SHIFT KEYING

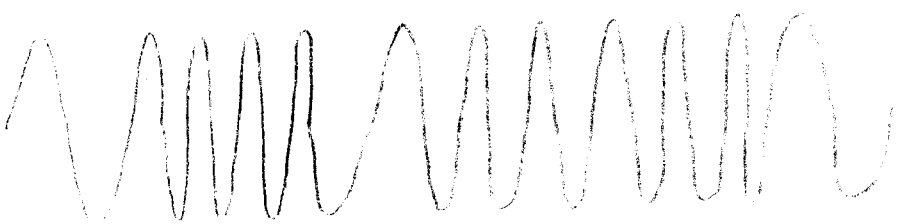


FIG 2·3 FREQUENCY SHIFT KEYING

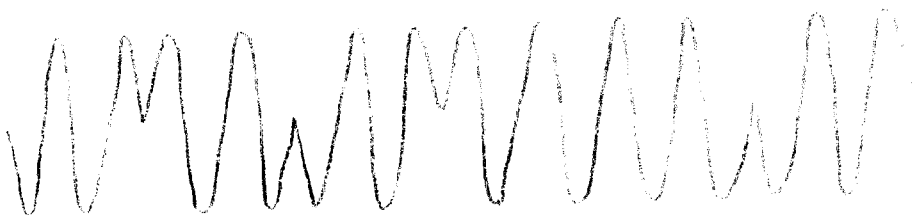


FIG 2·4 PHASE SHIFT KEYING

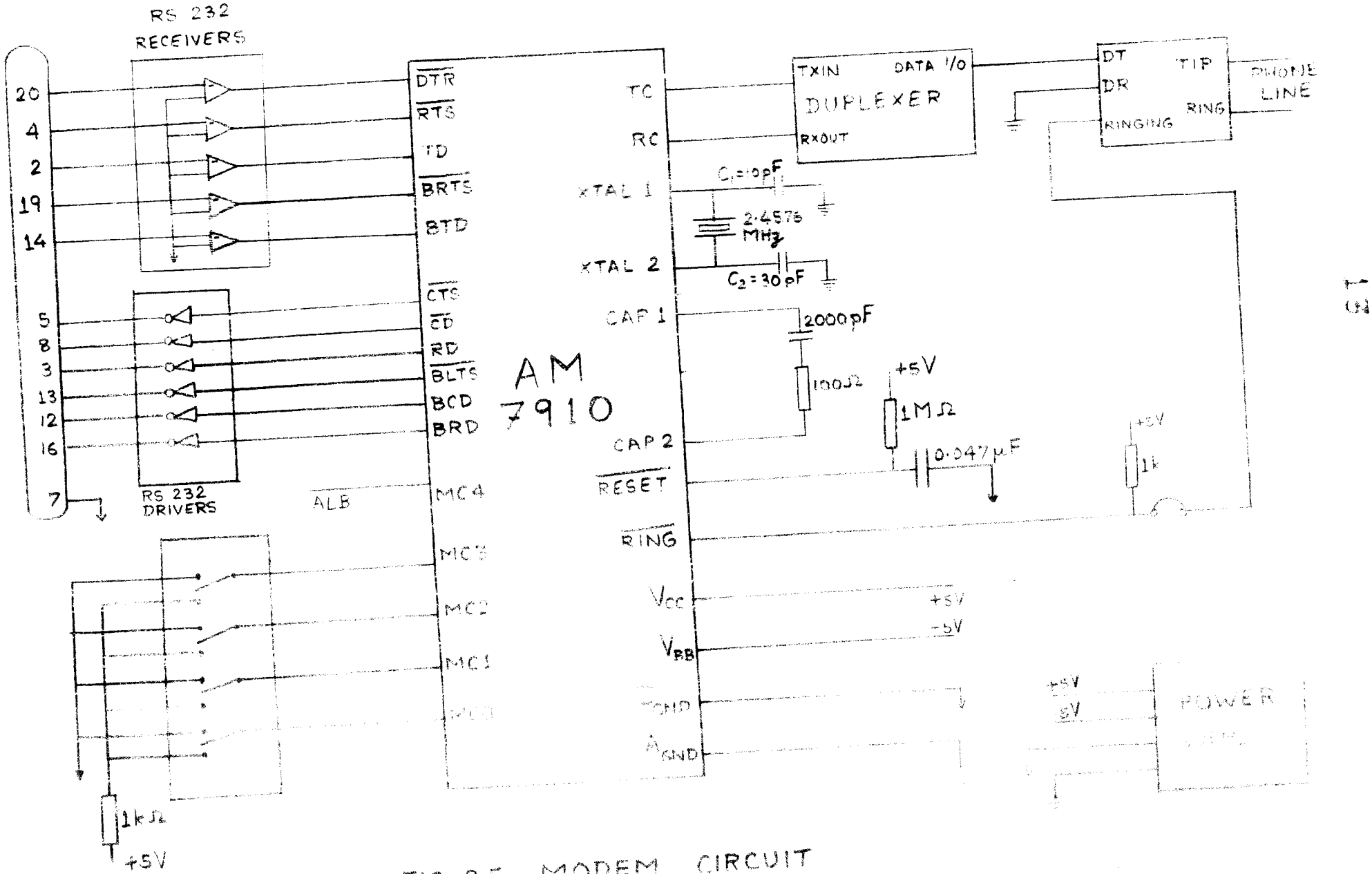


FIG 2.5 MODEM CIRCUIT

CHAPTER III

MODEM DESIGN

3.1. Introduction

This chapter deals with the hardware design of the project. The actual circuit which was fabricated has been designed infour separate modules. The design of these modules are explained in individual sections of this chapter. In the first module, the various external connections required by AM7910 for its operation such as clock, Reset etc are explained. The next section deals with the design of the Data Terminal Equipment (DTE) interface module. This module consists of the modem status indicator circuits and the line receivers/drivers to provide RS232C-TTL conversions. The design of the phone line interface module consisting of the buffer amplifiers and Data Access Arrangement (DAA) is covered in the next section. the power supply module has been designed to produce the various voltages required by the different IC's. The last section of this chapter deals with the design of the PCB required to fabricate the modem circuitry. The overall circuit diagram incorporating all the modules has been given in Chapter V.

3.2 AM7910 Device Details

In this section, external connections to the AM7910 are designed.

3.2.1. Clock

The AM7910 employs digital signal processing to perform both modulation and demodulation operations. Since the performance of the digital signal processing circuit is directly dependent on the sampling (clock) frequency, the exact clock frequency is a critical parameter. Therfore, a clock value of 2.4576 MHz with a tolerance of 0.01 % is required for proper operation of AM7910.

The generation of the AM7910 clock is accomplished by the connection of a crystal to pins XTAL1 & XTAL2. The characteristics of the crystal required are : Type : Parallel resonant, fundamental mode.

Frequency : 2.4567 MHz

Frequency tolerance : + 100 ppm (+ 0.01%)

Frequency stability : + 100 ppm (+ 0.01%)

Maximum shunt capacitance (Co) : 7 pf

Minimum drive level : 1.0 MW

Maximum resonant resistance : 300 ohms

Load capacitance : Shown in fig 3.1

A 32pf load capacitance crystal is used and hence capacitance required are 30pf and 15pf. The standard value of 33pf is used for one of the capacitors & a value of 22pf is used for the other capacitor.

3.2.2 Reset

A RESET pin is provided on the AM7910 which halts the processing of the transmitter and receive inputs. There are critical parameters of voltage and timing for the RESET pin. Timing between the power supplies and RESET should be as follows: Both supplies will initially rise from 0 to 3.5 V in magnitude. After both supplies have reached 3.5 V in magnitude for atleast 1.0 micro seconds an active low pulse of atleast 406ns duration must be applied to RESET. Exact timing and levels of voltages applied to RESET are as shown in fig 3.2. A circuit which automatically resets the modem when power is applied is shown in fig.3.3. The time constant RC = 0.470 sec which is high enough to hold the RESET pin in the low state for a time greater than TRL = 406 ns.

The RESET pin is also provided connection externally which is used to reset the modem while the power is ON.

3.2.3 External A/D Converter Connections

Pins CAP1 and CAP2 of the AM7910 are provided to allow exact selection of a RC time constant which is used by the analog-digital converter of the AM7910 receiver. The nominal values of 100 for R and 2000 pf for C will provide the optimal A/D converter performance and in turn will reduce the distortion of the receiver.

3.2.4. Ground connections

The AM 7910 has two ground pins : AGND and DGND. AGND is the analog ground pin. It has very low impedance connection, which will increase the noise immunity of the receiver, analog to digital converter and transmitter, digital to analog converter when properly connected. AGND should be connected to a low inpedance ground bus near the power supply circuitry. DGND on the other hand, is the digital ground pin which is more tolerant of power supply noise and ripple. DGND is a higher impedance connection because the digital circuitry of the AM 7910 is less susceptible to noise than the analog circuitry.

3.2.5 Power Supply Ripple

The AM 7910 has symmetric positive and negative power supplies, (+5V and -5V) symmetric supplies are required to provide power to the analog sections of the AM 7910. The A/D converter and D/A converter are sensitive to ripple on either power supply. Ripple should be kept to less than 50mV, particularly in the band from 200Hz to 30KHz. In between Vcc and DGND, Vbb and AGND capacitors of 0.1 micro farads is connected close to the device to provide proper decoupling.

3.2.6 DC Carrier Offsets.

The DC offsets of the TC (Transmit Carrier) output of the AM 7910 can vary as much as + 200 mV depending on power supply values and environmental conditions. In constrast, the Receive Carrier (RC) input of the AM 7910 will tolerate only about + 30mV of DC input offset. A blocking capacitor is used to eliminate the DC offset of the TC output before it is placed on the telephone network. The value of this capacitor should be 2.2 micro farads or greater so that frequencies above 60Hz are allowed to pass through the DAA unattenuated

3.2.7. Mode Selection Switches

The AM 7910 can be configured in any one of the 32 possible configurations according to the Bell or CCITT specifications as shown in Appendix. Only 19 of these modes are actually available to the user. The modes are selected by means of five input pins MC0-MC4. These 5 pins are connected to a Dual-in Package (DIP) switch through a resistance pad of 1K as shown in fig.3.4. The DIP switches are used to either ground or connect +5 supply to the 5 mode selection pins. By setting different DIP switch positions, different speeds of operation such as 300 bps, 600 bps and 1200 bps can be obtained.

3.3. Data Terminal Equipment Interface

The AM 7910 can interface to either parallel or serial I/O ports of standard data terminal equipments with minimal external components. Connection to an intelligent terminal or serial computer port is accomplished without extra computer interface components. A universal Asynchronous Receiver Transmitter (UART) is required to connect the AM 7910 modem to a parallel micro computer bus. The UART handles the control/status of the modem to the micro computer as well the parallel-to-serial conversion of data sent over the bus.

The Data Terminal Equipment used is the IBM PC-XT. The serial ports of it are COM1 and COM2 . The RS232C standard level output is obtained at the 9 pin connector attached to the Asynchronous Communication Adapter. This adapter card contains the UART chip IN8250, which is responsible for the parallel/serial conversion.

3.3.1. Transmit Data Connection

The connection of the Transmit Data (TD) pin of the AM 7910 is shown in fig.3.5.

Since the AM 7910 interfaces only to TTL level devices RS232C line drivers and receivers are required for connection to devices accepting standard RS232C/V.24 voltage levels. The MC1489 is the RS232 line receiver which converts the RS232 input from the serial port to TTL level, which can be connected to TD pin of AM 7910. The LED1 is a healthy condition indicator, indicating that the data is being transmitted from the computer to the remote terminal. The LED is off when the input at TD is low.

3.3.2. Receive Data Connection

Fig.3.6 shows the connection to the Receive Data (RD) pin of the AM 7910.

The MC 1488 is a RS 232 line driver converting the TTL level of Receive Data output of AM 7910 to the RS 232 level required by the serial port. The LED2 is an indicator that the data is being received by the computer. When the RD output is low the LED2 is OFF.

3.3.3. Carrier Detection

AM 7910 has a pin named Carrier Detect (CD). A low on this output indicates that a valid carrier signal is present at the receiver and has been present for atleast a time tCDON, where tCDON, depends on the selected modem configuration. A HIGH on this output signifies that no valid carrier is being received.

Fig 3.7 shows the Carrier Detect circuit diagram. When the CD pin is low, indicating that the carrier is present, the LED3 becomes ON thus indicating that the carrier is present. When the carrier is not present, CD output is high and the LED3 is OFF, thus indicating that there is no carrier signal being received in the RC input.

3.3.4. Handshake Signals

These signals are the signals exchanged prior to data transfer between the Data Terminal equipment and the modem, so that the data transfer takes place at the right instants.

3.3.4.1 Data Terminal Ready

A low level on this input indicates that the data terminal desires to send and/or receive data via modem. A high level disables all TTL I/O pins and the internal logic.

Fig 3.8 shows the DTR pin connection. The 74121 is a notriggerable monostable circuit. The logic diagram and truth table of 74121 is shown in fig 3.9.

The input to the 74121 are A1, A2 and B. The trigger input to the monostable appears at the output of the AND gate. A logic equation for the trigger input is

$$T = (A1 + A2) B Q$$

If either A1 or A2 or both are held low, a positive transition at B will trigger the circuit. When triggered, the 74121 produces output pulse at Q whose width is set according to the values of the timing resistor R and capacitor C as

$$t = 0.69 RC.$$

In the circuit connected to DTR, R = 1K , C = 10

t = 6.9 ms. The width of the triggered pulse output is 6.9 ms. During this time the DTR is switched off and then switched on after 6.9 ms.

This switching is done, as soon as power supply is given to the circuit. The delay between the instant of giving the power supply and the occurence of the trigger is timed by the RC combination connected to the B input of 74121. The value of time constant chosed is 0.1 sec. Therefore approximately 0.1 s after power is supplied, the DTR is set high. After 6.9 ms, DTR is reset i.e., switched ON. Whenever DTR is turned to the OFF state from an ON condtion, each state machine and external signals return to the initial conditions within 25 microseconds. After DTR is turned ON, the AM 7910 becomes operational as a modem.

3.3.4.2. Request to Send (RTS)

A low level on this input instructs the modem to enter transmit mode. This input must remain low for the duration of data transmission. A high level on this input turns off the transmitter. Fig 3.10 shows the RTS pin connection. The RTS is a handshake signal output from the DTE (computer) which is pin no.7 of the DB-9 RS232 connector and pin 4 of the DB25 RS232 connector. This is connected to the RTS input of the modem through the RS232.

3.4. AM 7910 Phone Line Interface.

There are two different ways of interfacing modems to the phone line. One of these is to use a direct connection device called a Data Acces Arrangement (DAA).

A DAA is a device which is physically connected to the phone line through an approved connector. The second method uses an acoustic coupler.

3.4.1. Acoustic Coupler

An acoustic coupler uses the existing telephone handset and is not directly connected to the phone line. Acoustic couplers suffer from a number of disadvantages relative to DAA's.

1. They require a standard telephone handset.
2. They are not easily acceptable to automatic calling and answering applications.
3. They will not work well in noisy environment because of the nature of the way the signal is coupled to the home line.
4. They will work only with lower speed modem-typically they are not used above 1200 band.
5. They are much larger than DAA's because they must accomodate the telephone handset.

3.4.2 Direct Network Connection

Direct connection to the telephone network means that the modem is connected to the line through an approved interface device (DAA). The requirements of the DAA are: AC and DC impedance, lightning strike surviveability, billing delay and maximum transmit level specifications. Fig 3.11 shows a basic DAA circuit.

The circuit interfaces to the telephone network on the Tip and Ring leads. The relay provides the on-hook/off-hook control for connection or disconnection from the line. The series resistor is for short circuit current limiting. The IN6048 transient protector is a bidirectional zener diode which limits the potential between Tip and Ring to a threshold. The transformer provides the electrical isolation between the locally powered user's equipment and the balanced telephone line. the hybid or duplexor is used to allow the 2 wire telephone line to be connected to the 4 wires of the modem.

3.4.3. Transmit Carrier connection.

Fig 3.12 shows the connection of the TC pin of AM 7910 to the analog out of the modem.

The analog output signal is obtained from the transmit carrier (TC) output of modem through a variable attenator

This is used to match the voltage levels from the modem to the voltage level acceptable to the telephone line. The gain can be varied by adjusting the trimmer potentiometers. The potentiometers are preset to a particular value depending on the line used. The diode combination provides a clamping voltage which minimises the distortion to the data due to the line transients. The op-amp circuit is designed as an inverting voltage amplifier. The closed loop gain is given by

$$ACL = \frac{Rf}{Rs}$$

Where Rf is the feedback resistance

Rs is the source resistance.

The value of Rf and Rs are chosen to the Rf = 10K and Rs = 2K so that the gain can be upto 5. The resistor in the non-inverting input is used to reduce the output offset voltage.

3.4.4 Receive Carrier Connection

The circuit in fig 3.13 is similar to the Transmit carrier circuit but the direction of the data flow is reversed. The received signal is amplified/attenuated with a gain which can be adjusted by means of trimmer potentiometer.

3.5. Power Supply

The modem circuit requires 4 voltage levels. +5V & -5V for AM7910 IC and +12V & -12V for 1488. The power supply required for AM 7910 has to be regulated with stringent requirement of ripples as given in sec.3.2.4. In order to achieve this, three-input voltage regulator ICs namely 7805, 7812, 7905, 7912 are used to provide +5V, +12V, -5V and -12V respectively.

Fig 3.14 shows the power supply circuit. The input to the power supply circuit is +15V DC with tolerable ripple limits. This DC input is regulated by the regulator ICs to produce a dc voltage with almost zero ripple. The capacitor C1=1microFarad connected in parallel to the input lead acts as a bypass capacitor to prevent oscillations within the regulator ICs due to lead inductance. To improve the transient response of the regulated output voltage, a bypass capacitor C2 =10microFarad is used.

3.6. PCB Design

The circuit is fabricated in 2 PCB s. The modem IC and related circuits are fabricated on a main board. The power supply circuits are fabricated in a separte smaller PCB. The output of the power supply PCB is connected to the main PCB by a set of wires. Single sided PCB was chosen because of its simplicity.

The pairs of the 2 PCB's were fabricated because one modem is required at each end of the transmission link. Manual soldering of components was done and connecting wires were soldered. The regulated power supply lines from the power supply PCB are connected to the power supply input lines of the main PCB, by means of connecting wires

The input leads to the modem are :

+15V, -15V, GND, RESET, RTS, Transmit Data and Analog in The output leads from the modem are Receive Data and Analog out.

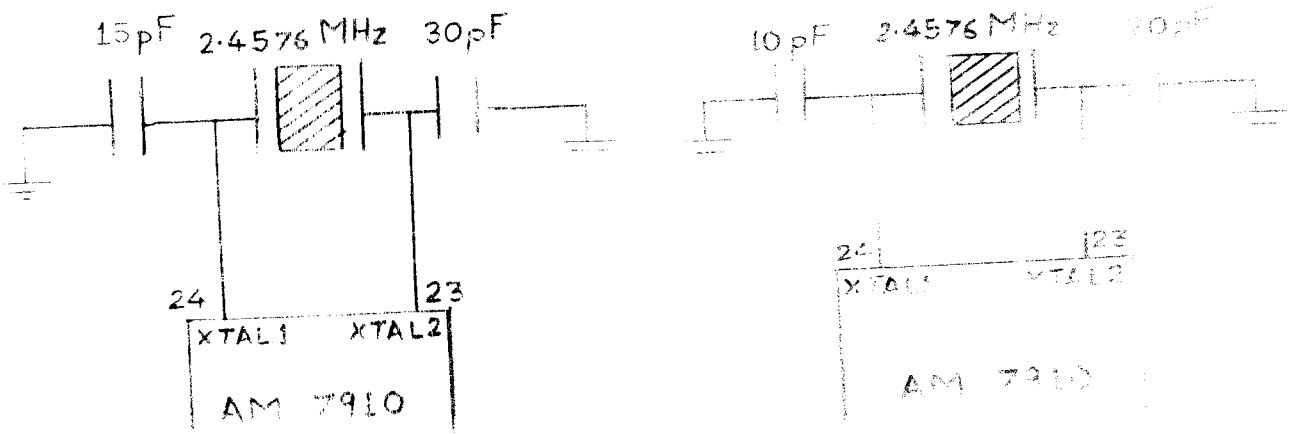


FIG 3·1 AM7910 CRYSTAL LOAD CAPACITORS

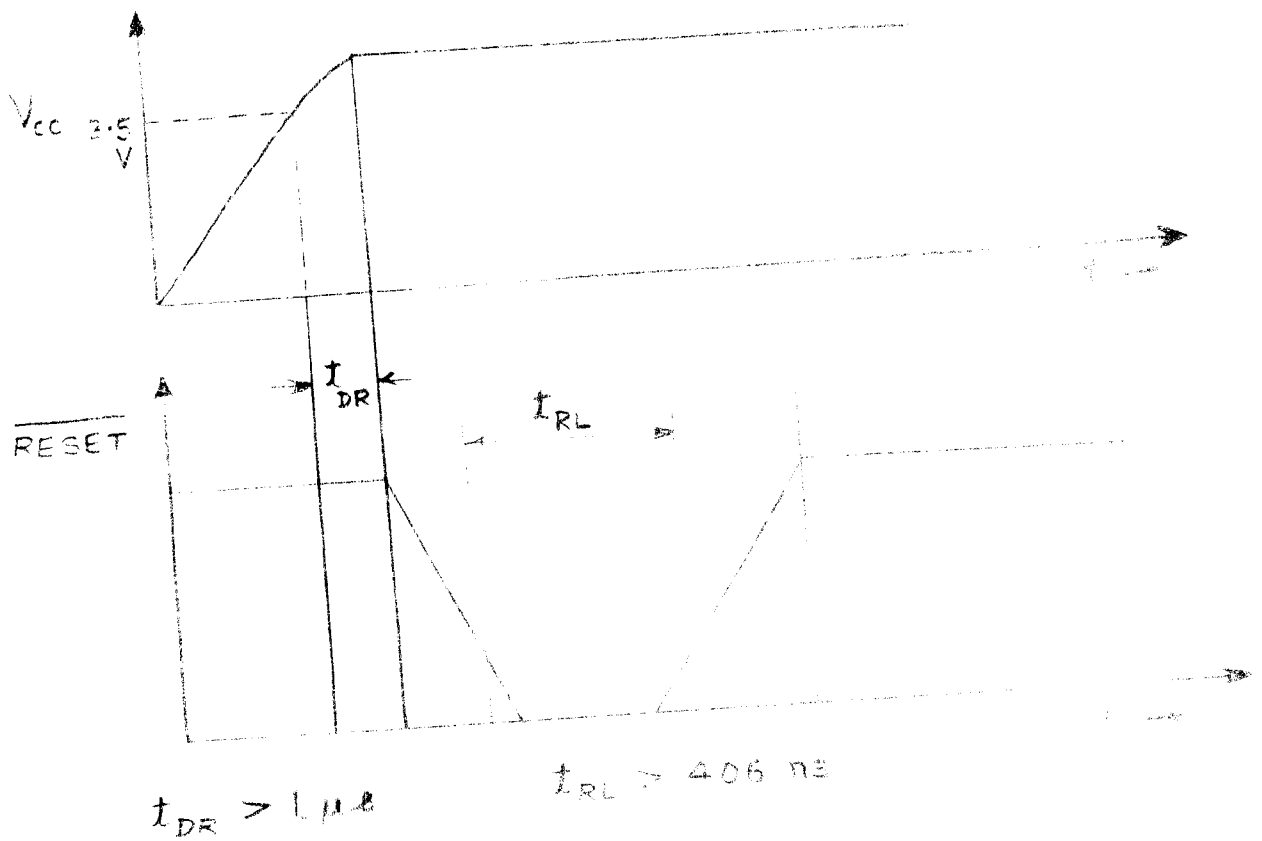


FIG 3·2 RESET TIMING DIAGRAM

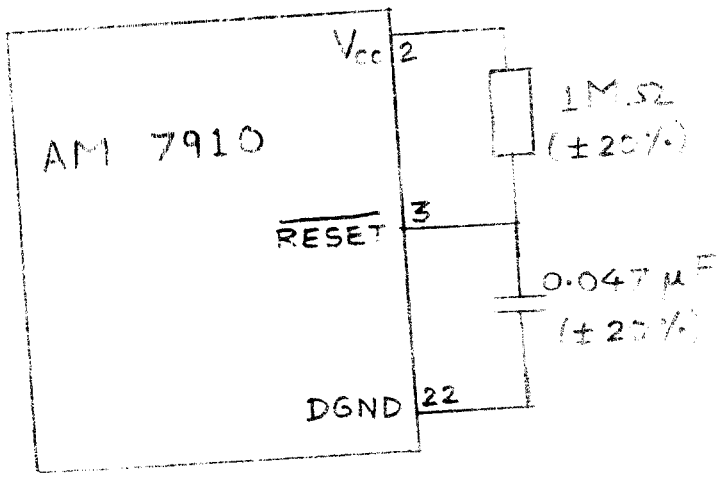


FIG 3.3 AUTOMATIC RESET CIRCUIT

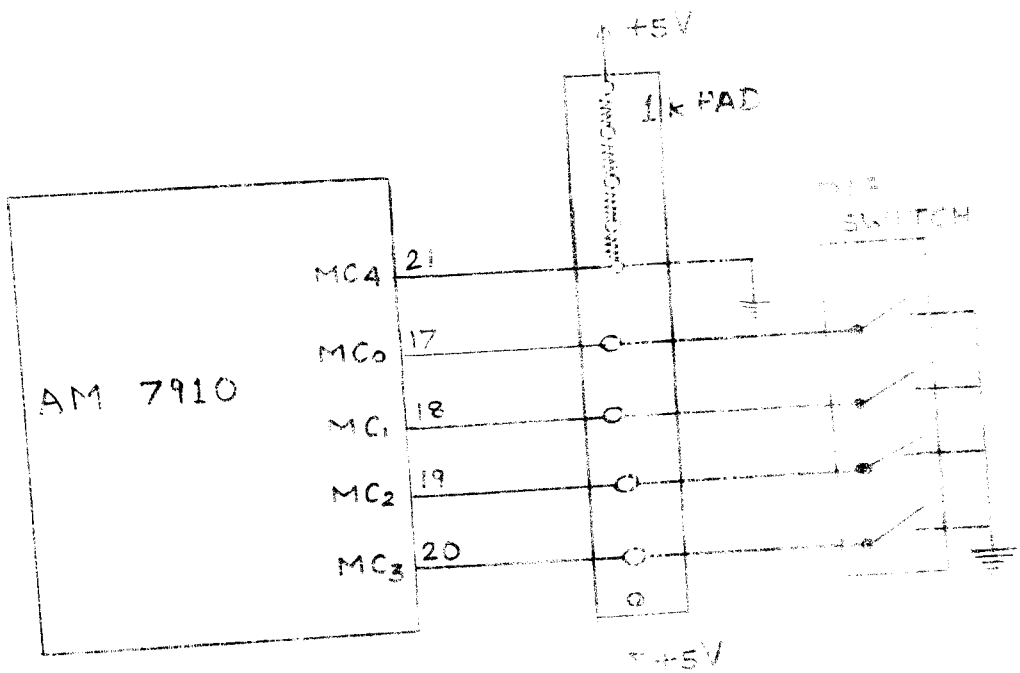


FIG 3.4 MODE SELECTION CIRCUIT

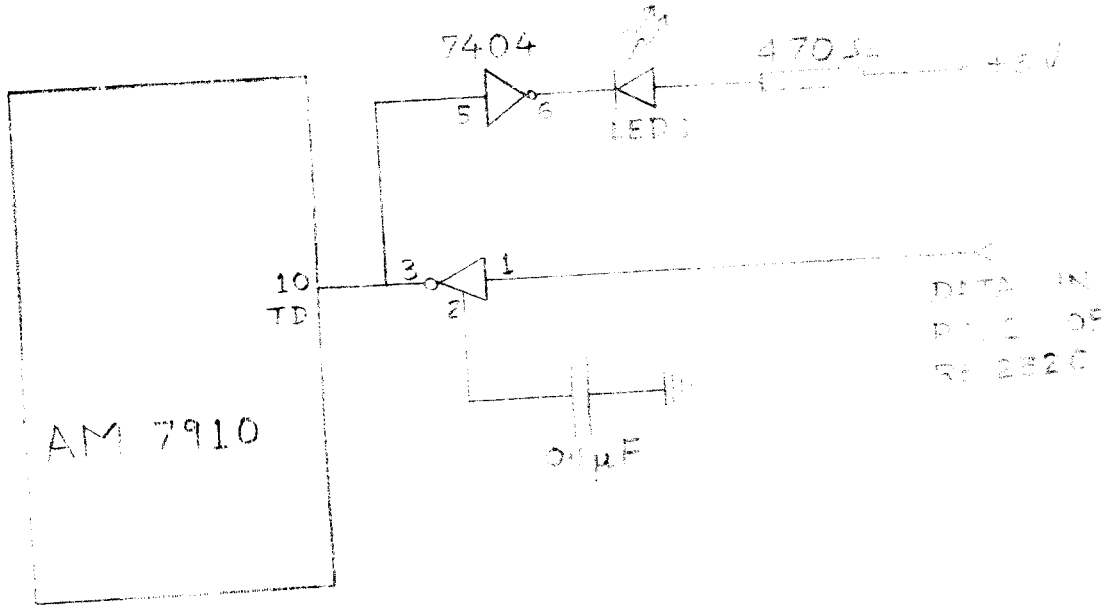


FIG 3.5 TRANSMIT DATA

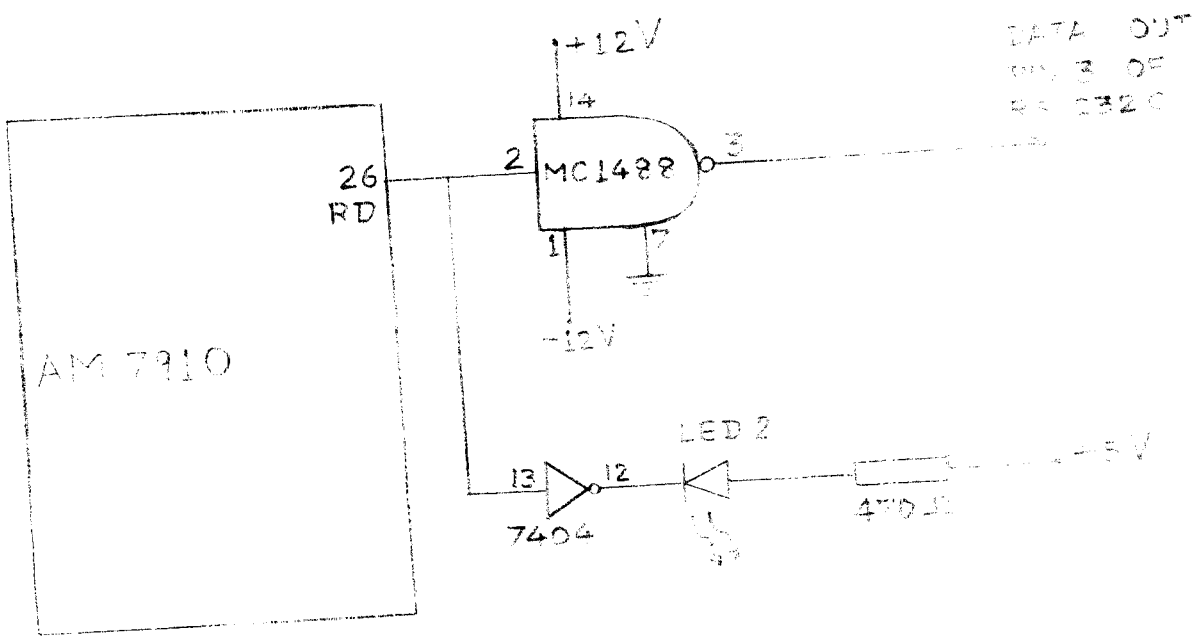


FIG 3.6 RECEIVE DATA

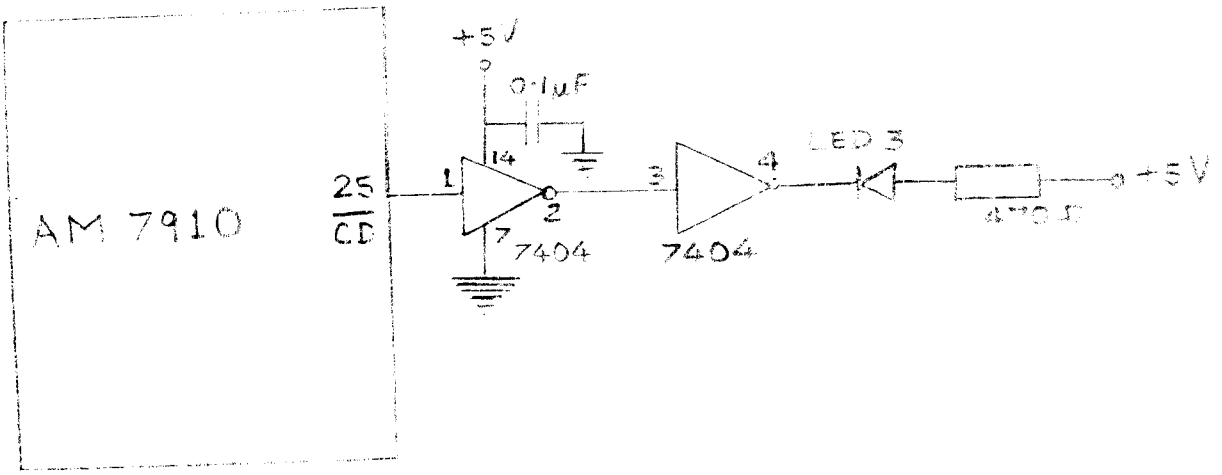


FIG 3.7 CARRIER DETECT CIRCUITRY

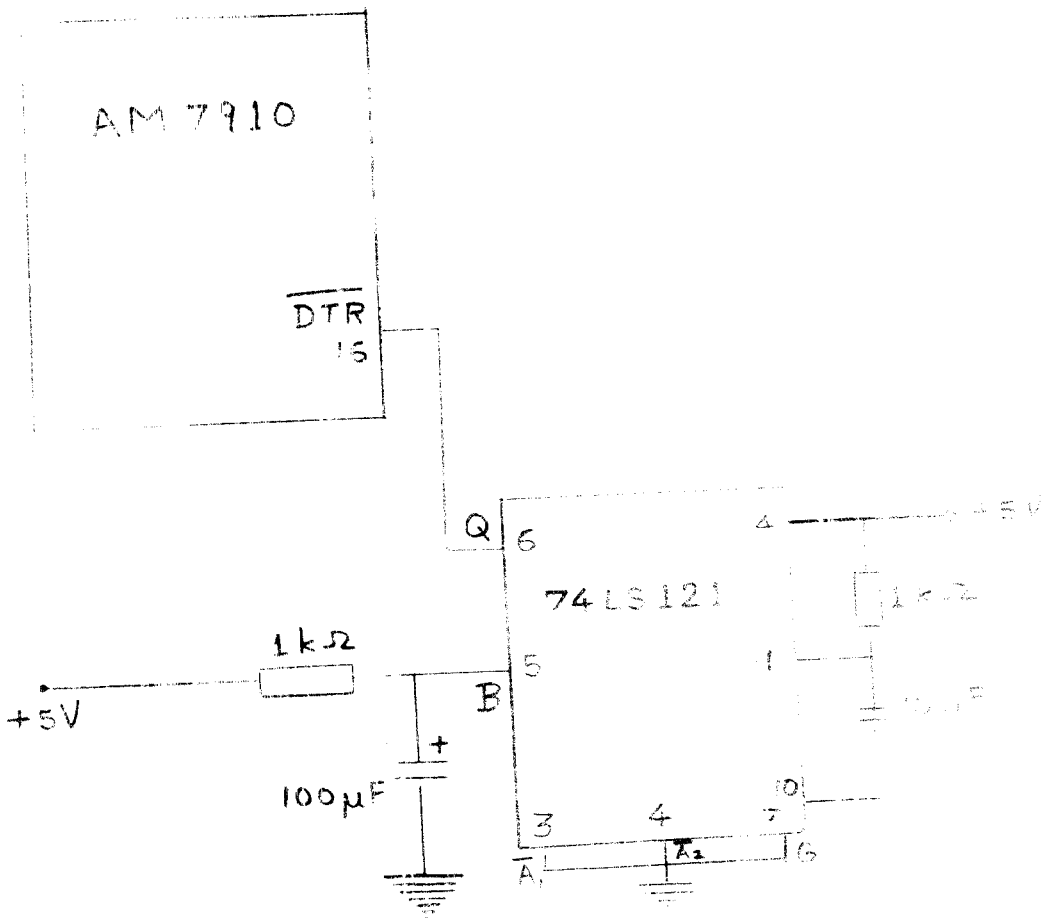


FIG 3.8 DATA TERMINAL READY

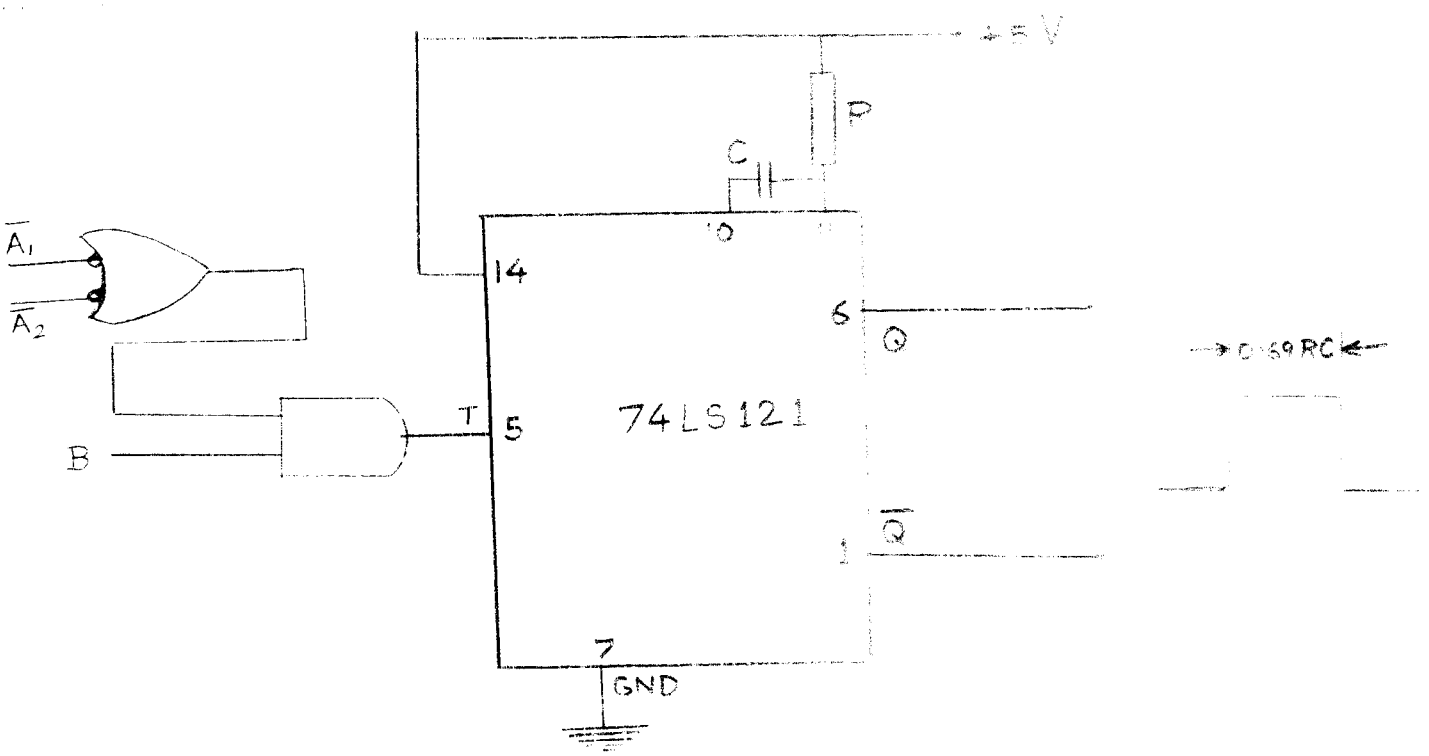


FIG 3.9 (a) LOGIC DIAGRAM OF 74LS121

$\overline{A_1}$	$\overline{A_2}$	B	RESULT
L	x	↑	TRIGGER
x	L	↑	TRIGGER
↓	H	H	TRIGGER
H	↓	H	TRIGGER

L → LOW

H → HIGH

x → DONT CARE

↑ → LOW TO HIGH TRANSIT

↓ → HIGH TO LOW TRANSIT

FIG 3.9 (b) TRUTH TABLE

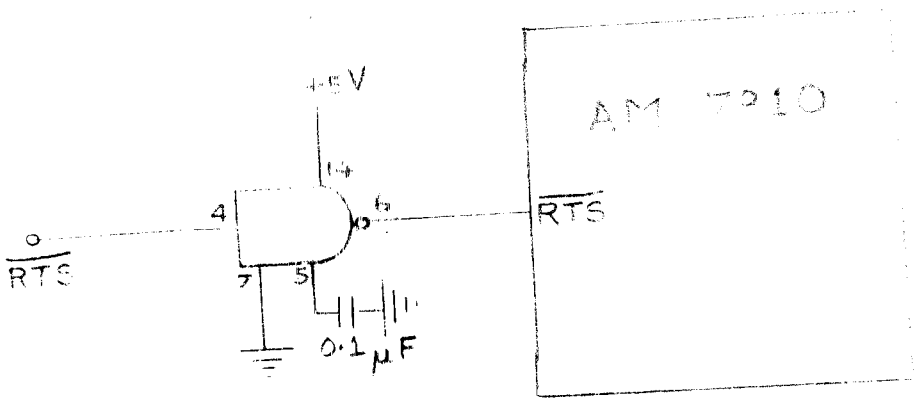


FIG 3.10 REQUEST TO SEND

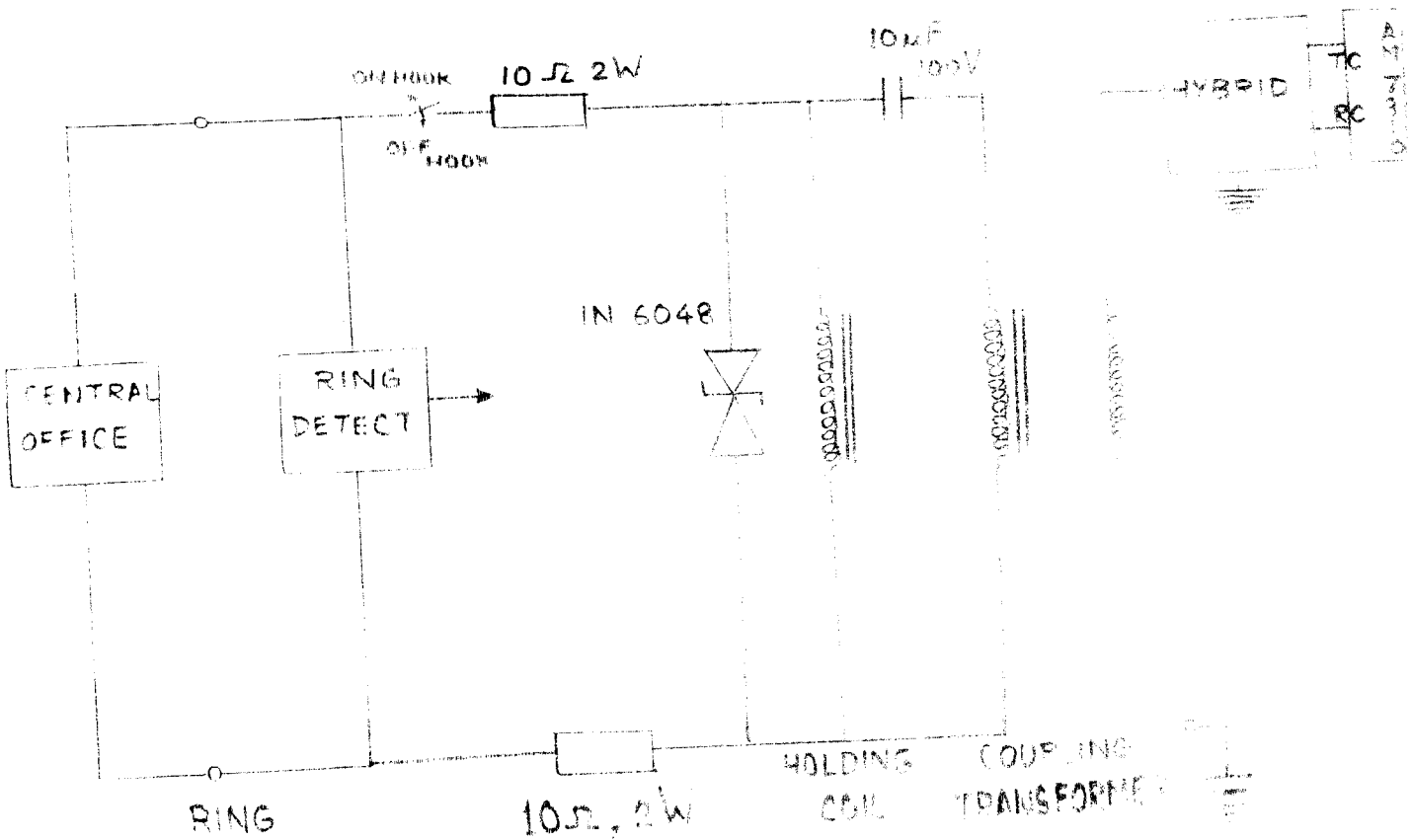


FIG 3.11 DIRECT CONNECT DAA

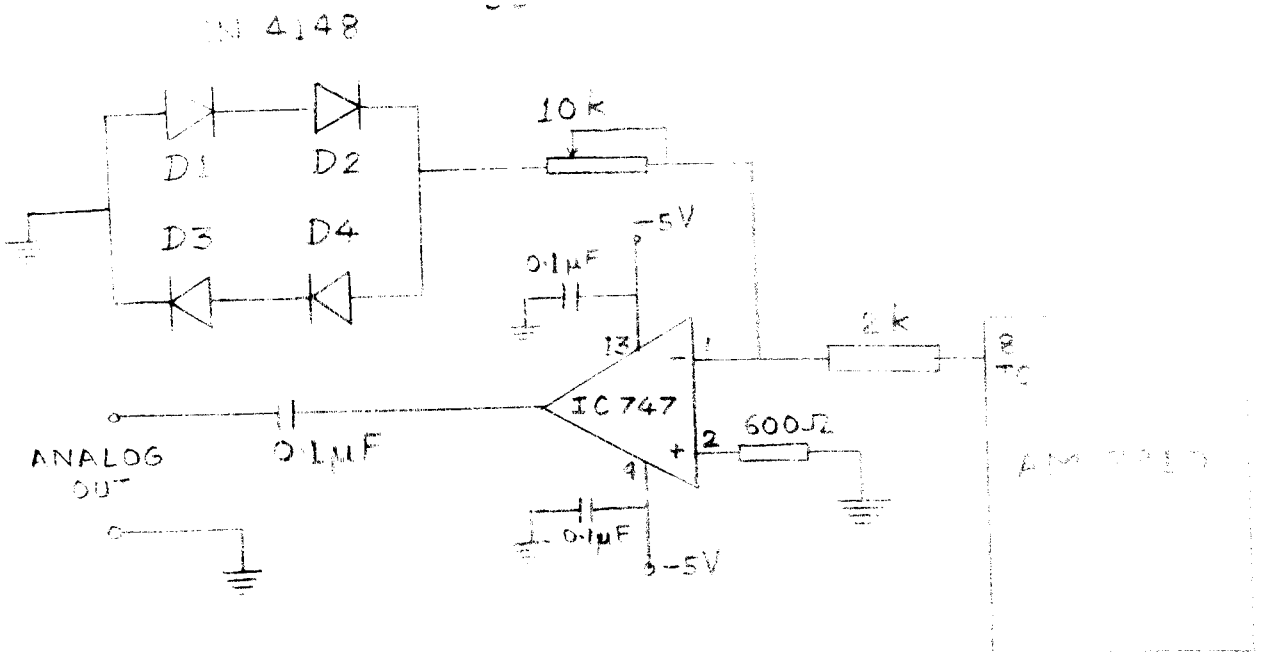


FIG 3.12 TC CONNECTION

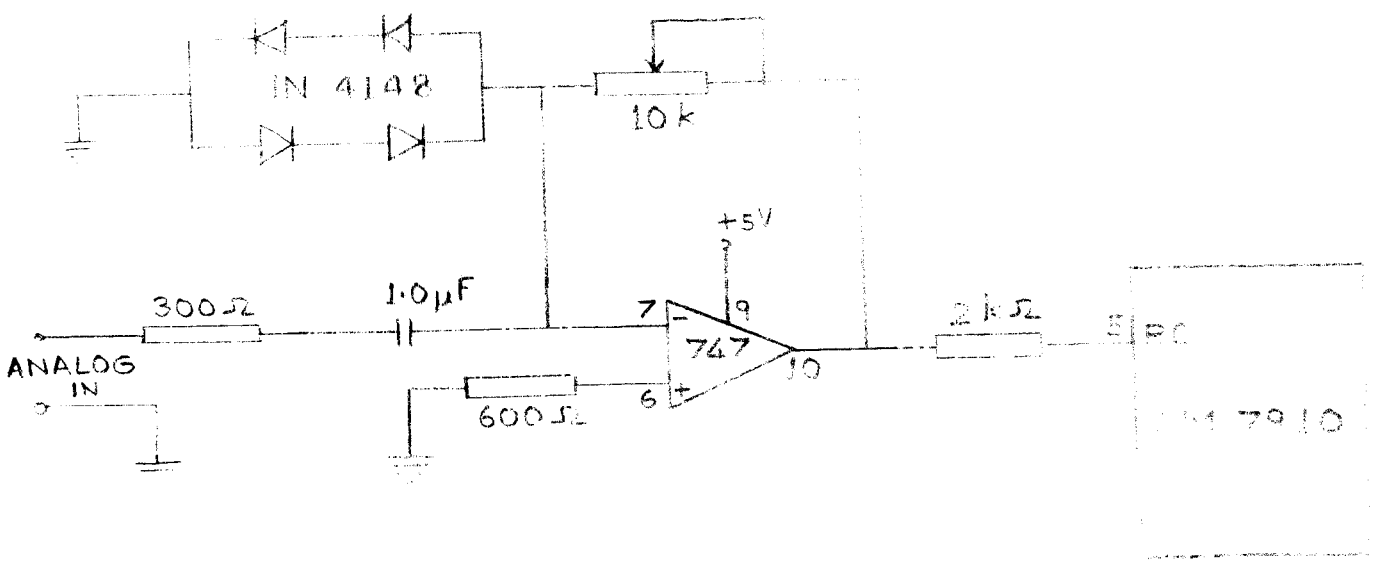
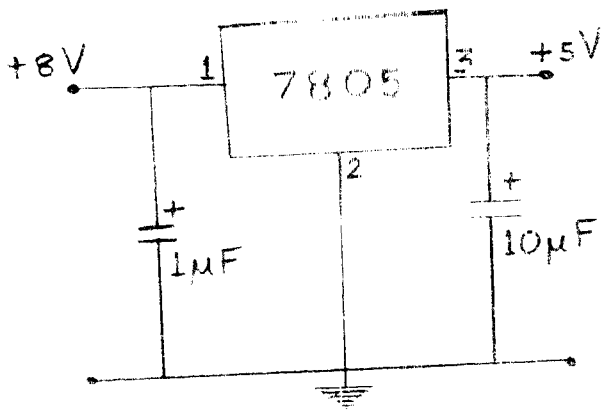
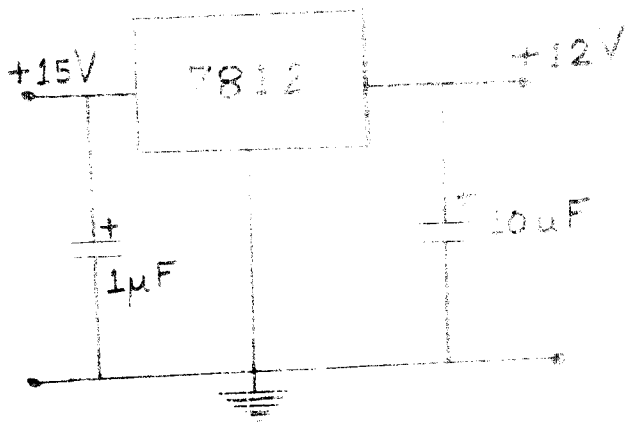


FIG 3.13 RECEIVE CARRIER CONNECTION

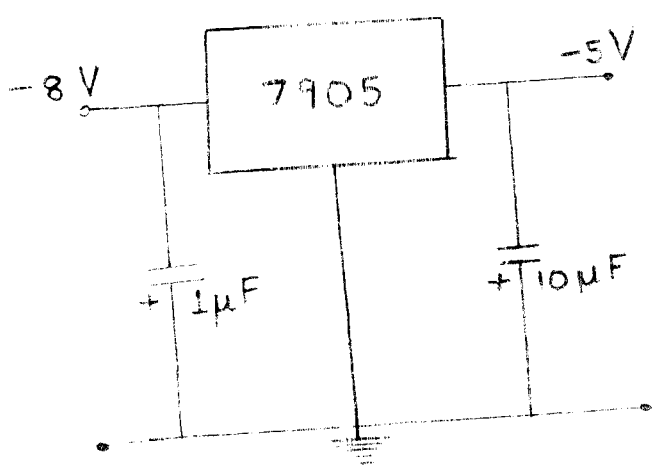


3·14 (a) +5 V SUPPLY



3·14 (b) +12 V SUPPLY

3·14 (c) −5 V SUPPLY



3·14 (d) −12 V SUPPLY

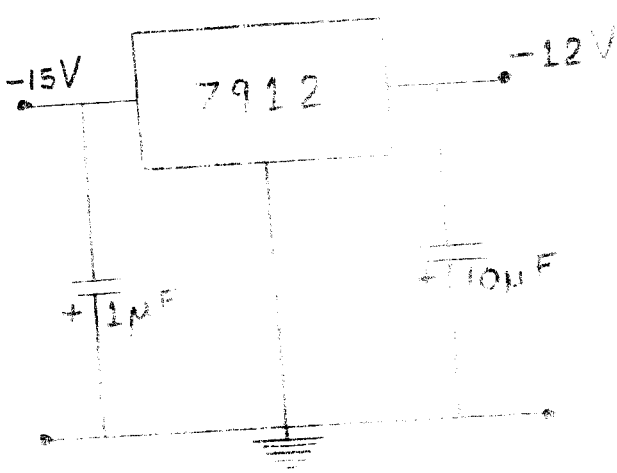


FIG 3·14 POWER SUPPLIES

CHAPTER - IV

COMMUNICATION SOFTWARE DEVELOPMENT

4.1 INTRODUCTION

This chapter deals with the design and development of the communication software. This software has to be run simultaneously on the two computers to be linked The design problem is first defined. The requirements and the facilities that the software should provide, are defined in the next section. Next, the process of development of the algorithm to implement our problem is also depicted

Once the design of the algorithm and the pictorial representation by means of flow charts are completed, the next step in the software development is translation into actual source code. The choice of the coding techniques and means of coding are dealt in detail in the last section. The testing stage of the software has been explained in the next chapter.

4.2. Design Objectives

The software is responsible for providing communication between two computers connected through telephone line. The software must be able to work in 2 modes

4.2.1. Terminal Emulator mode

In this mode, the computer acts as a dumb CRT terminal. Characters typed on the keyboard are sent out on an RS232 line to the modem. Simultaneously, any character that has been received into the PC on an RS232 line are displayed on the CRT. This is useful in communicating in an interactive manner.

4.2.2. File Transfer Mode

In this mode, the computer can send an entire file to a remote computer. This is the most important application of any computer link. This provides a fast method of transferring data in the form of files. The terminal emulator mode is used prior to the file transfer mode for informing the remote terminal about the file transfer.

The user must be able to readily switch over to either of the modes by the press of key. The program has been designed such that the user presses the function key F-2 to **begin file transfer.** After the file transfer is over, the program returns to the terminal emulator mode.

The user should be provided with the option of choosing the following :

1. Full duplex or half duplex mode
2. Baud rate : 300, 600, 1200 baud
3. No. of stop bits : 1 or 2
4. No.of data bits word : 7 or 8
5. Parity : None, Even, odd

The program **should exit to the operating system** when a particular key in pressed. The function key **F-10** is used.

In **full duplex mode,** the **message** that is **typed** is **not displayed on the sender's terminal but displayed on the remote terminals CRT.** However, in the half duplex mode, the message is displayed on both the terminals. If some error occurs, during the file transfer, appropriate error messages are shown and the program returns to the terminal emulator mode.

4.3. Algorithm and flow chart.

There are two strategies for receiving the character from the serial port. They are polling based and interrupt based, algorithms.

4.3.1. Polling algorithm

In this method, the serial port is periodically checked to see if a character is waiting. If it is waiting it is displayed on the screen. Otherwise, the program checks the

keyboard for any character. Fig.4.1 shows the polling algorithm for a terminal emulator mode.

```
INIT COM1
REPEAT
      IF CHARACTER READY IN UART THEN
                  READ CHARACTER
                  SEND TO CRT
      IF KEYPRESSED ON IBM KEYBOARD THEN
                  READ KEY
                  SEND TO SERIAL PORT
      UNTIL FOREVER
```

4.1 Polling Algorithm for terminal emulator

This algorithm, is suitable only for low data rates upto 300 bps. This is because the routine used for displaying the character on CRT takes a finite time before it returns to polling the serial port. This happens at the end of a screen, when the screen is scrolled up. In this time, any character coming from the serial port is missed, if the data rate is greater than 300 bps Therefore the first few characters of the next line after scrolling will be missing. Therefore this method is not used.

4.3.2 Interrupt driven algorithm

This algorithm can be used for speeds upto 9600 bps. The primary problem with the polling method is that data might come in before the program can poll the port again. This results in data loss.

In interrupt-driven communications, a byte of data arriving at the communication port interrupts the CPU so that it can retrieve the byte and put in a buffer. This is done by the interrupt service subroutine, for the serial communications interrupt. The interrupt vector for this interrupt is modified to point to our Interrupt service Routine (ISR). This ISR puts the receive data in a circular buffer. The terminal program can retrieve characters from the buffer at leisure.

Fig 4.2 shows the algorithm for the interrupt driven program.

```
INITIALISE EVERYTHING
    REPEAT
        IF KEYPRESSED THEN
            READ KEY
        IF KEY = EXITKEY THEN
            QUIT
        ELSE IF KEY = FILE TRANSFER KEY THEN
```

DOWNLOAD FILE FROM DISK TO PORT

ELSE SEND CHAR TO SERIAL PORT

IF UART BUFFER HAS CHARACTER THEN

SEND CHARACTER TO CRT

UNTIL QUIT

Fig 4.2 Algorithm for file transfer program

4.3.3 Flow chart

The flow charts for the communication software are given in fig.4.3(a)-4.3(i). Fig.4.3(a) gives the mainline flowchart for the program. It checks for the display to be in text mode. If not, an error message is displayed and the program terminates. Then, the various communication parameters are initialised by the user, as shown in detail by the flow chart of fig.4.3(b). After the initialisation, the interrupt vector for the serial port interrupt for COM1 (INT OCH) is modified so as to point to our **interrupt Service Subroutine titled asc-int**. The Data Terminal Ready output is enabled to indicate that the computer is ready to send data. Request to send control line is not enabled at this stage.

The choice of full duplex or half-duplex during initialisation causes the program to branch to two alternate routines, as shown in fig.4.3(a).

Ful Duplex mode

This mode is chosen by pressing '1' during intialisation [fig4.3(c)]. Since both the terminals communicate simultaneously, RTS of both modems are enabled setting the RTS output of the computer to low. Now the pressing of a key is checked and if a key is pressed it is compared with the exit key and the file transfer key. If it is exit key, control transfers to an exit subroutine [fig.4.3(b)] which disables the interrupts and restores the interrupt vector for the serial port interrupt. If the key pressed is file transfer key (F2) control transfers to the file transmit subroutine. If the key pressed is none of the above keys, the character is written to the serial port.

The program then checks for any character in the buffer, which is filled up by the Interrupt Service Routine (ISR) whenever a character is received at the serial port. If a character is present, it is displayed on screen and control once again checks for a character in buffer. When the buffer is empty, the control transfers to the checking of key pressing.

Half Duplex mode [fig.4.3(e)]

In this mode, one terminal is in transmitting mode, while the other is in receiving mode. Which terminal is in

the transmitting mode depends on who pressed the transmit key (control-5). When the Terminal 1 pressed cntl-5 a code is sent on the serial port which indicates to the remote terminal that it should enter the receive mode. Now the transmit terminal's RTS output is made low, while the receiving terminal's RTS is made high. The terminal 1 continues to transmit until it presses cntrl-z, when it goes into the receive mode. Now it receives messages from terminal 2. The message is displayed on both the terminal's display unit. This transmission (reception) continues until the transmitter presses F10 (To exit) or F2 (To begin file transmit).

File Transmit Module:

The flow chart is shown in fig 4.3h. This module is enered whenever F2 key is pressed. File transfer can be called from both halfduplex and full duplex mode. The transmitting terminals RTS is enabled. The filename is obtained from the user. The file is read into a file buffer. After closing the file, a control signal is sent on the serial port which signals the remote terminal to open a file to receive the transmitted file. The receiver can give any filename at this point. The transmitter waits for a ready singal from the receiver. Then it transmits character by character, displaying the file on both the screens. At

the end of file, an End of File character is transmitted which signals the receiver to close the file.

File Receive Module : (fig 4.3)

This module is executed when the remote terminal has sent a file transmit code through the serial port. On receipt of this code, the user is promoted to enter a filename in which the transmitted file is to be saved. Any path name can be given. The specified filename is created. Now, the receiving terminal sends a ready signal to the transmitter. Now the file received at the serial port is stored in a file buffer, character by character. When the End of File in encountered, the file is written to the disk from the file buffer and the file is closed. A receive acknowledge signal is sent back to the transmitter, to indicate that the file has been received properly.

Fig 4.3j shown the ISR flow chart. The ISR saves the registers and then reads the data from the received data register of the UART (8250), and puts it in the circular buffer in memory. The pointers to the buffer are incremented. The registers are restored to the previous value and an interrupt return is executed. This control is transferred to the state where interrupt had occured.

4.4. Coding

The 8088 assembly language program for the above flow charts is given. The program is created as the file PCLINK.ASM using the Norton Editor.

The program begins with symbol equates. The addrsses of the various registers and the ASCII codes for contrl keys are defined. At entry from MS-DOS, the main routine of the program the procedure called PCLINK initialises the display to be in text mode. Next, the communication parameters are initialised. The parameters are coded into a single byte and stored in memory location " serial-init ". This is used for cinfiguring the COM1 port using ROM BIOS Interrupt 14H.

The program then calls rountine "asc-enb' which takes over the serial port interrupt vector and enable interrupts. The COM 1 port is initialised using INT 14H.

Now, depending on the choice of full duplex or halfduplex, the program branches to different locations. The program enters a loop that reads the keyboard and sends the character out ot the serial port and then reads the serial port and puts the character on the display. If F2 key is pressed, control transfers to the "FILE-TRANSMIT" procedure, while the remote terminal's control transfers to

the "FILE RECEIVE" procedure on pressing F10 key the program terminated after restoring the interrupt vector for the serial port interrupt.

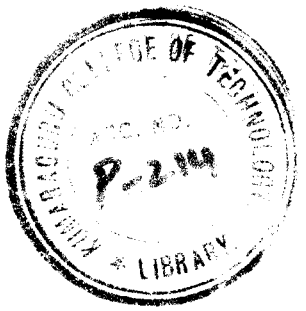
4.4.1 Subroutines used

The program calls many procedures which perform a particular action. The functions of these procedures are as follows:

Procedures	Function
asc-enb	Takes over the serial-port interrupt vector and enables interrupt by writing to the interrupt enable register of INS8250 and the interrupt mark register of 8259A.
asc-dsb	Restores the original state of the serial-port interrupt vector and disables interrupts by writing to the interrupt-mask register of the 8259A.
asc-int	This is the serial-port interrupt Service Routine (ISR) which places the received characters into a ring buffer.
com-stat	Tests whether characters from the serial port are waiting in the ring buffer Zeroflag is set if the character is not waiting.

com-in	Removes characters from the interrupt handler's ring buffer and increements.
com-out	The buffer pointers appropriately returns new character in AL register. Sends the character in AL register to the Serial Port.
cls	Calls the ROM BIOS video driver to clear the screen.
home	Places the cursor in the upper left corner of the screen.
gotoxy	Positions the cursor at the desired position on the display.
PC-out	Write character in AL to the PC's display.
PC-stat	Reads keyboard status Zeroflag = false if character ready Zeroflag = true if nothing waiting.
PC-in	Reads keyboard character and returns the main byte in AL and the scancode in AH.
transmit	Transmits the user specified file to the remote terminal.

freceive	Receives the incoming file from the port into a user specified file.
transmit	Functions as a terminal emulator in the halfduplex mode. Sends keyed in character to port and displays the character in display.
receive	Receives the character from the serial port and displays it.
rtson	Makes the RTS output low enabling the modem to transmit.
rtsoff	Makes the RTS output high disabling the modem from transmitting
delay	Produces a delay after changing the RTS Pin.




```
;       Title:PCLINK Program for file transfer and dumb
        terminal

;                       PCLINK.ASM
stdin   equ 0    ;standard input handle
stdout  equ 1    ;standard input handle
stderr  equ 2    ;standard error handle

cr      equ 0dh    ;ASCII carriage return
lf      equ 0ah    ;line feed
bsp     equ 08h    ;backspace
escape  equ 1bh;escape key
dattr   equ 07h ;display attribute for text mode

bufsiz  equ 4096   ;size of serial port buffer
pic_mask equ 21h ;8259interrupt mask number
pic_eoi  equ 20h   ;8259 EOI port

com_data equ 03f8h ;port assignments for COM1
com_ier  equ 03f9h
com_lcr  equ 03fbh
com_mcr  equ 03fch
com_sts  equ 03fdh
com_int  equ 0ch      ;COM2 interrupt number
int_mask equ 10h      ;IRQ4 mask for 8259

exit_key equ 44h    ;F10
file_key equ 3ch    ;F2
file_code equ 11h     ;DC1
file_ack equ 12h      ;DC2
eof equ 1ah           ;sub
close_ack equ 14h     ;dc4
maxchar equ 32768
transmit_key equ 13h
eot_key equ 1ah

exit_code  equ 15h
lock_key   equ 14h

count1 equ 2
count2 equ0ffffh

_TEXT segment word public'CODE'
        assume cs:_TEXT,ds:_DATA,es:_DATA,ss:S

pclink proc far  ;entry point from MSDOS

mov ax,_DATA
mov ds,ax
mov es,ax
```

```
                        ;initialize display for
                        ;text mode

        mov ah,15
        int 10h
        dec ah
        mov columns,ah

        cmp al,7
        je  talk2
        cmp al,3
        jbe talk2
        mov dx,offset msg1
        mov cx,msg1_len
        jmp talk3

talk2:  mov bh,data             ;clear screen and home cursor
        call cls

        mov dx,offset initial_msg  ;display message
        mov ah,9
        int 21h


        repeat1:mov dx,offset mode;initialize transmission
                mode
                        mov ah,9
                        int 21h
                        mov ah,8
                        int 21h
                        cmp al,'1'
                        je full_duplex
                        cmp al,'2'
                        je half_duplex
                        mov dx,offset entry_error
                        mov ah,09
                        int 21h
                        jmp repeat1
        full_duplex:        mov echo,0h
                            jmp over1

        half_duplex:        mov echo,-1
        over1:              nop
        repeat2:            movdx,offset speed
                            mov ah,9
                            int21h
                            mov ah,8
                            int 21h
                            cmp al,'1'
                            je baud300
                            cmp al,'2'
```

```
                        je baud600
                        cmp al,'2'
                        je baud1200
                        mov dx,offset entry_error
                        mov ah,9
                        int 21h
                        jmprepeat2
baud300:                or serial_init,01000000b
                        jmp over2
baud600:                or serial_init,10000000b
                        jmpover2
baud1200:               or serial_init,11000000b
over2:                  nop

repeat3:                mov dx,offset stop
                        mov ah,9
                        int 21h
                        mov ah,2h
                        int 21h
                        cmp al,'1'
                        je onestop
                        cmp al,'2'
                        je twostop
                        mov dx,offset entry_error
                        mov ah,9
                        int 21h
                        jmp repeat3
onestop:                or serial_init,00000000b
                        jmp over3
twostop:                or serial_init,00001000b
over3:                  nop


repeat4:mov dx,offset parity  ;initialize parity
                        mov ah,9
                        int 21h
                        mov ah,8
                        int 21h
                        cmp al,'1'
                        je noparity
                        cmp al,'2'
                        je oddparity
                        cmp al,'3'
                        je evenparity
                        mov dx,offset entry_error
                        mov ah,09
                        int 21h
                        jmp repeat4
noparity:               or serial_init,00000000b
                        jmp over4
```

```
oddparity:
                or serial_init,00000000b
evenparity:     jmp over4
over4:          or serial_init,00000000b
                nop

repeat5:mov     dx,offset wordlength ;initialise to 7 bits
                mov ah,9
                int 21h
                mov ah,08
                int 21h
                cmp al,'7'
                je bits7
                cmp al,'8'
                je bits8
                mov dx,offset entry_error
                mov ah,09
                int 21h
bits7:          jmp repeat5
                or serial_init,00000010b
bits8:          jmp over5
over5:          or serial_init,00000011b
                nop
                ;initilise modem interrupt INT14h

                mov ah,00
                mov dx,0000
                mov al,serial_init
                int 14h

                call asc_end
                ;capture serial port interrupt
                ;vector and enable interrupts
                sti
                mov bh,dattr
                call cls
                mov dx,offset startup
                mov ah,9
                int 21h

                cmp echo,0h;full duplex mode?
                je fullduplex1;yes,jump.
half_duplex1:   mov dx,offset startup
                mov ah,9
                int 21h
talk3:
                call pc_stat
                jz talk4;nothing waiting...
```

P 214

```
                call pc_in ;read keyboard char

                cmp al,0
                jne ord_key
                cmp ah,exit_key
                je exit
                cmp ah,file_key
                je file_transmit

ord_key:        cmp al,transmit_key;
                jne talk3

                call rtson ;set RTS
                mov al,lock_key
                call com_OUT

repeat: call transmit ;transmit mode
        call receive ;receive mode
        jmp repeat

talk4:  call com_stat
        jz talk3 ;nothing waiting,jump
        call com_in

        cmp al,exit_code
        je forcedexit

        cmp al,lock_key
        jne talk3

again:          call receive
                call transmit
                jmp again

exit:           call rtsoff

                mov al,exit_code
                call com_out

forcedexit:     mov bh,07h
                call cls

                mov dx,offset wincup

talk6;          push dx

                call asc_dsb
                  ;disable serial port interrupt
                  ;and release interrupt vector
```

```
                        pop ax
                        mov al,ch
                        int21h

                        mov ax,4c00h;terminate prog
                        int 21h
file_transmit           call filetransmit
                        jmp half_duplex1

full_duplex1:           mov dx offset buffer_in
                        mov ah,9
                        int21h

talk3:                  call rsum
                        call pc_stat
                        jz talk4

                        call pc_in

                        cmp al,0
                        jne ord_key1

                        cmp ah,exit_key
                        je exit
                        cmp ah,file_key
                        je file_transmit

ord_key1:               call com_out

talk4:                  call com_stat
                        jz talk3

                        call com_in

                        cmp al,file_code
                        je file_receive

                        cmp al,exit_code
                        je forceduxit


                        call pc_out

                        cmp al,c
                        jne talk45
                        mov al,1
                        call ...

talk45:                 jmp talk4
```

```
file_transmitter:       call ftransmit
                        jmp fullduplex1

file_receiver:call freceive
                        jmp fullduplex

pclin1
                        endp
;PROCEDURES START

com_stat proc near
        ;checks asynch status.returns
        ;Z=false if character ready
        ;Z=true if nothing waiting

        push ax


                in al,pic_mask
                or al,10h
                out pic_mask,al

                mov ax,asc_in
                cmp ax,asc_out
                pushf
                in al,pic_mask
                and al,0efh
                out pic_mask,al
                popf
                pop ax

                ret

com_stat endup

com_in proc near
;get char from serial port buffer returns new char in al
                push bx
                in al,pic_mask
                or al,10h
                out pic_mask,al

        com_in1:

                mov bx,asc_out
                cmp bx,asc_in
                je com_in1

                mov al,[bx+asc_buff]

                inc bx
                cmp bx,bufsiz
```

```
            jnz com_in2
            xor bx,bx
com_in2:
            mov asc_out,bx
            push ax

            in al,pic_mask
            and al,0efh
            out pic_mask,al

            pop ax

            pop dx

            ret

com_in endp

com_out proc near
   ;write char in AL to serial port

            push dx
            push ax
            mov dx,com_sts

com_out1:
            in al,dx
            and al,20h
            jz com_out1

            pop ax
            mov dx,com_data
            outb dx,al
            pop dx

            ret

com-out endp

pc_stat proc near
 ;reads keyboard status & returns
 ;Z=false if character is ready
 ;Z=true if nothing waiting
 ;register DX destroyed

            mov al,in_flag
            or al,al
            jnz pc_stat1

            mov dx,1
```

```
                ret
                ;zero_status

                mov ah,0
                int 16h

                mov in_char,al
                mov scan_code,ah
                mov in_flag,0ffh

pc_stat1:

                ret

pc_stat endp

pc_in proc near
        ;read keyboard character
        ;return it in AL
        ;DX maybe destroyed
            mov al,in_flag
            or al,al
            jnz pc_in1

            call pc_stat
            jmp pc_in

pc_in1:mov in_flag,0
            mov al,in_char
            mov ah,scan_code


            ret

pc_in endp

pc_out proc near
 ;write char in AL to the PC's screen

            mov ah,0eh
             push bx
             xor bx,bx
             int 10h
             pop bx

            ret

pc_out endp


pc_pub proc near
```

```
;capture serial-port interrupt
;vector and enable interrupt

        mov ax,3500h+com_int
        int 21h
        mov wordptr oldvec+2,es
        mov wordptr oldvec,bx
        push ds
        mov ax,cs
        mov ds,ax
        mov dx,offset asc_int
        mov ax,2500h+com_int
        int 21h
        pop ds

        in al,pic_mask
        and al,not int_mask
        out pic_mask,al

        mov dx,com_lcr
        in al,dx
        and al,7fh
        out dx,al

        mov dx,com_ier
        mov al,1
        out dx,al

        mov dx,com_mcr
        mov al,09h
        out dx,al

        ret

asc_enb endp
asc_dsp proc near   ;disable interrupt
                    ;release interrupt vector

        in al,pic_mask
        or al,int_mask
        out pic_mask,al

        push ds
        lds dx,oldvec
        mov ax,2500h+com_int
        int 21h
        pop ds
```

```
asc_dsb     ends

asc_int proc far
;interrupt service routine for serial

            sti
            push ax
            push bx
            push dx
            push ds
            mov ax,_DATA
            mov ds,ax
            cli
            mov dx,com_data
            in al,dx
            mov bx,asc_in
            mov [ase_buf+bx],al
            inc bx
            cmp bx,bufsiz
            jne asc_int1
            xor bx,bx

asc_int1:
            mov asc_in,bx

            sti
            mov al,20h
            out pic_eoi,al

            pop ds
            pop dx
            pop bx
            pop ax

            iret
asc_int endp


cls   proc near
  ;clear display using char attribute in BH
  ;registers AX,CX & DX destroyed

            mov dl,columns
            mov dh,24
            mov cx,0
            mov ax,600h
```

```
                        int 10h
                        call home

                        ret

          clrs          endp

home proc near

                        mov dx,0
                        call gotoxy
                        ret
home endp
gotoxy proc near
  ;position cursor,callwithDL=X,DH=Y
                        push bx
                        push ax
                        mov bh,0
                        mov ah,2
                        int 10h
                        pop ax
                        pop bx
                        ret

          gotoxy endp
ftransmit               proc near ;ftransmit file proc...
                          push cx

                          call rfscn

                          mov dx,offset prompt
                          mov ah,9
                          int 21h

                          mov dx,offset file_name
                          mov file_name,40
                          mov ah,0ah
                          int 21h

                          mov bl,file_name+1
                          add bl,02
                          mov bh,00
                          mov file_name[bx],00
                          mov dx,offset file_name
                          add dx,02h
                          mov al,0
                          mov ah,3dh
                          int 12h
                          jnc filesk
                          rol ax,1
                          mov bx,ax
```

```
        mov dx,err_mess_...
        mov bh,0
        mov ah,09h
        int 21h
        mov dl,cr
        mov ah,02h
        int 21h
        jmp exit

filook:
        mov bx,ax
        push bx
        mov cx,maxchar
        mov dx,offset file_...
        mov ah,3fh
        int 21h

        jnc read_ok
        mov ah,9
        mov dx,offset file_e...
        int 21h
        pop bx
        mov ah,3eh ;close f...
        int 21h

        jmp exit

read_ok:
        pop bx
        push ax
        mov ah,3eh
        int 21h
        mov al,file_...
        call com_out

        call rtsoff

        mov dx,offset ...
        mov ah,9
        int 21h

wait_ack:
        call com_stat
        jz wait_ack

        call com_in

        cmp al,file_a...
        jne wait_ack

        mov dx,offset ...
        mov ah,9
        int 21h
```

```
                call rtson
                call delay

                pop ax
                mov cx,ax
                mov bx,offset [illegible]
nextchr:
                mov al,[bx]
                push ax
                call pc_out
                pop ax
                call com_out
                inc bx
                dec cx
                jnz nextchr

                mov al,eof
                call com_out

                call rstoff
wait_ack2:
                call com_stat
                jz wait_ack2

                call com_in
                cmp al,close_ack
                jne wait_ack2

                call clson

                mov dx,offset file_sent
                mov ah,9
                int 21h
exit1:
                pop bx
                ret

transmit        endp


receive
                proc near
             ;procedure for receiving [illegible]
                call rtsoff
                mov dx,offset [illegible]
                mov ah,9
                int 21h
```

```
        mov dx,offset file_n[illegible]
        mov [illegible]
        mov ah,0ah
        int 21h

        mov bl,file_name+1
        add bl,02
        mov bh,00
        mov file_name[bx],00
        mov dx,offset file_name
        add dx,02h
        mov cx,0
        mov ah,3ch
        int 21h

        jnc fileok1
        call [illegible]
        mov [illegible],ax
        mov dx,err_[illegible]
        mov bh,0
        mov ah,09h
        int 21h
        mov dl,cr
        mov ah,02h
        int 21h
        jmp exit
fileok1:
        mov dx,offset [illegible]
        mov ah,[illegible]
        int 21h

        mov bx,ax
        push bx
        mov bx,offset file_buf
        mov cx,[illegible]

        call [illegible]
        call delay

        mov al,file_e[illegible]
        call com_out

        call rtsoff

        mov dx,offset beg[illegible]
        mov ah,9
        int 21h
wait7:
        call com_stat
        jz wait7
```

```
next_char:
        call com_in
        cmp al,eof
        je write_file

        push ax
        call fx_out
        pop ax
        movsb al
        inc ax
        inc cx
        jmp wolzy

write_file:
        pop bx
        push bx
        mov dx,offset file_out
        mov ah,40h
        int 21h

        jnc write_ok

        mov dx,offset write_e
        mov ah,9
        int 21h

        jmp exit2

write_ok:
        pop bx
        mov ah,3eh ;close file
        int 21h

        jnc close_ok

        mov dx,offset close_err
        mov ah,9
        int 21h
        jmp exit2


close_ok:
        call crlf
        call crlf

        mov si,close_ok
        call com_out
        call fx_save

        mov si,offset file_write
        mov ah,9
        int 21h


                        65
```

```
exit2:
freceive
transmit
  ;procedure for transmitting in half duplex
  mode
start:
char_wait:
cont 1:
ord_key1:
cont:
file_transmit1:call ctransmit
```

```
                        jmp start
transmit                endp

receive                 proc near
 ;procedure for receive in half duplex mode
start1:                 mov dx,offset rec_mesg
                        mov ah,9
                        int 21h

                        call rtsoff
wait_char:
                        call com_stat
                        jz wait_char

                        call com_in

                        cmp al,eot_key
                        jz okmode

                        cmp al,file_code
                        jz file_receive1

                        cmp al,exit_code
                        jne cont2

                        jmp forcedexit
cont2:
                        call pc_out

                        jmp wait_char
file_receive1:          call freceive
                        jmp start1
okmode:
                        ret
receive
                        endp
rtson
                        proc near

                        mov dx,com_mcr
                        in al,dx
                        or al,02h
                        out dx,al

                        ret
rtson
                        endp
```

```
rtsoff                  proc near

                        mov dx,com_ncr
                        in al,dx
                        and al,0fdh
                        out dx,al
                        ret

rtsoff                  endp

delay                   proc near

                        push bx
                        push cx
                        mov bx,count1
cntdn1:
cntdn2:                 mov cx,count2
                        loop cntdn2
                        dec bx
                        jnz cntdn1
                        pop cx
                        pop bx

delay                   endp

_TEXT                   ends ;code segment ends

_DATA segments word public'Data'

in_char
in_flag         db 0
scan_code       db 0
columns         db 0
                db 0

msg1
                db cr,lf
                db'DISPLAY mode on text only'
                db cr,lf,'$'
msg1_len equ $-msg1

initial_msg     db'
                        INITIALISATION OF
COMMUNICATIONS PORT
PARAMETERS
                .cr,lf,cr,lf
                db cr,lf,'$'

                        'ENTER MODE OF TRANSMISSION
                db cr, lf,lf,lf,'$'
                db'    If full duplex
```

```
        db cr,lf,cr,lf
        db'          If Half duplex Enter 2'
        db cr,lf,cr,lf,cr,lf,'$'
speed           db 'ENTER SPEED OF TRANSMISSION:'
          db'   If 300 Baud Enter 1',cr,lf,cr,lf
          db'   If 600 Baud Enter 2',cr,lf,cr,lf
          db'   If 1200 Baud Ente. 3',cr,lf,cr,lf
          db'   (See more settings of Modem and
                to e.a.  Depth',cr,lf,cr,lf,'$'


stop            db'ENTER NO. OF STOP BITS:'
          db'   Enter 1 or 2',cr,lf,cr,lf
          db cr,lf,'$'

parity          db'ENTER TYPE OF PARITY BIT:',cr,lf,cr,lf
          db'   If None enter 1',cr,lf,cr,lf
          db'   If oddparity enter 2',cr,lf,cr,lf
          db cr,lf,'$'


wordlenth                 db'ENTER WORD LENGTH:'
                          cr,lf,cr,lf
          db'   If 7 bits,Enter1',cr,lf,cr,lf
          db'   If 8 Bits,Enter2',cr,lf,cr,lf,'$'


echo                      db 0


entry_error               db'Invalid entry,Enter',cr,
                          cr,lf
                          db cr,lf,cr,lf,'$'

serial_init               db 0

startup                   db'          INITIALIZATION
                          OVER',cr,lf,cr,lf,cr,

startup1                  db cr,lf,cr,lf,cr,lf
                          db'You are in half-duplex
                          mode',cr,lf,cr,lf
                          db'To begin ,  enter,
                          press Ctrl-E',cr,lf,cr,lf
                          db'To begin ,  enter,
                          press CR key',cr,lf,cr,lf
                          db'To exit ,  enter,
                          press Ctrl-C',cr,lf,cr,lf
```

```
startup2
                db'You are in transmi...
                Begin Typing',cr,lf...
                db'To goto ...
                press Ctrl-Z',...
                db'To transmit ...
                F2',cr,lf,...
                db'Press F10 to exit ...
                DOS',cr,lf,...
msg2 db
                ...erminal emulator ...
           db cr,lf
msg2_len equ $-msg2

msg3   db 'Exit ... Terminal Emulato...

  db cr,lf
msg3_len equ $-msg3

oldvec   dd0

...c_in dw 0
...c_out dw 0

...c_buf db bufsiz dup(?)

...indup      db'Exit from Terminal Emu...
          db cr,lf,'$'
prompt    db cr,lf,cr,lf,'Please type ...
          filename.ext',cr,lf,...
     db' File name format is d:path\
          filename.ext',cr,lf,...,'$'

...le_name              db 40 dup(0)

...r_mess_pointer  dw 0
                   dw offset err_mess1
                   dw offset err_mess2
                   dw offset err_mess3
...r_mess1            db'      Invalid function ...
                             ...
...r_mess2            db'      File not found',...
                             ...
...r_mess3            db'      Path not found'
                             ...
```

```
file_error:     db      'Error in receiving file'
                db      0dh,0ah,'$'
                db      'File transfer'
                db      'failed.',0dh,0ah,'$'

file_buf        db 'Another Buffer$'

file_success    db      'File was transmitted'
                db      'successfully'
                db      0dh,0ah,'$'

prompt1         db'The remote terminal
                db'about to transmit a fil
                db'to you.',0dh,0ah
                db'Please enter a filename
                db'which you want to save
                db'the file',0dh,0ah
                db'Format of filename
                db'path',0dh,0ah,'$'

file_name       db 40 dup(0)

write_error     db'Error in writing to
                db'creating
                db'file',0dh,0ah,'$'

close_error     db'Error in closing the
                db'file',0dh,0ah,'$'

write_success   db'The file was successfully
                db'received',0dh,0ah,'$'

file_mesg       db'You are now ready to
                db'receive
                db'a file',0dh,0ah,'$'

startup_full    db'Begin typing your
                db'The begin
                db'Press F2'
                db'Press F10
                db'write,0dh,0ah,'$'

create_success  db'
                db'File was created
                db'

identity        db'Begin typing your
                db'
```

```
                                    the ready sig
end_wait
                                    ;0 cr,lf,cr,lf,'Rele
begin                               acknowledgement',cr,
                                    db cr,lf,cr,lf,'Pkt
                                    received',cr,lf,'$'

_DATA  ends ;data segment ends

STA    segment  para stack 'STACK'  ;set
                  segment begin

           db 128 dup(?)

STA    ends

end pclink
```

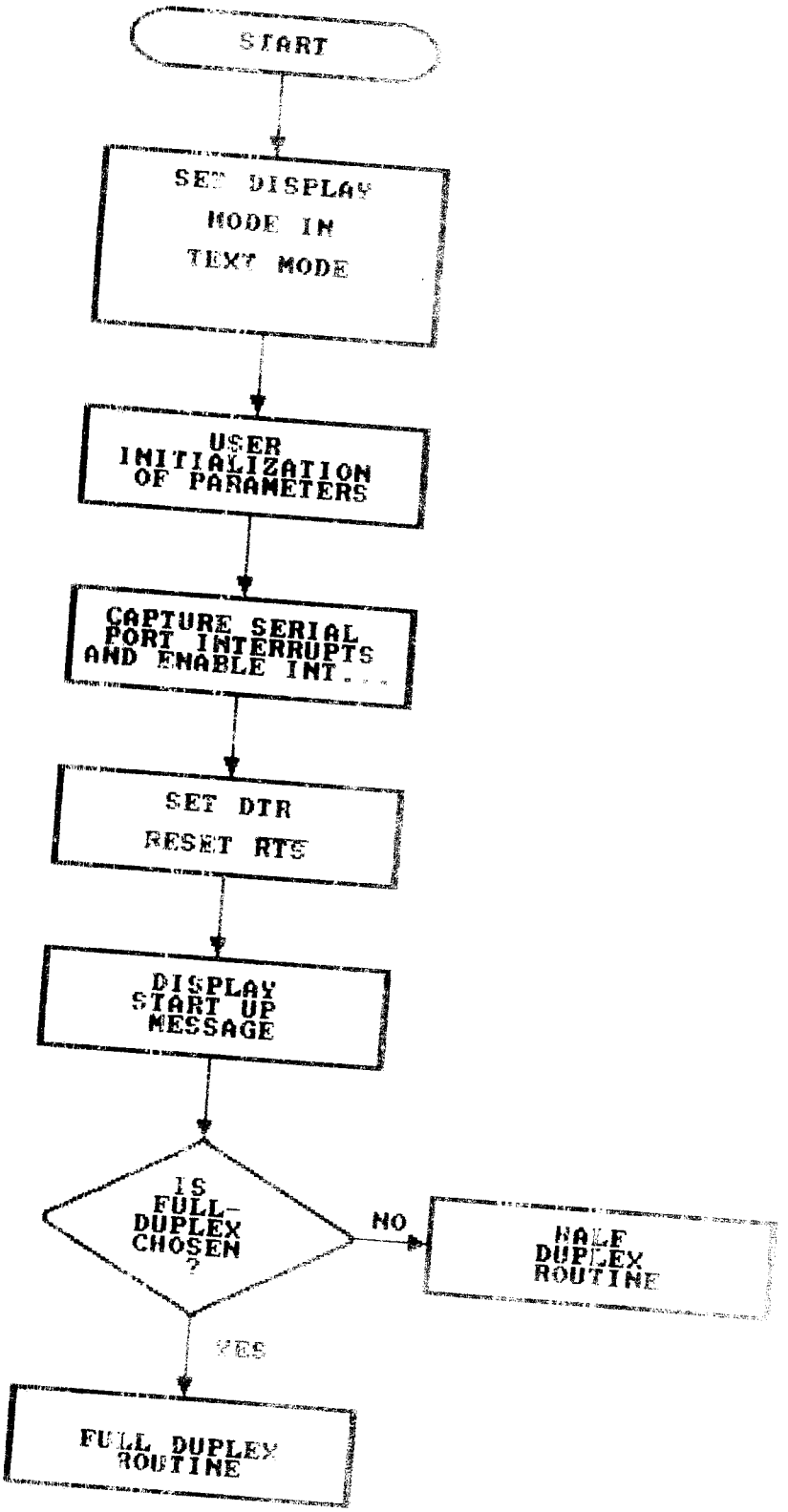


FIG 4.[illegible] FLOWCHART FOR PC LINK PROGRAM

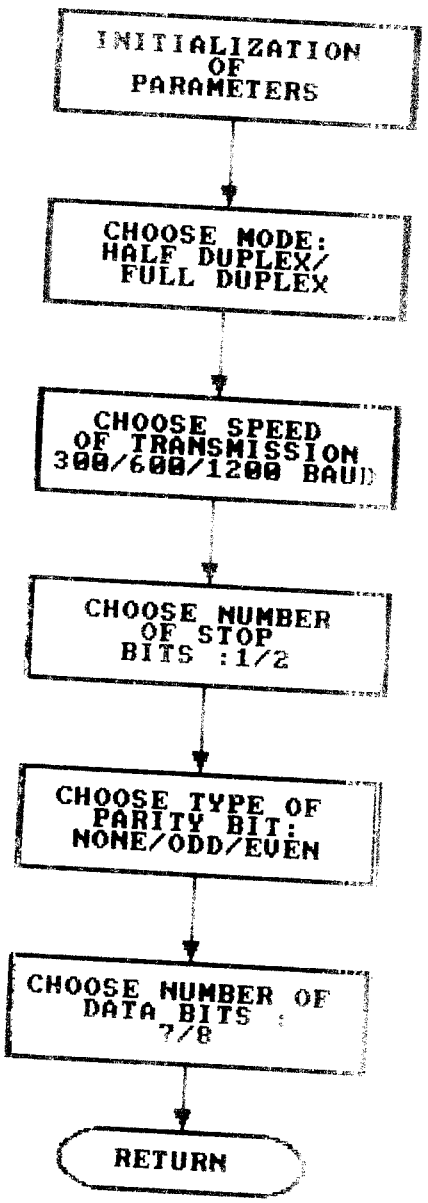


FIG 4.3b: INITIALIZATION FLOWCHART

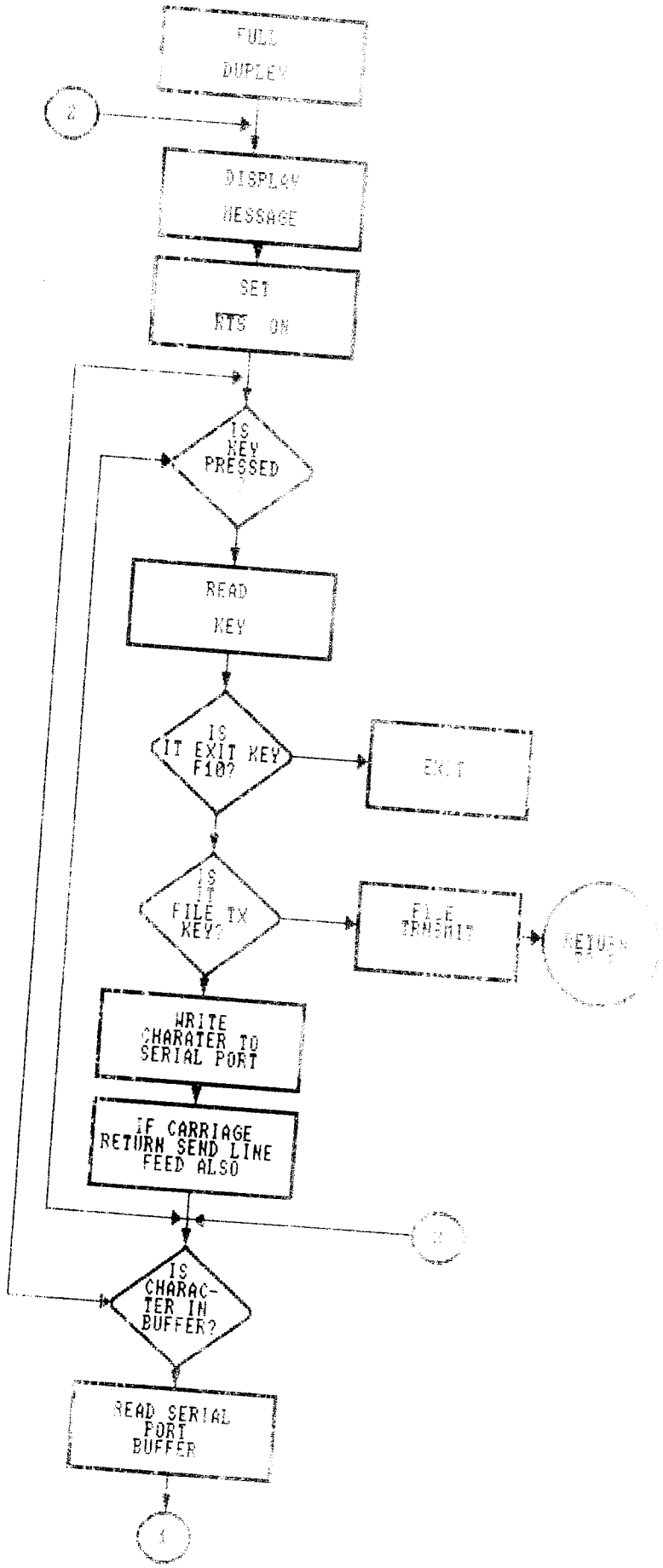


FIG 4.3C FULL DUPLEX MODE (CONTD...

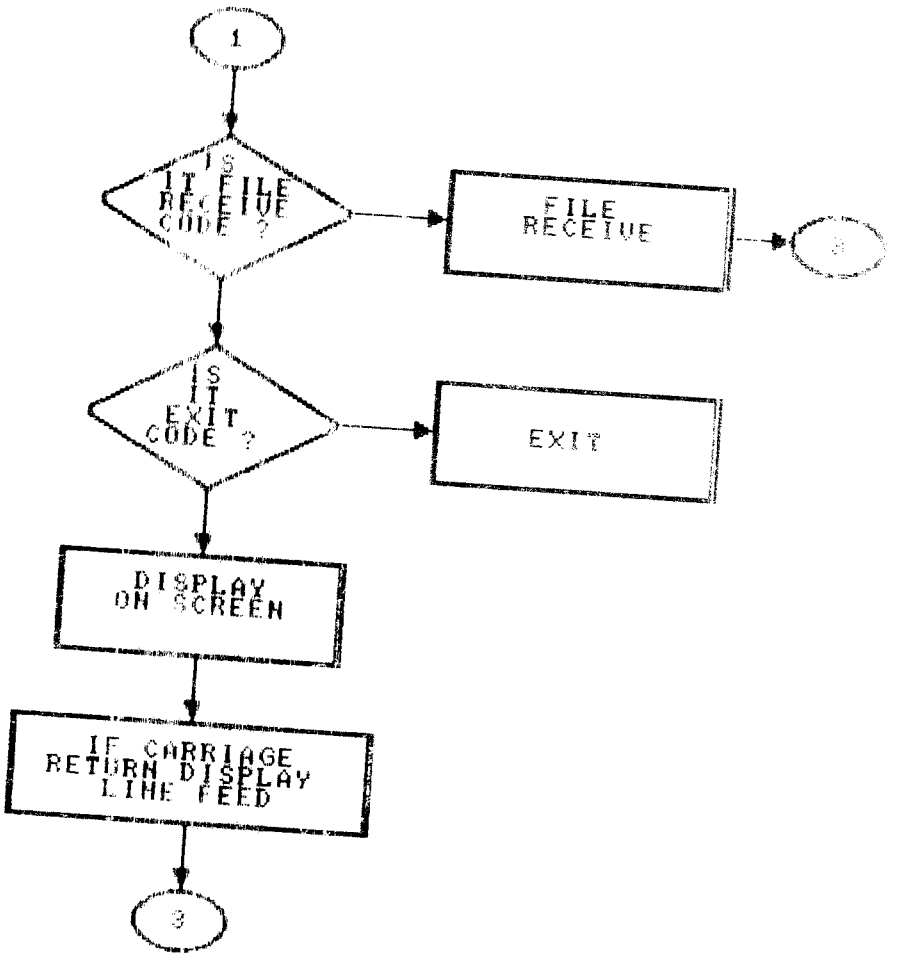


FIG 4.3C FULL DUPLEX MODE

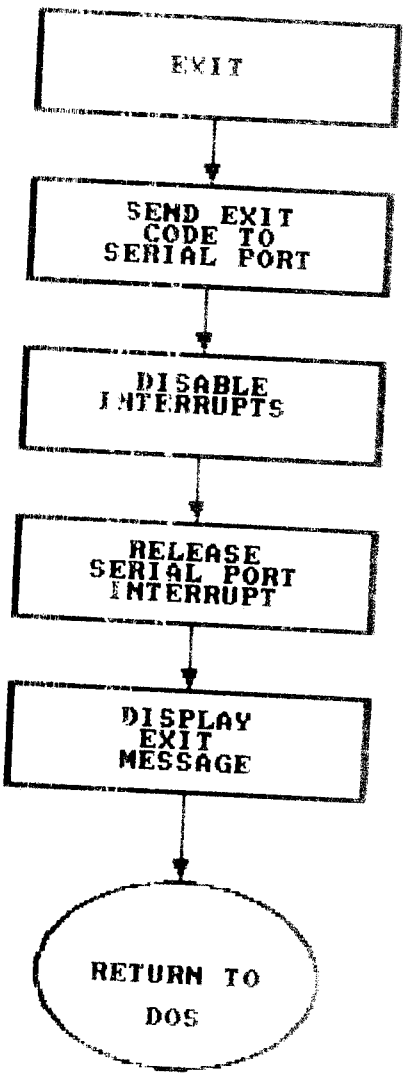


FIG 4.3d EXIT FLOW CHART

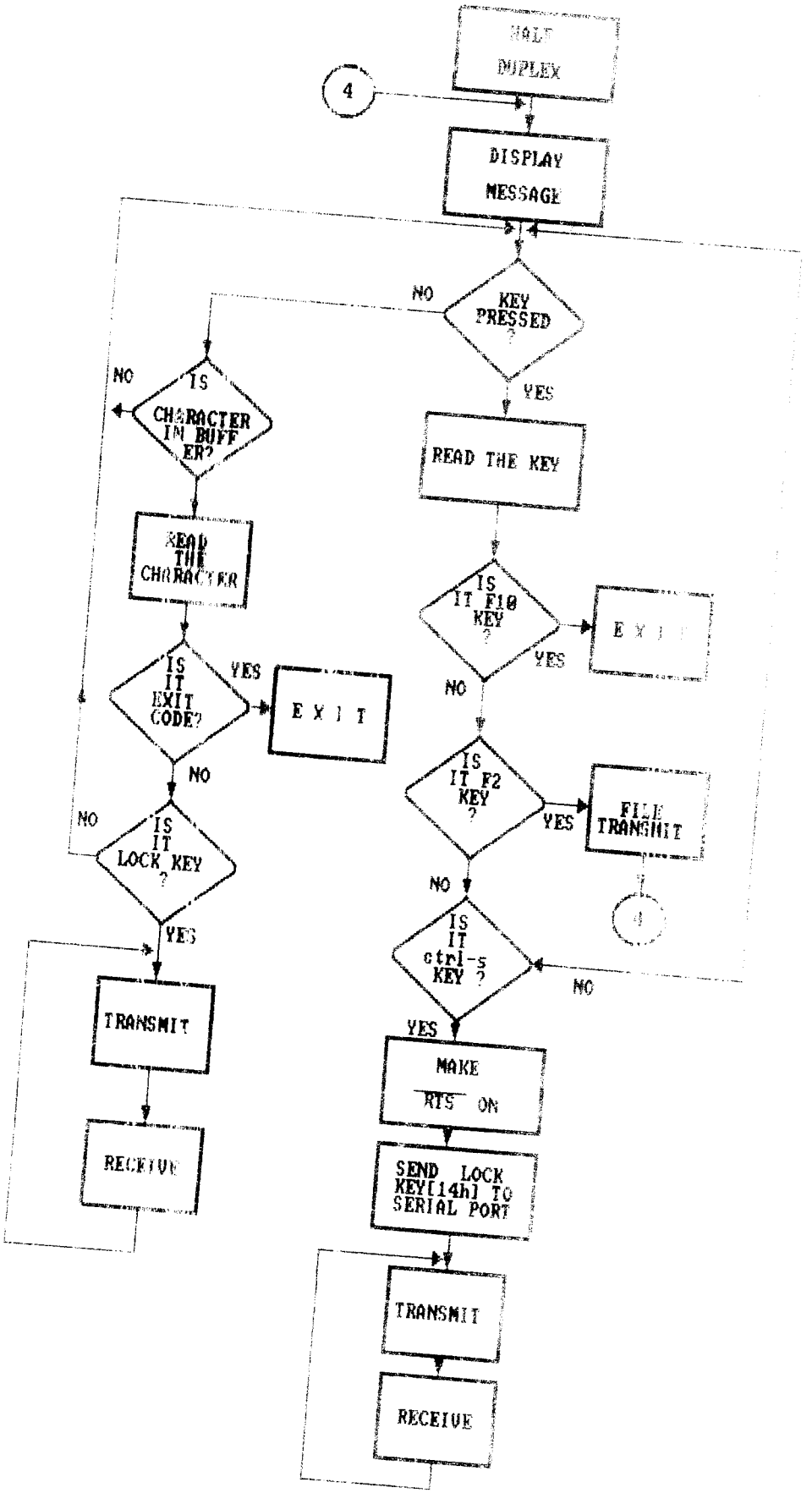
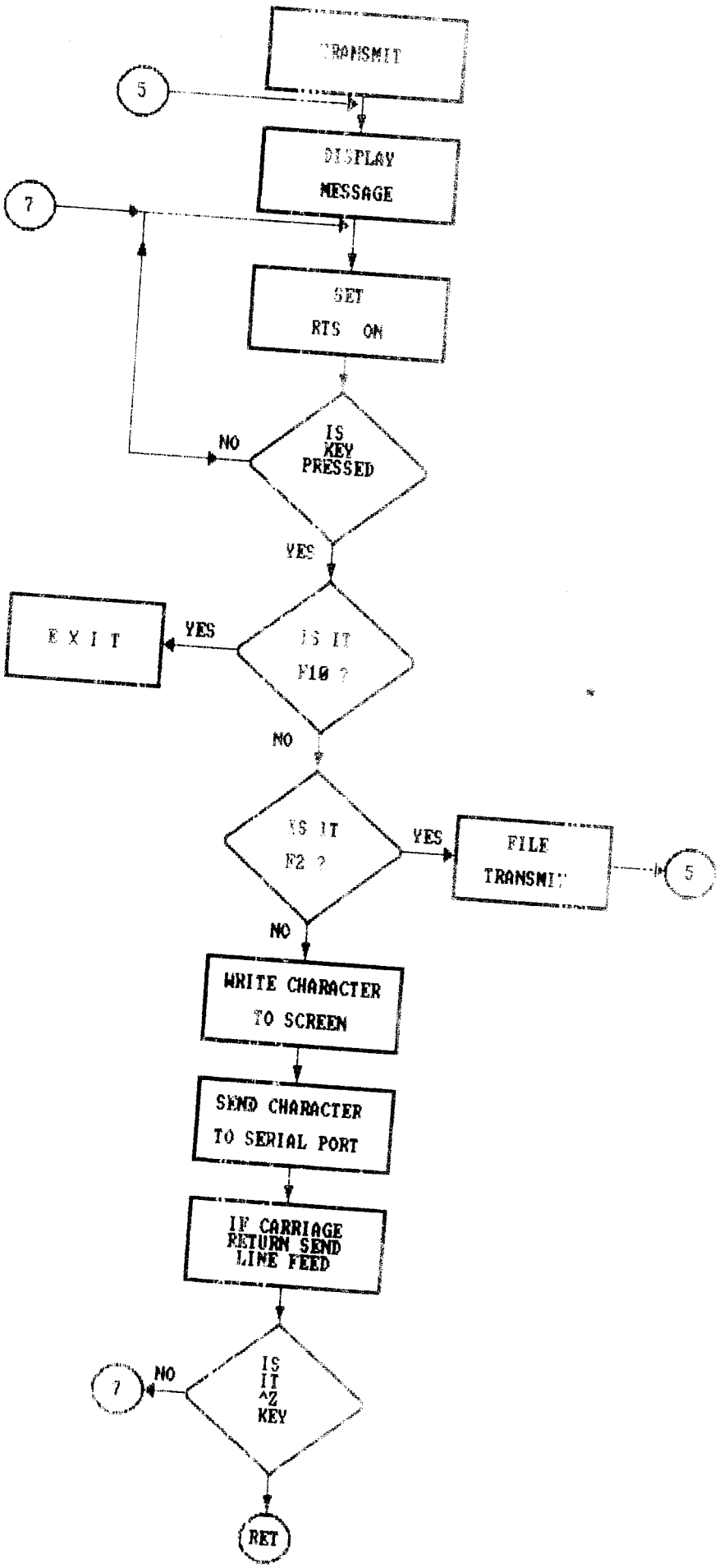


Fig 4.3e HALF DUPLEX MODE


TRANSMIT
5
DISPLAY
MESSAGE
7
SET
RTS ON
NO
IS
KEY
PRESSED
YES
EXIT
YES
IS IT
F10 ?
NO
IS IT
F2 ?
YES
FILE
TRANSMIT
5
NO
WRITE CHARACTER
TO SCREEN
SEND CHARACTER
TO SERIAL PORT
IF CARRIAGE
RETURN SEND
LINE FEED
7
NO
IS
IT
^Z
KEY
RET

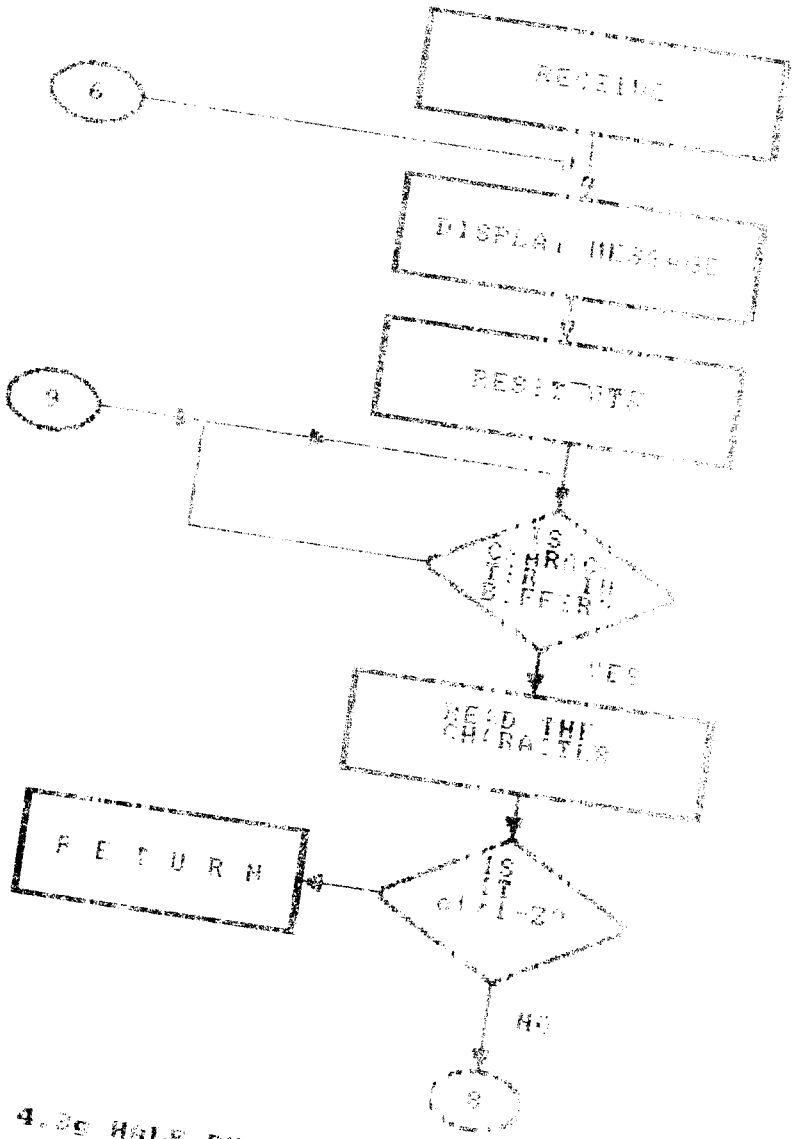


FIG 4.2c HALF DUPLEX RECEIVE MODE

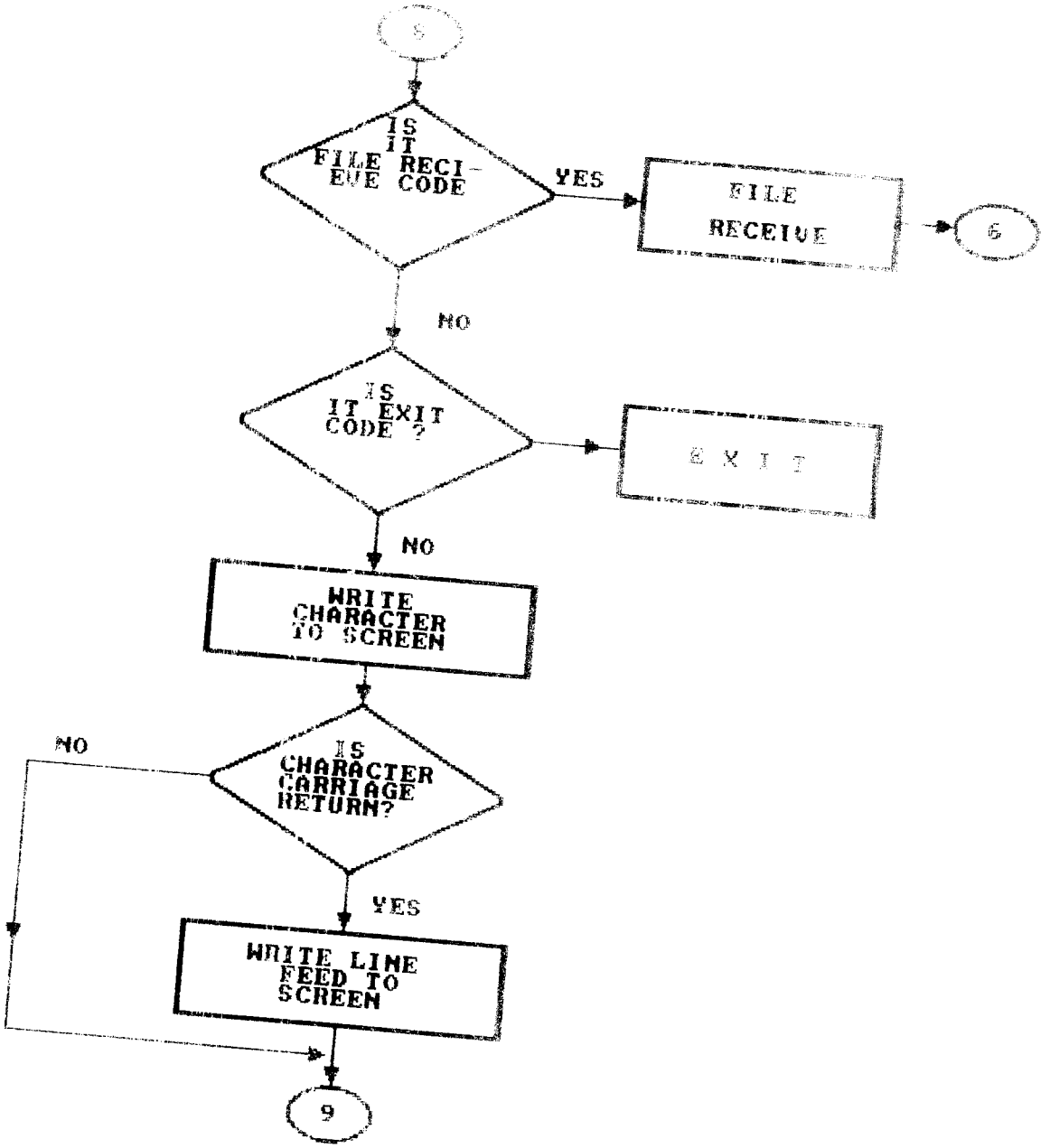


FIG 4.39 HALF DUPLEX RECEIVE MODE

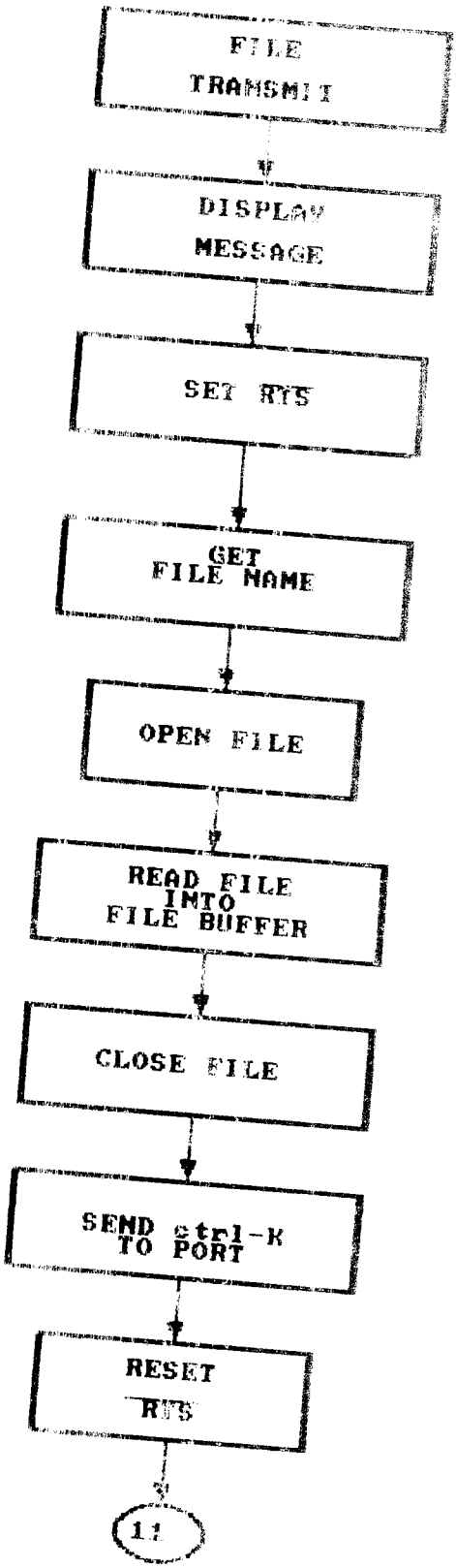


FIG 4.3b FILE TRANSMIT MODULE (CONTD.)

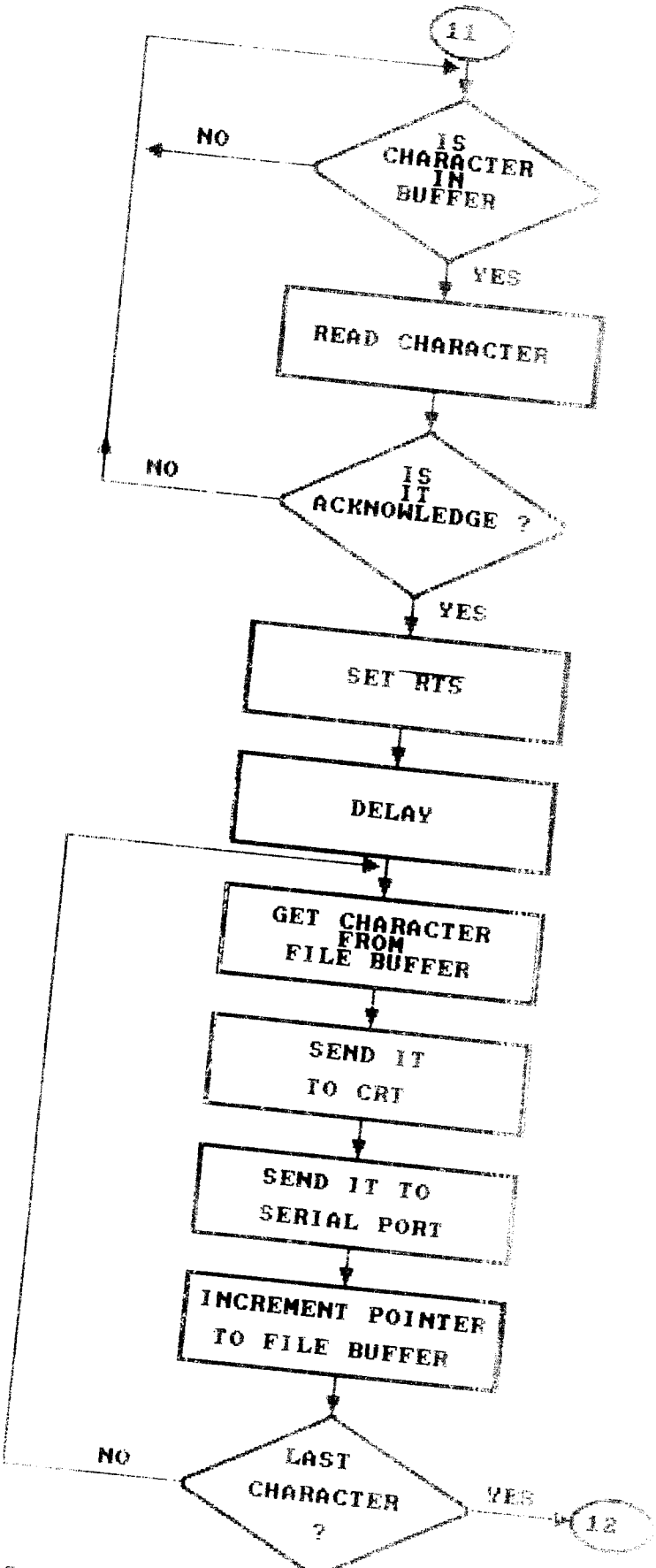


FIG 4.3h FILE TRANSMIT MODULE (CONTD.)

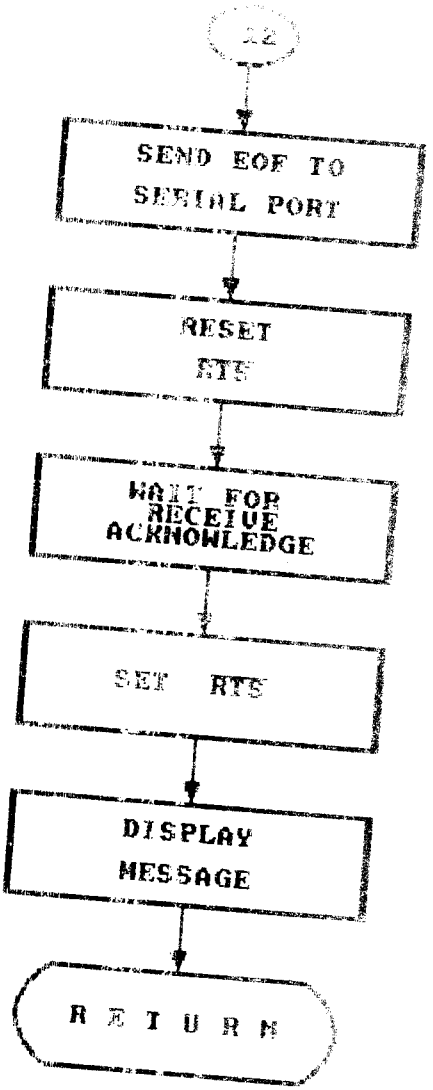


FIG 7.3h FILE TRANSMIT MODULE

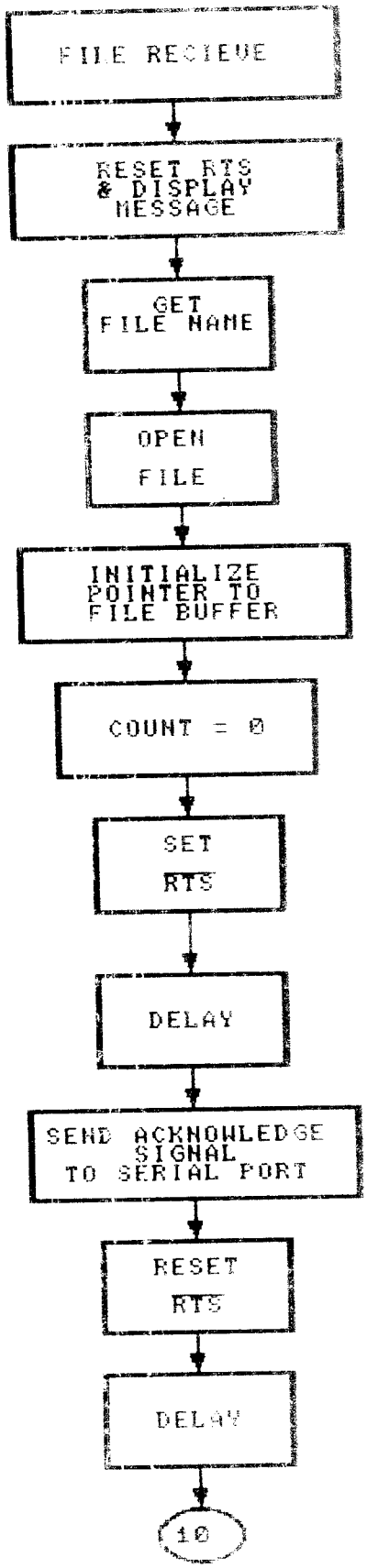


FIG 4.3(:FILE RECEIVE MODULE (CONTD...)

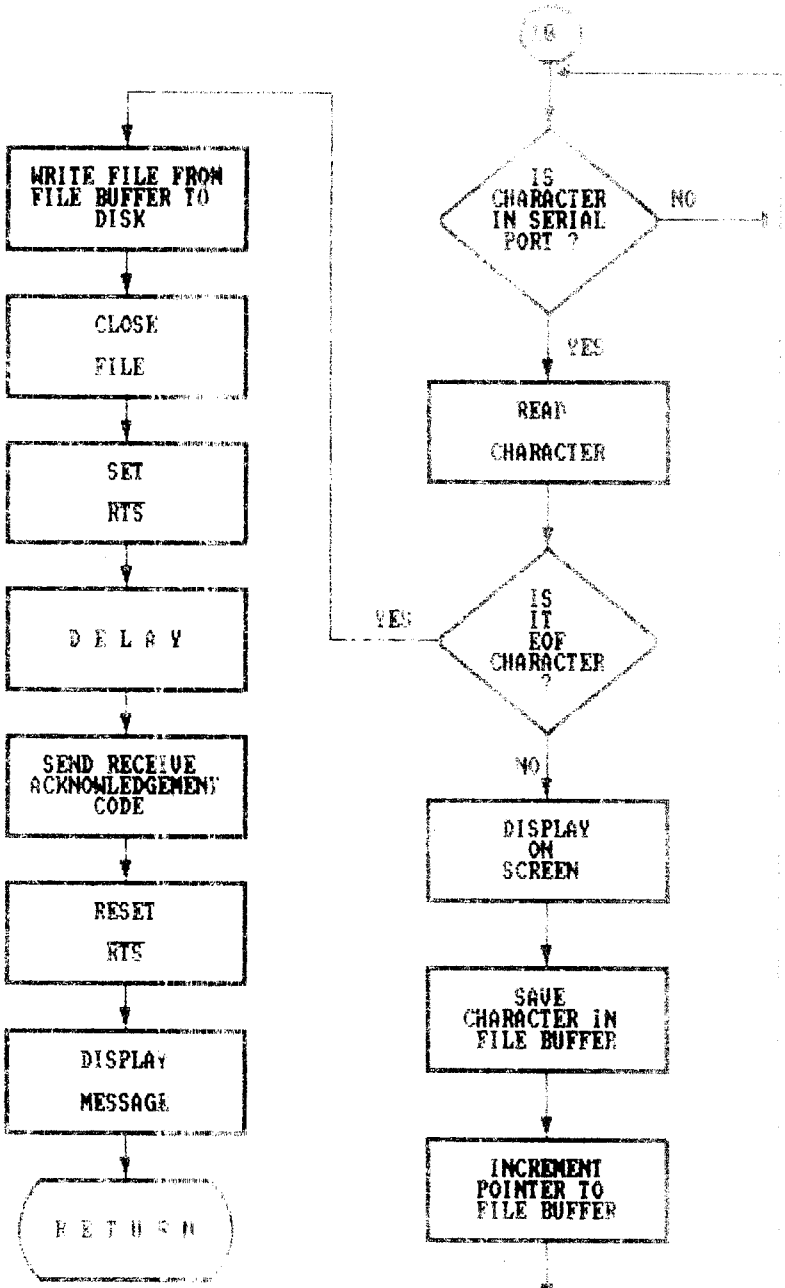


FIG 4.3: FILE RECEIVE MODULE

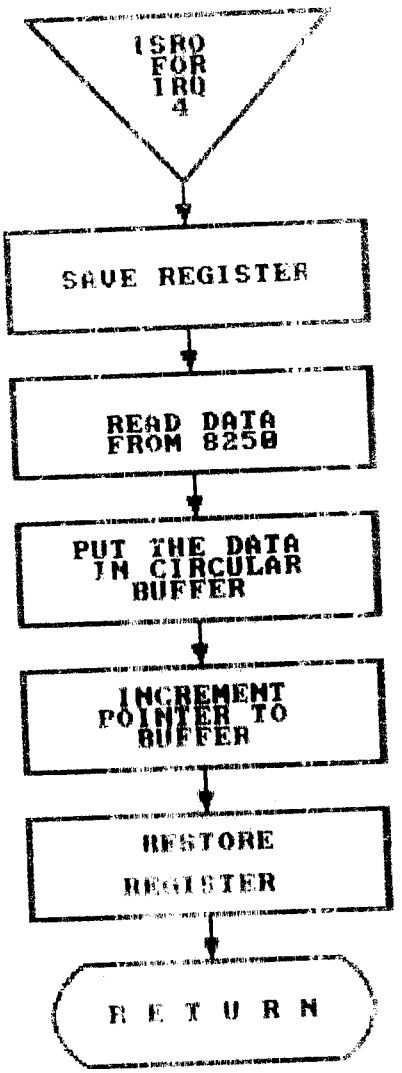


FIG 4.3J :INTERRUPT SERVICE ROUTINE

CHAPTER V

System Integration, Installation and Testing

5.1 Introduction

The block diagram of the PC to MODEM and MODEM to PC transfer is shown in figure 5.1. This chapter deals with the IBM PC communication hardware details, RS-232 standards and the results obtained.

The circuit diagram is shown in the Fig 5.3. The system testing was performed seperately for hardware and software. Then, the integrated testing was performed.

5.2 PC Hardware Details

The IBM PC and its compatible are porvided with a serial communications adapter using 8250 UART. It is an Asynchronous Adapter System providing for 50 baud to 9600 baud. The adapter can be programmed to any of these values. Five, six, seven or eight bit characters with 1, 1 1/2 or 2 stop bits are supported.

5.2.1 Universal Asynchronous Receiver Transmitter (UART)

The heart of the system communications Adapter is the 8250 UART LSI chip. The block diagrammatic representation of the same is presented in figure 5.2. There are 10

addressable registers. There is a baud rate generator, line control register, Modem control register which are the control registers. They are used to configure the 8251 for the required specifications. The status can be verified by examining the status registers. The Appendix 3 gives details about the 8250 UART and the communication adapter.

5.2.2 RS-232 C Levels

The personal computer output from the UART is given to a level converter (TTL to RS-232) which converts the +5V to -12V which is logic 1 and 0V to +12V which is logic 0.

5.3 RS-232 C standards

The RS-232 standards were developed by the EIA (Electronics Industries Associaton) for serial input, communication. The RS-232 connector details are shown in fig 5.4. The serial I/O data pattern is given in fig 5.5.

The serial data shown is in Asynchronous format as supported by the communication Adapter in the PC. The number of stop bits baud rate, number of data bits can be varied by programming. The voltage +3 to +15 is taken as logic 0 and the voltage between -3 to -15V is taken as logic 1. The voltage levels of +12 and -12V are commonly used.

5.4 Hardware Test Results

This section deals with the testing of [illegible] circuit designed in chapter 3

5.4.1 Device connection Tests

The power supply PCB is designed for + 15V dc and 250 mA, input. It provides a regulated power supply of -5V and + 12. The input +15V is given from a voltage source and the output voltages are noted. The absence of ripples is verified in a CRO. The observed voltage are

V1 = 4.97 V V2 = -5.02 V

V3 = +12.01 V V4 = -12.01 V

which are acceptable.

5.4.2 DTE interface tests

The data terminal equipment (computer) has to be connected to the modem through the RS-232. The level conversion from TTL to RS-232 is performed by a MC1488 which is a line driver-Transmitter. The reverse level conversion is performed by MC 1489 which is a line driver receiver.

The RS-232 voltages are applied to data IN and the output at TXD and the status of LED was noted and tabulated in Table 5.1

The test results are compared with theoretic[illegible] voltages and found to be equal. Next the data out connections are tested. The received data output produces '1' or '0' bit information at TTL level. These are applied as input and the data output and the status of the LED 2 are noted. They are tabulated in Table 5.2

The LED2 should go ON when the received data is low (0) and should go 'OFF' when the received data is high (1). This was observed to be so.

Next the carrier detect signal was tested. The carrier detect output is an active low signal. Therefore when the carrier detect output is low, it indicates the presence of carrier signal on the line. Therefore the carrier indicator (LED3) should glow. This was tested to be so. The results are tabulated in Table 5.3

5.4.3 Line interface test

The received carrier (RC) input and the transmit carrier (TC) output are interfaced to the telephone line through a variable gain buffer. The gain can be adjusted by adjusting the trimmer potentiometer. The trimmer is set to a position which gives the voltage level as required by the telephone line standards. the analog-in input is connected to a signal generator and the RC terminal is connected to

CRO. The input signal's amplitude is also noted [illegible] for different settings of the potentiometers [illegible] amplitude is noted. The gain is set for unity. This is verified for frequencies in the audio frequencies in which the modem is to operate. Simislarly TC input [illegible] is set

5.4.4 Testing the assembly

The PCB as shown in fig 5.2 is tested. A signal generator is used as the data terminal input to produce +12 and -12 levels. The frequencies for mark and space are noted for 1200 bauds

The FSK waveform is given as an input to [illegible] another modem. The demodulation was performed and the original signal was got back. For receiving in 1200 baud alone the RTS input must be high in the case of receiving modem. In other mode settings the RTS input can be low as they involve full duplex transfer. The manual of AM7910 is given in the appendix.

5.5 Software Testing : Null modem testing

The software testing was performed by connecting two computers directly using a RS-232 connector. Only three pins TXD, RXD and GND were used. The program written in 8088 assembly language is executed

5.5.1 Full Duplex Testing

In full duplex mode, both terminals can transmit and receive data simultaneously. The keyed in details from the keyboard of one computer got transferred to the other computer. At the same time of transmitting data the terminals receive data. The program is terminated by pressing F10.

5.5.2 Half duplex testing

In half duplex mode, only one terminal can transmit data at a time while the other can only receive. When it goes into receive mode, the control is taken over by the remote terminal which is transmitting. The message typed in at the transmitting terminal appears in both the terminals. The data transfer is stopped only by the transmitting terminal.

5.5.3 File Transfer

In this mode files can be tranferred from one computer to another in either full duplex or half duplex fashion. The file transferred can be given a new name and extension. The testing is performed.

5.6 Integrated Testing

The modem is connected to the RS-232 output of the port COM1 of the computer. The transmitting modem is configured to the following specifications.

Speed of transmission	:	1200 baud
No.of stop bits	:	1 bit
No.of data bits	:	7 bits
Parity	:	Even
Mode of transmissions	:	Half-duplex

The receiving modem is similarly configured and the software is executed. The file transfer is performed. Data is keyed in and the data transfer is observed.

Table 5.1 DTE INTERFACE TEST RESULTS

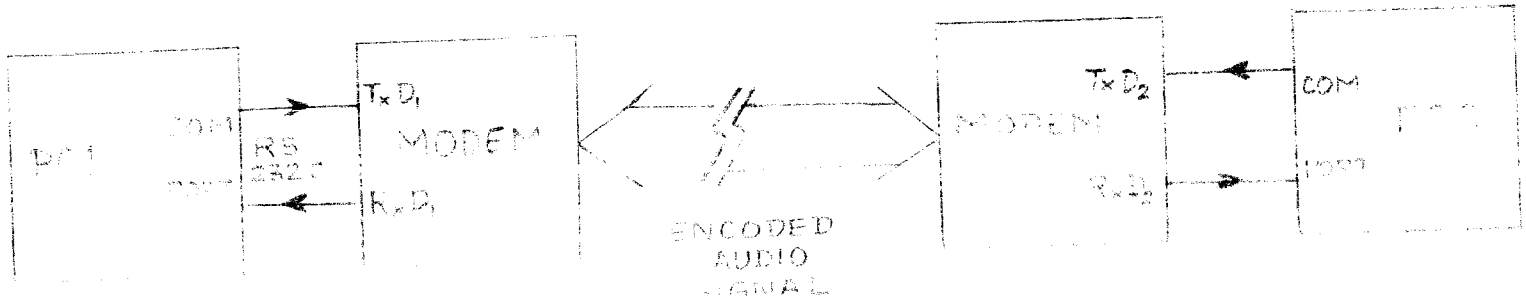
DATA INPUT	TYPE			
	THEORETICAL	PRACTICAL	THEORETICAL	PRACTICAL
RR				
No Load	High	High	ON	ON
Logic 1 -12V	+5V	+5V	ON	ON
Logic 2 +12V	0 V	0 V	OFF	OFF

TABLE 5.2 DATA OUTPUT TEST RESULTS

RD	DATA OUTPUT		LED 2 STATUS	
	EXPECTED	OBSERVED	EXPECTED	OBSERVED
0V Logic 0	+12V	+12V	OFF	OFF
+5V Logic 1	-12V	-12V	ON	ON

TABLE 5.3 CARRIER DETECT TEST

CARRIER DETECT	LED 3 STATUS		MEANING
	EXPECTED	OBSERVED	
Low	ON	ON	Carrier is present at RC input
High	OFF	OFF	No carrier is present


T×D1
MODEM
R×D1
ENCODED
AUDIO
T×D2
COM

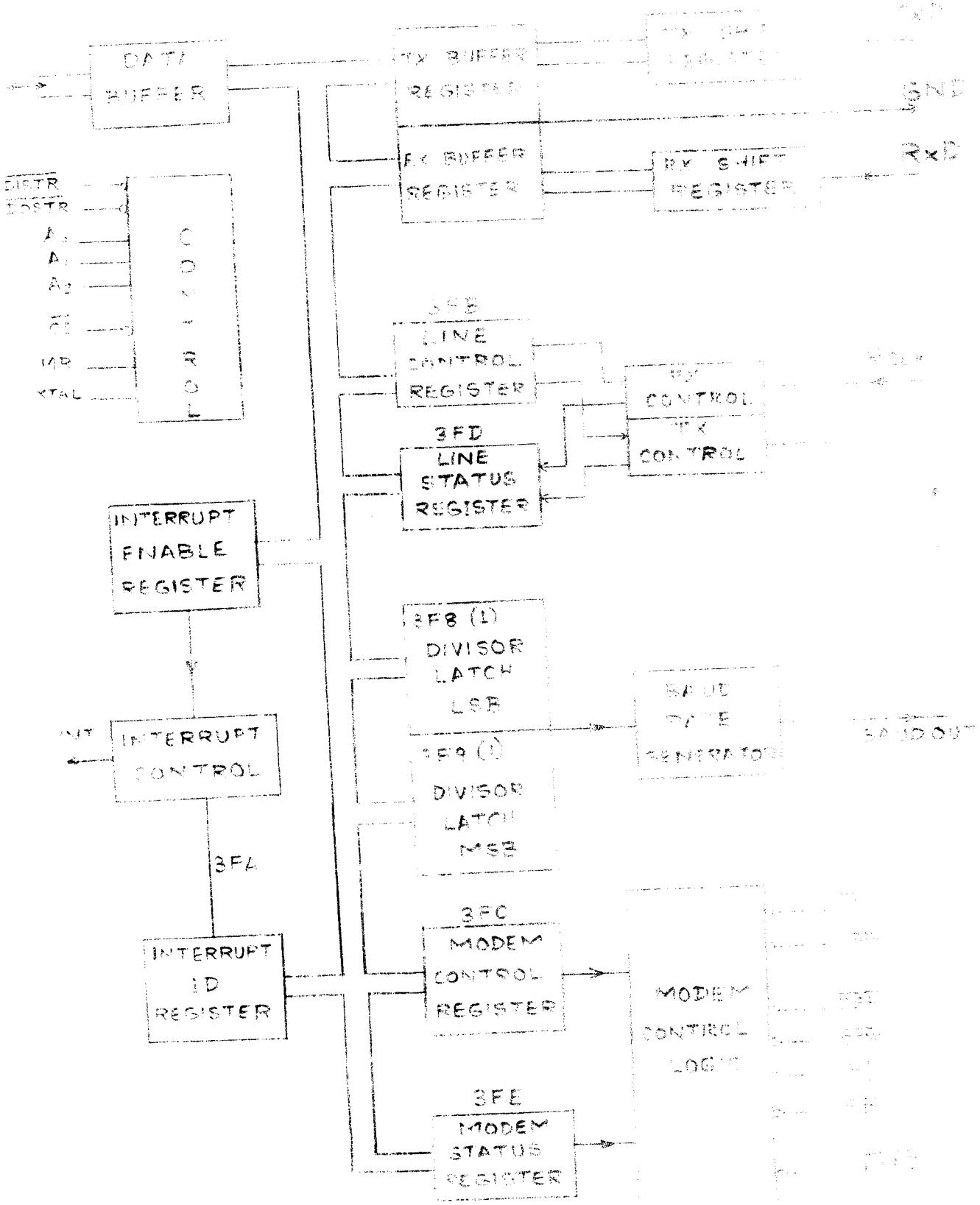


FIG 5.2 BLOCK DIAGRAM OF INS

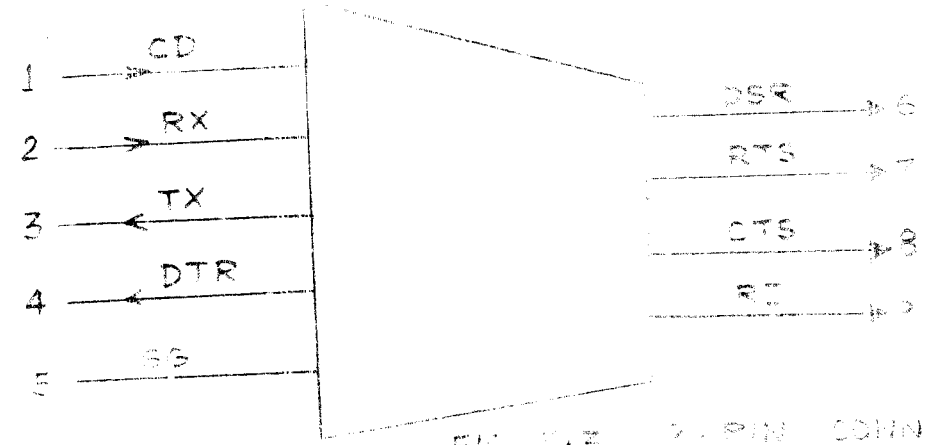
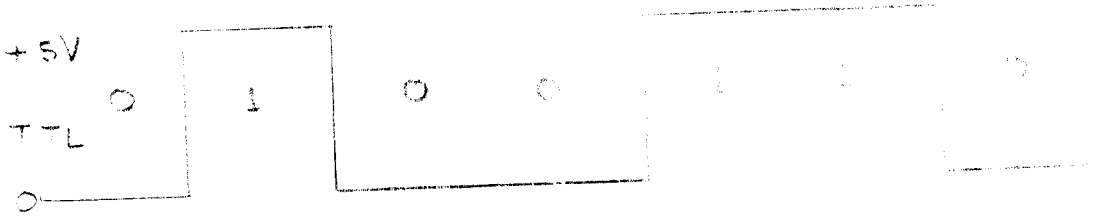


FIG 5.3 9 PIN CONNECTOR DETAILS



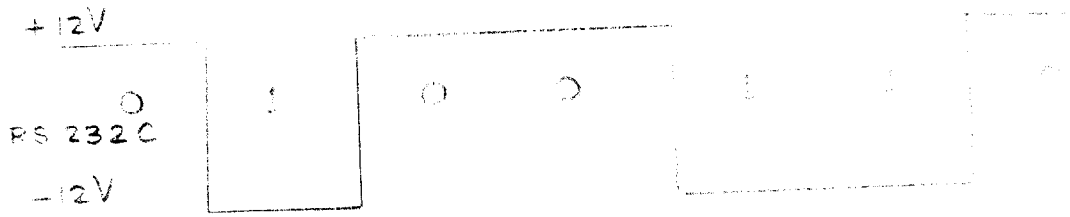


FIG 5.4 TTL AND RS 232 LEVELS

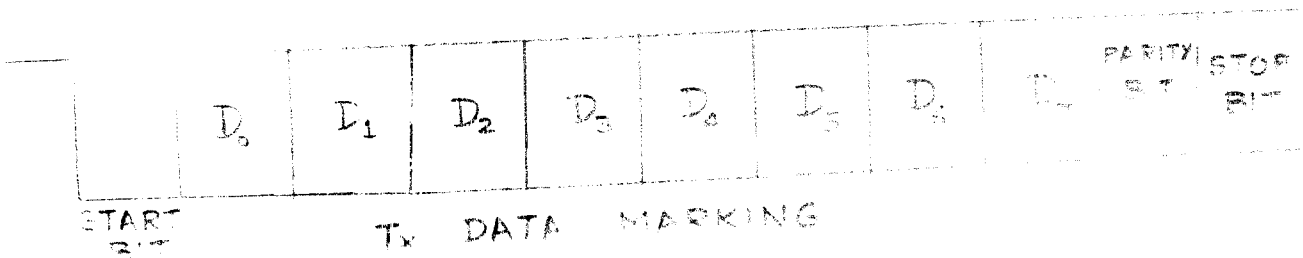
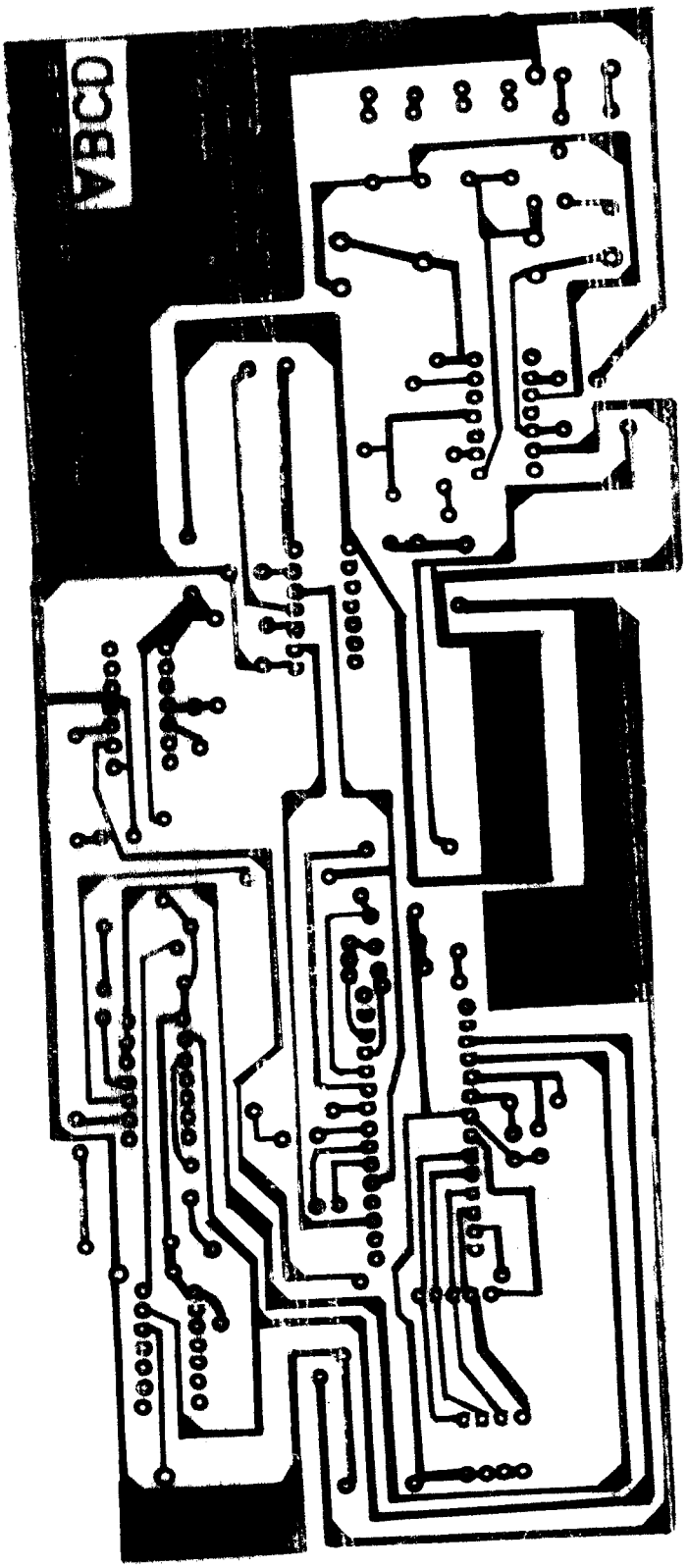


FIG 5.5 SERIAL OUTPUT DATA PATTERN


ABCD

CHAPTER VI

CONCLUSION

In this project, a modem has been used to transfer data between two computers, using telephone lines. The project has been successful and the transfer of data between computers, has been demonstrated.

The highlights of this project are as follows

1. The communication software has been developed for the terminal emulator mode and also the file transfer mode.

2. The voltage supply required for the circuit operation is easily arranged.

3. The speed of data transmission can be programmed by the software.

The circuit design has the following important considerations.

1. The ground connections are critical. The maximum allowed voltage difference should be 50 mV.

2. All the voltages are to be within + 10% tolerance.

In recent days fibre optic communication is rapidly advancing. However, fibre optic cables have not yet been provided for individual subscribers and hence these modems are now put on as Add-on cards in the recent models of IBM PCs.

PCs.

REFERENCES

1. Ray Duncan, 'Advanced MS-DOS Programming' Micro Soft Press, New York, 1988.

2. Douglas V.Hall, " Micro Processors and Interfacing - Programming and Hardware ", McGraw Hill Inc, New York, 1986.

3. K.Padmanathan, S.Ananthi and R.S.Sankaran " Modem -Electronics For You - pp.105-115 August 1994

APPRENDIX

MODEM CONFIGURATIONS

STANDARD	BIT-RATE	DUPLEX	FEATURES
Bell 103	300	Full	Originate
Bell 103	300	Full	Answer
Bell 202	1200	Half	
Bell 202	1200	Half	Line Equalizer
Bell 202*	1200	Half	150b Back Channel
Bell 202*	1200	Half	150B/Line Equalizer
CCITT V.21	300	Full	Originate
CCITT V.21	300	Full	Answer
CCITT V.23 Mode 2	1200	Half	
CCITT V.23 Mode 2	1200	Half	Line Equalizer
CCITT,V.23 Mode 2*	1200	Half	150B/Back Channel
CCITT V.23 Mode 2*	1200	Half	150B/Line Equalizer
CCITT V 23 Mode 1	600	Half	