THE VOICELESS TELEPHONE

THE VOICELESS TELEPHONE

by

STEVEN MARK VAN DER LINDE

Submitted in part-fulfilment of the requirements laid down

for the

Master's Diploma in Technology

in the School of Electrical Engineering,

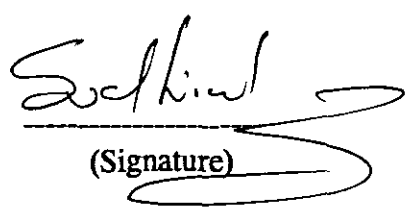
Cape Technikon.

Date of submission : 1 December 1991

DECLARATION

I declare that the contents of this thesis represents my own work and the opinions contained here are my own. It has not been submitted before for any examination at this or any other institution.

S.M. Van der Linde



(Signature)

SYNOPSIS

Communication in all its various forms, has always played an important role in both the business and social environments. The conventional telephone, taken more often than not for granted, is responsible for keeping over five million people in South Africa alone, in daily contact. For the deaf and mute society, of which their are approximately 300 000 in South Africa, the telephone, on its own, has remained a useless gadget.

Without the aid of a personal computer or terminal and a modem, communication for the deaf via the PSTN (Public Switched Telephone Network) has been impossible. Use of computers may be one way of overcoming this obvious problem, but expense now becomes the more important issue.

To analyse the situation, two issues had to be taken into consideration. The first is, what makes the above solution so expensive, and the secondly, is this expensive equipment necessary to complete the relatively simple task of interactive communication. The technology built into todays personal computers is continuously changing and in order to keep up with these changes, regular upgrades to the computer are necessary if one intends being able to recover ones investment at a later stage. The cost of a modem, with its sophisticated error-correction routines and auto-dial software, can also increase the initial outlay considerably. Bearing these costs in mind, it must now be investigated how one can achieve the objective of communicating with only the bear essential.

By replacing the PC's monitor with a Liquid Crystal Display, the powerful processor with a relatively simple one, eliminating the disk storage entirely, reducing the on-board

ROM/RAM memory, and finally, substituting a single-chip low speed modem for the free standing modem, the cost can be drastically reduced. By combining all these components together and developing a program to control them, the result is the "Voiceless Telephone".

OPSOMMING

Kommunikasie, in al sy verskillende fasette, het nog altyd 'n belangrike rol in die besigheids en sosiale wêreld gespeel. Die konvensionele telefoon, wat gewoonlik as van selfsprekend aanvaar word, is verantwoordelik om 5 miljoen mense in Suid Afrika alleen, in daaglikse verbinding te hou. Vir die ongeveer drie honderd duisend doof en stom inwoners van Suid Afrika, is die telefoon, soos ons dit ken, heeltemal ontoeganklik.

Sonder die hulp van 'n persoonlike rekenaar, of terminaal en modem, was kommunikasie deur middel van die Poskantoor se telefoon netwerk, nog altyd onmoontlik vir hierdie deel van ons gemeenskap. Om n rekenaar te gebruik, is een metode om die probleem te oorbrug, maar die koste verbode aan so 'n stelsel plaas dit buite bereik van die verbruiker.

Om die situasie te analiseer, moet daar na twee aspekte gekyk word. Ten eerste: wat maak die rekenaar so duur, en tweedens; is daar nie alternatiewe goedkoper metodes wat gebruik kan word om dieselfde taak te verrig nie.

Die tegnologie in vandag se rekenaars verander by die dag, en om tred te hou met die veranderings, is dit nodig om gereeld opgraderings te maak aan jou rekenaar toerusting.

Die koste van die modem, met sy gesofistikeerde fout-korrigerings vermoëns en automatiese skakeling sagteware, vergroot die aanvanklike koste baie. Met hierdie koste in gedagte, moet daar nou gekyk word hoe kommunikasie bewerkstellig kan word tussen

dowes en stommes met net die nodigste toerusting.

Deur die rekenaar se monitor te vervang met 'n vloeibare kristal eenheid, die kragtige mikroverwerker aanboord die rekenaar met 'n relatief eenvoudige een, om weg te doen met die harde skyf, om die geheue van die rekenaar te verminder en die modem te vervang met 'n enkele geintegreerde laespoed modem "chip", kan die koste baie verminder word. As ons al hierdie komponente kombineer en 'n program ontwikkel om die mikroverwerker te beheer, is die resultaat die "Stemlose Telefoon".

TABLE OF CONTENTS

TABLE OF CONTENTS(Cont.)

TABLE OF CONTENTS(Cont.)

LIST OF DIAGRAMS AND FLOWCHARTS

LIST OF TABLES

APPENDICES

1.0 INTRODUCTION

The Voiceless Telephone has been specifically designed to allow the deaf society to make use of the Public Switched Telephone Network (PSTN) within South Africa. It is a cost affective substitute for the conventional Personal Computer, modem and related software; ie. communication package. Its use is limited to South Africa because it has been designed around the specifications laid down by the South African Post and Telecommunications (SAPT).

The Voiceless Telephone can be divided into several unit, namely:

* The Processor - responsible for control of the unit in accordance with the application program;

* Display - responsible for the visual communication to the user;

* The Tone Detection Circuitry - responsible for interrogating the telephone line and establishing the state of the line at specific times;

* The Ring Detection Circuitry - responsible for the detection of both dial pulses and ring tone; and

* The Keyboard - responsible for command input to the processor and editing of messages being transmitted.

Each of the above units will be discussed in detail under the relevant headings.

A complete circuit diagram can be found on page 2.

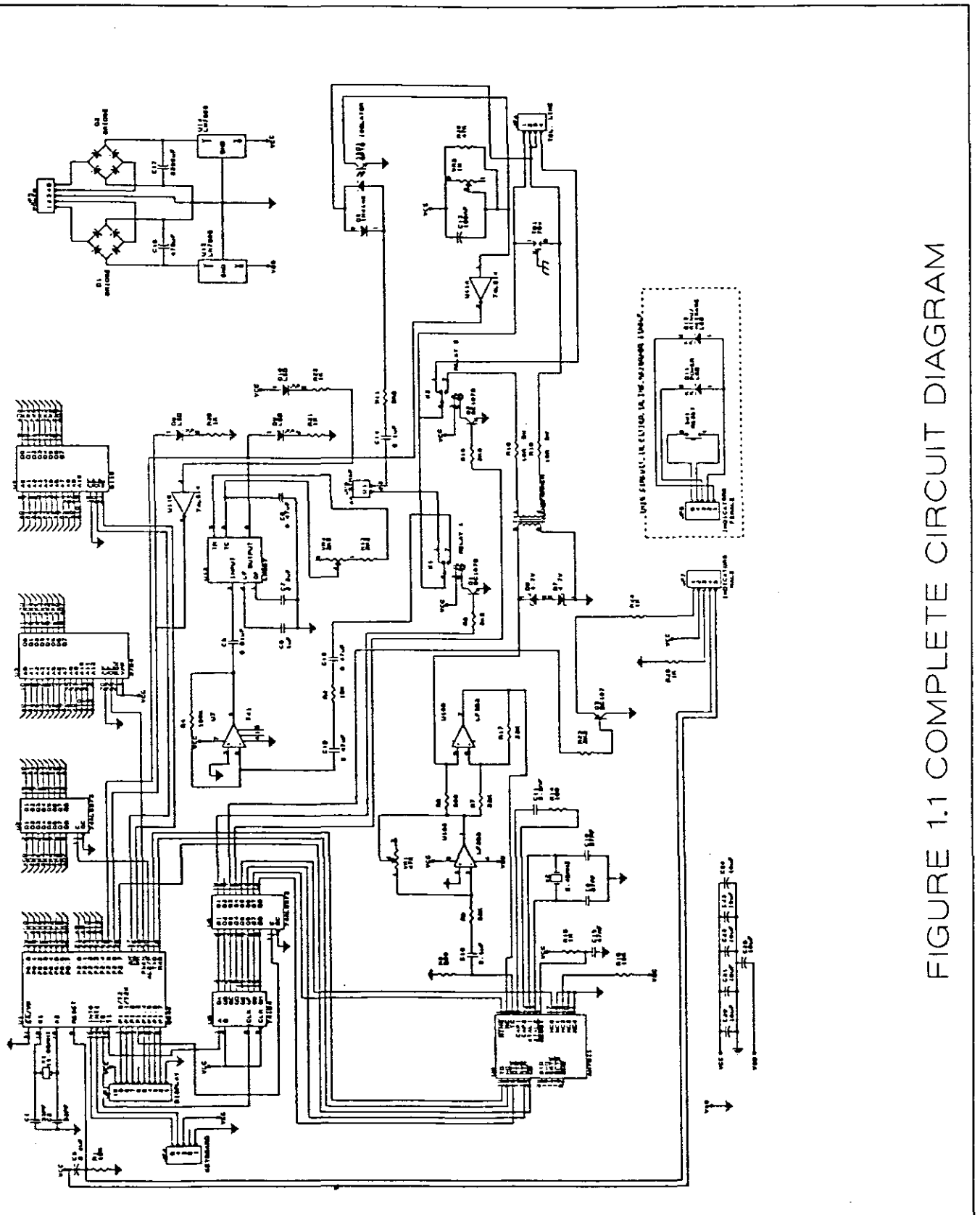


FIGURE 1.1 COMPLETE CIRCUIT DIAGRAM

2.0 OPERATING INSTRUCTIONS

Refer to figure 2.1 on page 5.

2.1 Installation procedure.

To install the Voiceless Telephone, unplug the telephone from the wall socket. Insert the male telephone jack that extends from the back of the unit into the telephone jack attached the wall.

Plug the telephone into the back of the unit. Finally, plug the mains extension into the unit. Before connecting the mains, ensure the power switch is off. Plug the mains plug into the wall socket and switch on. The following message (referred to as the LOGO) will then be displayed;

```
**** VOICELESS ****
**** TELEPHONE ****
```

2.2 Placing a call (F1).

To place a call, lift the telephone handset. Press F1 on the keyboard. The unit will then test the telephone line for dial tone. If no dial-tone is detected, the unit will respond with "No dial tone", followed by "Call cancelled" and return to the LOGO. Once dial-tone has been detected, the unit will respond with; "Dial tone detected.. Go ahead and dial". Dial the number on the telephone in the normal manner. NB: Do not pause for more than one second during dialing. The unit will then test the line and respond with one of the following messages:

ENGAGED - Remote telephone is busy.

NO ANSWER - Call has not been answered.

LINE FAULT - An unexpected condition has been detected.

Under all the above conditions, the unit will automatically cancel the call and reset itself.

CARRIER DETECTED followed by CONNECTED - This is a normal response. The unit will then automatically clear the screen and the cursor will flash in the top left hand corner indicating that it is ready to receive input from the keyboard. The telephone hand set may now be replaced.

2.3 Answering a call (F3).

An incoming call will be indicated by a flashing light (RING/MESS) on the keyboard if not in the automatic answer mode. To answer the call press F3. The unit will respond to the ringing by terminating the line and testing for carrier from the remote caller. If carrier is not detected, the call will automatically be cancelled and return to the LOGO. If carrier is detected, the following message will be displayed:

"Carrier detected" followed by "Connected".

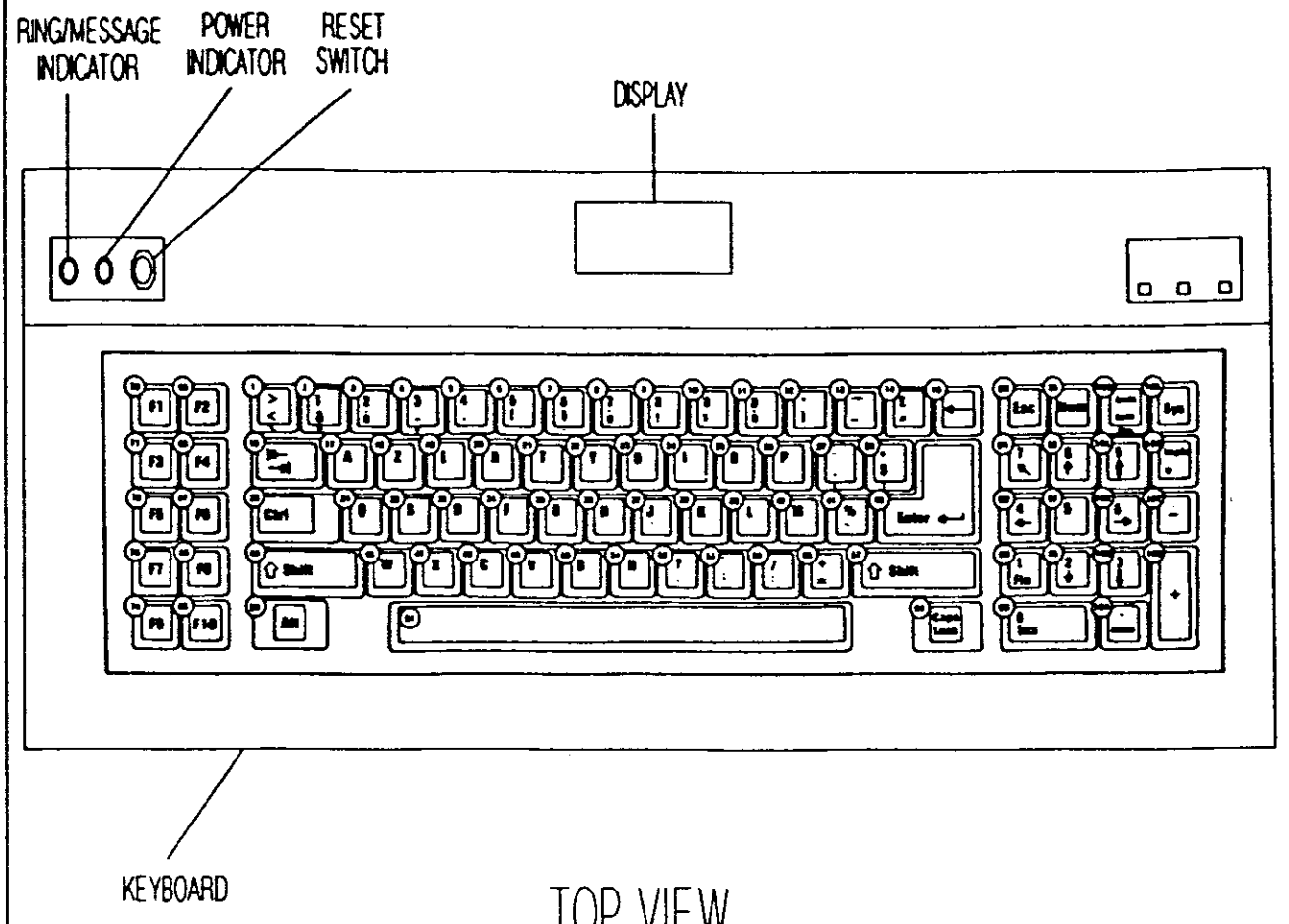
The screen will then automatically be cleared and the cursor will flash in the top left hand corner indicating that it is ready to receive input from the keyboard.

2.4 Invoking automatic answering (F2).

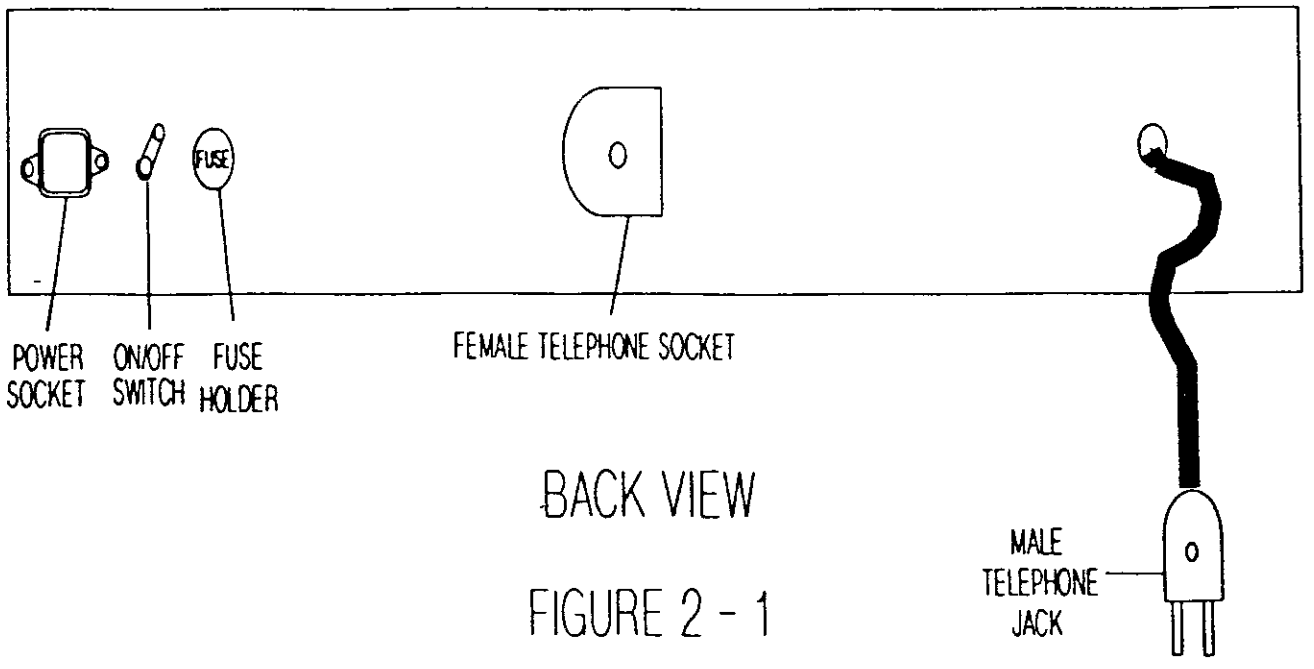
To place the unit into automatic answer mode, press F2. The following message will then be displayed;

"Auto answer enabled".

This will result in the call being automatically answered after three rings. The rest of the procedure will be the same as in the case of manually answering the call.



TOP VIEW



BACK VIEW

FIGURE 2 - 1

NB: Auto answer will be cancelled after the first call has been answered.

To cancel the auto answer mode, press any key while the above message is being displayed. The unit will respond with:

"Auto answer disabled",

and return to the LOGO.

2.5 Transmitting and receiving messages.

Once the "CONNECTED" message has been displayed, and the screen cleared, the unit is ready to transmit and receive messages. To enter a message, type on the keyboard as if it was a standard typewriter or computer keyboard except that automatic line-feed has been incorporated. NB: Do not use ENTER/<CR> for this.

To transmit a message press ENTER. A message indicating a transmission will then be displayed.

If a message has been received, the RING/MESS. light on the keyboard will glow. This will remain illuminated until such time as the message has been retrieved from its memory. To retrieve the message, press F1. The received message will then scroll across the screen in single-line mode. To read the received message while editing a message, simply press F1 and after displaying the received message, the edited message will be re-displayed.

2.6 Terminating a call.

F10 is used to terminate a call during the connected stage and returns the unit to the idle state. Ensure that the telephone handset is replaced, to prevent holding the local exchange line.

2.7 General.

The following is a list of all the keys used by the Voiceless Telephone.

- F1,F2,F3 and F10;
- CAPS LOCK, BACK SPACE and ENTER;
- All alpha-numerical characters;

The remainder have no function and if pressed will display,"Inactive key" while in the idle state, or a question mark while in the connect state.

Should any problem occur which prevents commands from being entered on the keyboard, the reset button can be used to return the unit to the power-up state, although this will result in the call; if in progress; being cancelled.

Do not attempt to dial a number on the telephone, unless the message;"Go ahead and dial",is being displayed.

The Voiceless Telephone has a transmit and receive buffer of 78 characters. Should this be exceeded, the message;"Buffer full..",will be displayed. By pressing ENTER, the stored message will be transmitted, and buffer cleared. Should either buffer overflow, any further

information being transmitted or received, will be lost. To prevent the receive buffer overflowing, retrieve the received messages as soon as possible, as this will clear the receive buffer ensuring the next message is correctly captured.

3.0 HARDWARE DESIGN

3.1 The Processor Unit

Refer to figure 3.1 for the processor and associated memory circuit diagram (Page 10).

3.1.1 The 8032 Processor.

An 8032 8-Bit control processor is used which provides on-chip support for one-bit variables as a separate data-type, allowing direct bit manipulation and testing. It also supports direct serial communication, which simplifies the modem interface.

3.1.2 I/O Ports.

The 8032 has 24 programmable I/O pins (three 8-bit ports). These ports have been configured as follows;

Port 0: Address/data bus.

Port 1: Display control and data.

Port 2: Remaining address lines(P2.0 - P2.4) and serial-to-parallel control inputs.

3.1.3 Memory.

The processor has two memory facilities at its disposal, selected using the PSEN line. The first is an 8Kbyte EPROM (2764) used to store the application software, the second is an 2 Kbyte RAM (6116) used to store temporary data.

3.1.4 Special Function Registers.

Refer to Appendix H.

The 8032 has several Special Function Registers (SFR's) which contain control and status bits for the interrupt system, timers/counters and serial port.

The SFR's used and their configuration are described below.

SCON - The serial port control register is used to set the mode of operation, enable reception, and contains the ninth data bit and interrupt flags for transmission and reception. Setting SCON to 50H during the initialisation (init) routine, selects mode 1 (8-bit UART operation) and enables serial reception.

TMOD - The timer/counter mode control register is used to select the mode of operation of the two 16-bit timer/counter registers, configure as either a timer or counter, and provide gating control for timer 0 and 1. Initialising this register to a value of 20H, selects mode 2 (8-bit auto-reload) for timer 1 and mode 0 for timer 0.

TCON - The Timer/Counter Control Register is used to control the run control bit and overflow flag for timer 0 and 1 and the external interrupts 0 and 1. Initialising this register to a value of C1H, turns timer 1 on and enables its overflow flag, sets interrupt 1 to low-level triggered and interrupt 0 to edge triggered.

IE - The Interrupt Enable Register controls the following interrupts;

- External interrupt 0 and 1,
- the serial port interrupt and,
- the timer overflow interrupts.

Each of these interrupt sources can be individually enabled or disabled by setting or

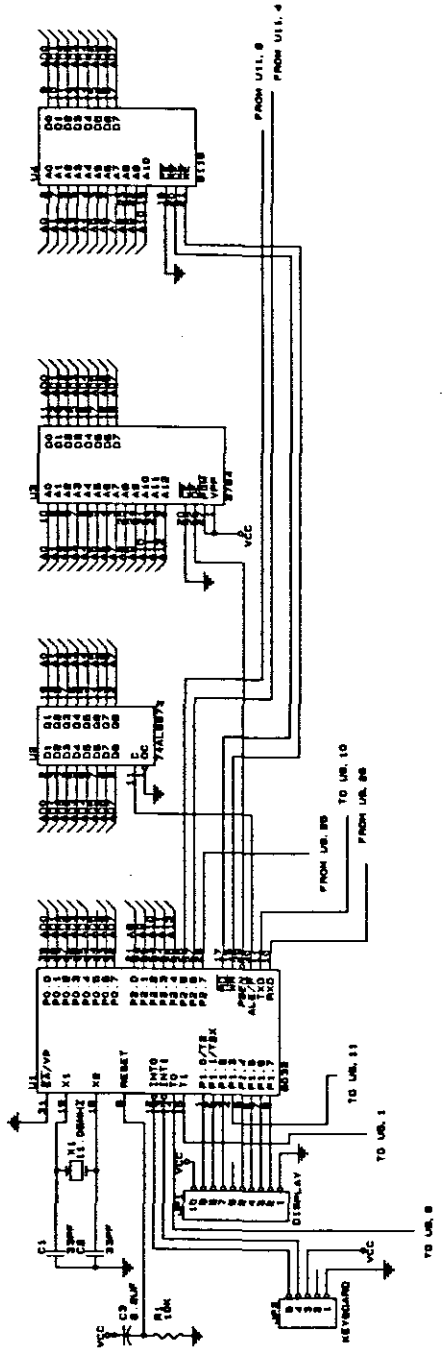


FIGURE 3.1 THE PROCESSOR

clearing a bit in the IE register. The EA bit provides a global disabling facility which disables all interrupts at once.

3.1.5 Serial Interface.

The serial port can transmit and receive simultaneously (full duplex). The transmit and receive registers are both accessed at the special function register, SBUF. Writing to SBUF loads the transmit register and reading SBUF accesses a physically separate receive register. Operating in mode 1 (defined in SCON), 10 bits are transmitted through TXD or received through RXD. The 10 bits are made up of a start bit (always 0), 8 data bits (LSB first), and a stop bit (always 1). The baud rate in this mode is variable and is generated by Timer 1. The baud rate is calculated as follows;

$$\text{Baud Rate} = \frac{1}{16} \times (\text{Timer 1 overflow rate})$$

Since a machine cycle consists of 12 oscillations, the count rate is 1/12 the oscillator frequency.

$$\text{Baud Rate} = \frac{1}{16} \times \frac{\text{Oscillator frequency}}{12\ (256 - \text{TH1})}$$

$$= \frac{1}{16} \times \frac{11.059\ \text{Mhz}}{12\ (256 - 80)}$$

$$= 327\ \text{Baud}$$

3.1.6 Specifications:

8032 Processor :	
Supply voltage	5 V
Supply current	175 mA
High-level input voltage	2 to 5.5 V
Low-level input voltage	-0.5 to 0.8 V
High-level output voltage	2.4 V
Low-level output voltage	0.45 V
Internal oscillating frequency	12 Mhz (Max)
Internal Ram	256 X 8
Timers/Counters	3 X 16 Bit
Interrupts	6
2764 Eprom:	
Supply Voltage	5 V
Supply current	100 mA (Max)
High-level input voltage	2 V
Low-level input voltage	0.8 V
High-level output voltage	2.4 V
Low-level output voltage	0.45 V
Memory Capacity	8K X 8
Maximum Access Time	200 nS
6116 Ram:	
Supply voltage	4.5 to 5.5 V
Supply current	35 mA
High-level input voltage	2 V
Low-level input voltage	0.8 V
High-level output voltage	2.4 V
Low-level output voltage	0.45 V
Max Address Access Time	200 nS
Memory Capacity	2K X 8

3. 2 THE MODEM

Refer to figure 3.2 page 13.

3.2.1 The EF7910 Modem Chip.

The EF7910 is an asynchronous Frequency Shift Keying (FSK) voice band modem. It has been used in the Voiceless Telephone for several reasons; namely:

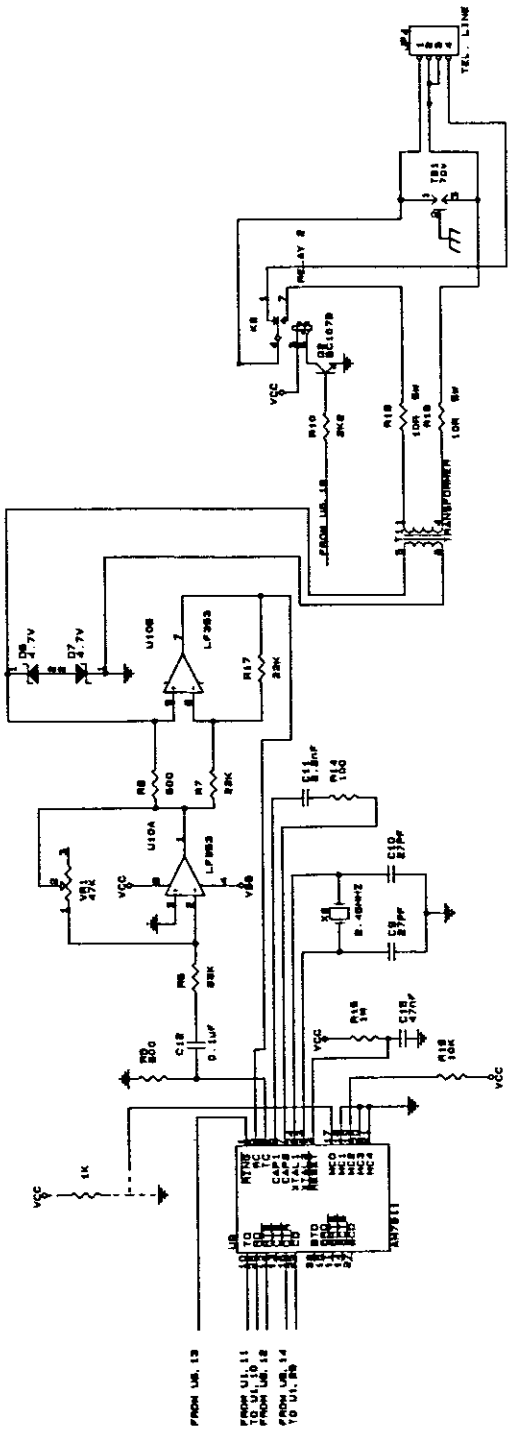


FIGURE 3.2 THE MODEM CIRCUIT

- no external filters are required;

- it has auto-answer capabilities;

- high speed was not a neccesity;

- and it is relatively cheap.

3.2.2 Modem configuration.

The EF7910 has thirty-two configuration option selected using MC0 - MC4 (control inputs). The units have been hard-wire strapped for CCITT V.21 full duplex 300 baud. Using this configuration, one unit has to be strapped in the answer mode, while the other is in originate mode. This mode does not determine which unit must originate or answer a call, it simply determines which frequency band is used by which unit. The V.21 protocol, stipulated by SAPT , specifies the various line frequencies as shown below. All 'backward' channel handshaking lines are ignored in this mode of operation.

PROTOCOL	TRANSMIT FREQUENCY		RECEIVE FREQUENCY		ANSWER TONE
	Space	Mark	Space	Mark	
CCITT V.21 Originate	1180	980	1850	1650	
CCITT V.21 Answer	1850	1650	1180	980	2100

LINE FREQUENCIES

TABLE 3 - 1

HANDSHAKING LINE	DESCRIPTION	INITIAL CONDITION
DATA TERMINAL READY ($\overline{DTR}$)	A LOW level on this input indicates that the unit is ready to send or receive data via the modem. A HIGH level on this input disables all TTL I/O pins and internal logic.	OFF (High)
REQUEST TO SEND ($\overline{RTS}$)	A LOW on this input places the modem in transmit mode. $\overline{RTS}$ must remain LOW for the duration of the transmission. A HIGH on this input turns off the transmitter.	OFF(High)
CLEAR TO SEND ($\overline{CTS}$)	This output goes LOW at the end of a delay(208mS) initiated when $\overline{RTS}$ went LOW. Data should not be present at the TD input until a LOW is indicated on the $\overline{CTS}$ output. $\overline{CTS}$ returns to a HIGH after a delay following $\overline{RTS}$ having gone HIGH.	OFF(High)
CARRIER DETECT ($\overline{CD}$)	A LOW on this output indicates a valid carrier signal is present at the receiver and has been present for at least 11.4 mS. $\overline{CD}$ remains HIGH while $\overline{DTR}$ is HIGH.	OFF(High)
RING ($\overline{RING}$)	This input is used to permit auto-answer capabilities by responding to a ringing signal from a DAA. If $\overline{DTR}$ is LOW and $\overline{RING}$ is made LOW, the modem begins a sequence to generate an answer tone at the TC output.	OFF(High)

TD - Transmit Data Input
TC - Transmitted Carrier Output
DAA - Data Access Arrangement
(Telephone Line Interface)

TABLE 3-2 MODEM HANDSHAKING LINES

3.2.3 Data Access Arrangement.

The data access arrangement provides an interface between the telephone line and modem. It comprises of an op-amp (LF353), and a matching transformer. The op-amp, functions as a duplexor by placing the transmitter output onto the line while receiving adjacent channel data from the line. The matching transformer (600 ohms impedance), inductively couples the line to the unit while remaining electrically insulated.

3.2.4 Modem Operation.

The handshaking lines are responsible for controlling the operation of the modem. Refer to table 3-2 for the function and initial condition of each line. These lines are in turn controlled by the processor. All the outputs from the modem and TD (Transmit Data) are connected directly to the processor while the inputs are driven via the serial-to-parallel converter. The control of these lines and ultimately the modem, will be discussed in section 4.

3.2.5 Component specifications:

EF7910 Modem Chip	
Supply voltage	+5 and -5 V
Supply current	125 mA
High-level output voltage	+2.4 to +5 V
Low-level output voltage	-0.5 to +0.4 V
High-level input voltage	+2.0 to +5 V
Low-level input voltage	-0.5 to +0.8 V
LF353 Op-amp	
Supply voltage	+ 18 V
Power Dissipation	500 mW
Common-Mode Rejection Ratio	70 - 100 dB
Gain Bandwidth Product	MHz
Large Signal Voltage Gain	25 - 100 V/mV

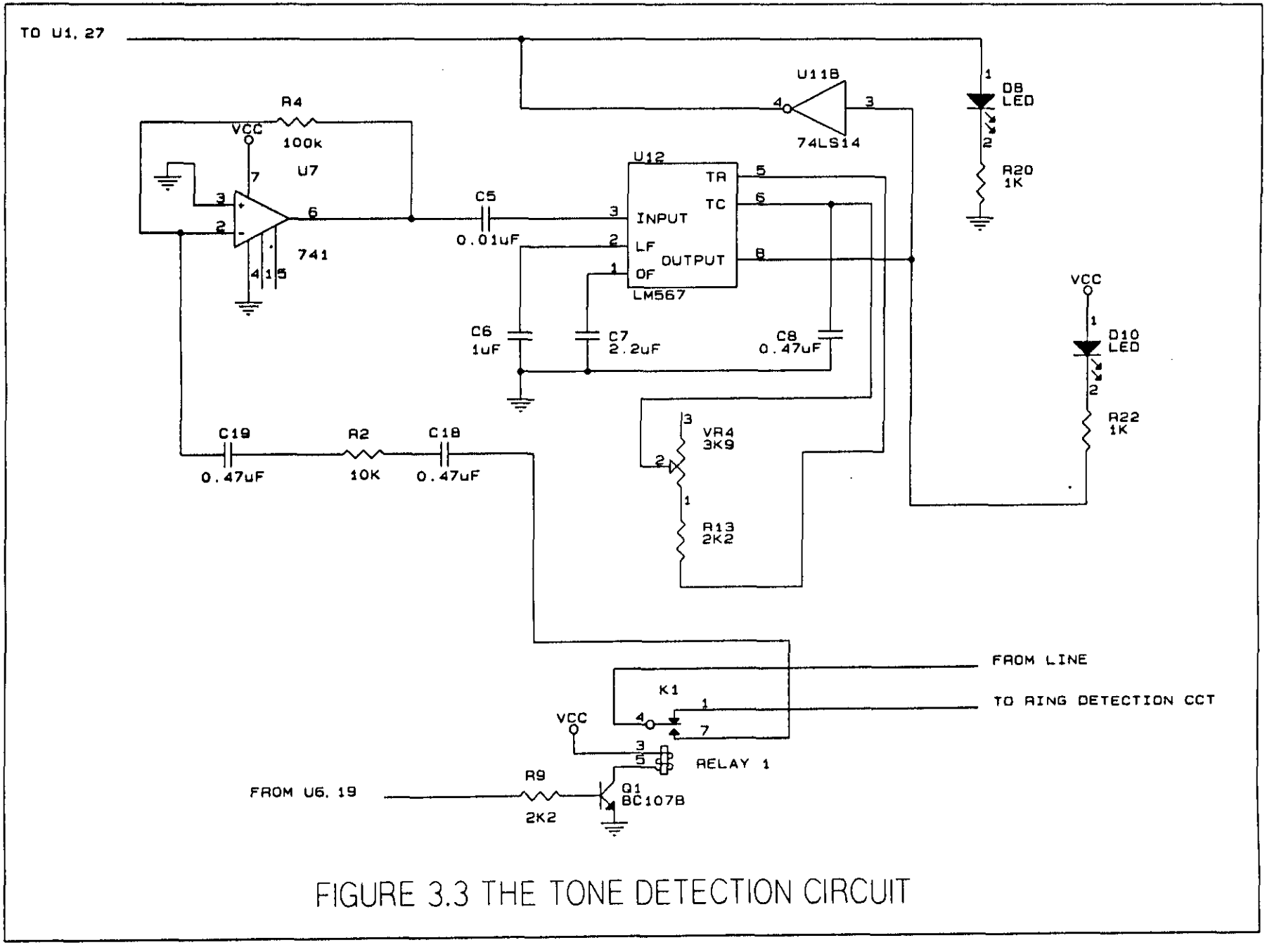


FIGURE 3.3 THE TONE DETECTION CIRCUIT

3.3 TONE DETECTION CIRCUITRY

Refer to figure 3.3 (page17).

3.3.1 Reasons for the Tone Detection Circuit.

The user of the Voiceless Telephone, will not be able to respond to the various tones generated on the telephone line. It is for this reason that a tone detection circuit had to be incorporated into the unit.

The tones generated on a standard telephone line (eg. engaged-tone, ring-tone, etc.) all consist of burst of a 400Hz tone with the exception of dial tone. Dial-tone is a 400 Hz tone modulated at 33Hz. Table 3 - 3 gives details of the various tones.

TONE	ON	OFF	ON	OFF
DIAL TONE	Continuously ON (33Hz mod.)			
RING	400mS	200mS	400mS	2S
NU (Number Unavailable)	250mS	500mS	250mS	500mS
ROUTE BUSY	250mS	250mS	250mS	250mS
SUBSCRIBER BUSY	500mS	500mS	500mS	500mS

Note: Route busy and subscriber busy, both have a 50% duty cycle and result in an engaged signal.

TELEPHONE TONES DESCRIPTION TABLE

TABLE 3 - 3

The tone detection circuit comprises of three main elements, namely an amplifier, tone decoder and sampler.

3.3.2 The Amplifier.

The amplifier makes use of the LM741 general purpose operational amplifier configured as a basic inverting amplifier. The following features make it ideal for this application;

- Overload protection on the input and output;
- non latch-up when common mode range is exceeded and;
- freedom from oscillation.

The op-amp is used not only for amplification, but to convert the analog line signal into a square wave. This improves the performance of the tone decoder. Gain of the amplifier is calculated as follows:

$$\text{GAIN} = -\frac{\text{FEEDBACK RESISTANCE}}{\text{INPUT RESISTANCE}}$$

3.3.3 The Tone Decoder.

The tone decoder (LM567) provides a saturated transistor switch to ground when an input signal is present within the bandwidth. The centre frequency is set equal to the free running frequency of the internal VCO. This is set using C8, R12 and VR4 for calibration. The centre frequency is calculated as follows:

$$fo = \frac{1}{1.1 \times C8 \times R(total)}$$

$$= \frac{1}{1.1 \times 0.47\mu F \times 4K8}$$

$$= 398\ Hz$$

The centre frequency can be adjusted from 0.01Hz to 500 Khz and has a stability equal to

35+-60 ppm per degree Celsius within its operating temperature. The bandwidth of the filter determined using the following calculation:

$$BW = 1070 \sqrt{\frac{Vi}{fo \times C6}} \quad \text{in \% } fo$$

where Vi = input voltage (RMS), Vi ≤ 200 mV
C6 = Capacitance at Pin 2 (μF)

NB: A pull-up resistor is necessary on the output of this device.

3.3.4 Sampling Method.

The output of the tone decoder is fed via a Schmitt-trigger inverter to the processor which is responsible for sampling this output. The sampling is achieved by making 20 samples in 2 seconds and counting the number of high values recorded. The comments on the program listing below are self explanatory. Further explanation, however, will be provided in section 4.1.13.

```
void tonetst()
{
    v=0;                          /* Reset the variable 'v'to zero*/
    latch(0x00);                  /* Clear the display            */
    latch(0x01);                  /* Set cursor                   */
    lcd(&txt10);                  /* Display 'Testing line'       */
    serial=(serial¦0x80);         /* Switch TDC on to line        */
    control(serial);
    for (g=0;g<20;g++)            /* Sample 20 times              */
        {
        set_bit(TONE);            /* Convert I/O line to input    */
        for (i=0;i<25;i++)        /* Delay                        */
                delay(10);
        z=input(P2);              /* Sample port                  */
        z=(z&0x40);               /* AND input with 40H           */
        if(z!=0)                  /* If sample is high,           */
            {
            v=v+1;                /* increment the variable 'V'   */
            }
```

```
        for(i=0;i<23;i++)              /* Delay for next sample        */
            delay(10);
        }
    delay(100);
    serial=(serial&0x7f);              /* Switch TDC out of line       */
    control(serial);
}
 * TDC = Tone Detection Circuit
```

3.3.5 Component specifications:

LM741

Supply voltage	+-22 V
Input voltage range	+-13 V
Input resistance	2 MΩ
Power dissipation	500 mW
Differential Input voltage	+-30 V

LM567

Supply voltage	4.75 to 9.0V
Power dissipation	300 mW
Input resistance	15-25 KΩ
Bandwidth	12-16% *fo*
Smallest detectable input voltage	20mV RMS
Fastest ON-OFF cycling rate	*fo*/20

3.4 RING/DIAL DETECTION CIRCUIT

Refer to figure 3.4 (Page 22).

3.4.1 Functions of the Ring Detection Circuit.

The ring/dial detection circuit which is permanently connected to the line, has two functions. The first is to detect ringing current, and the other is to detect dial pulses. In both cases, pulses are applied to the processor via a Schmidt trigger inverter. The processor discriminates between the two depending on whether a call is being initiated or received.

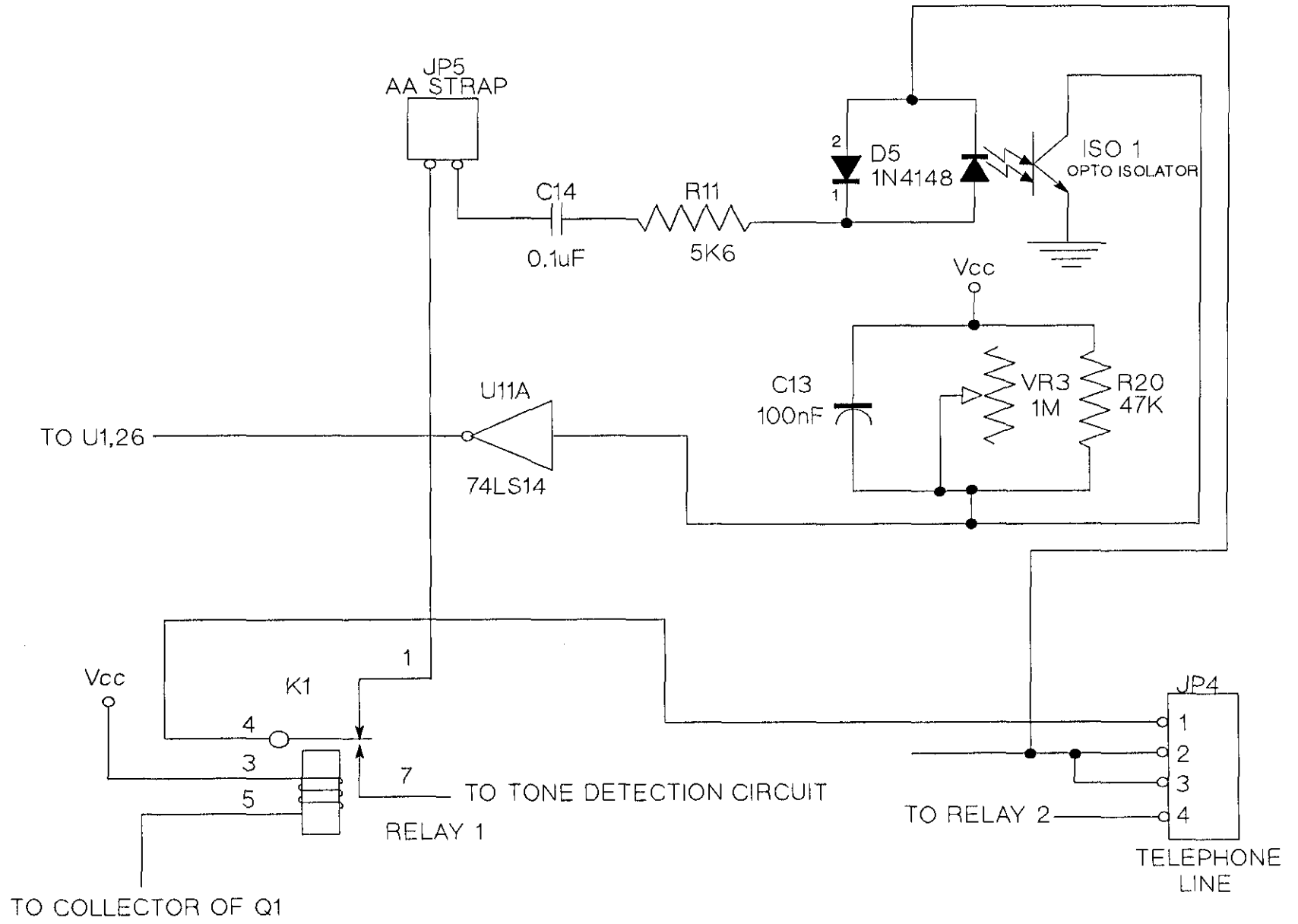


FIGURE 3.4 THE RING DETECTION CIRCUIT

3.4.2 Operation of the Ring Detection Circuit.

The detection circuit is made up of an opto-coupler and tank circuit, which is capacitively coupled to the telephone line. The opto-coupler isolates the 50V DC exchange voltage from the unit while the tank circuit extends the switch on time of the opto-coupler. R11 is a current limiting resistor and the header, JP6, was originally incorporated to manually disable automatic answering.

3.6.3 Component specifications:

74LS14	
Supply Voltage	5+-0.5 V
Supply Current	60 mA (Max)
High-level Input Voltage	1.7 V
Low-level Input Voltage	0.9 V
High-level Output Voltage	3.4 V
Low-level Output Voltage	0.2 V
Propagation Delay	15 nS
4N25 Optocoupler	
Isolation Voltage	2500 V
Input Diode Forward Voltage	1.5 V
Reverse Leakage Current	0.05 uA
Collector Output Current	5 mA
Emitter Collector Voltage	7 V
Power Dissipation	150 mW

3.5 SERIAL TO PARALLEL CONVERTER

Refer to figure 3.5 (Page 24).

3.5.1 Reasons for the STP converter.

The number of I/O lines on any processor is limited, the 8032 being no exception. After connecting the keyboard, memory and display to the processor, only three lines remained that could be utilised for I/O. This presented a problem as a further three were needed to control the handshaking signals of the modem.

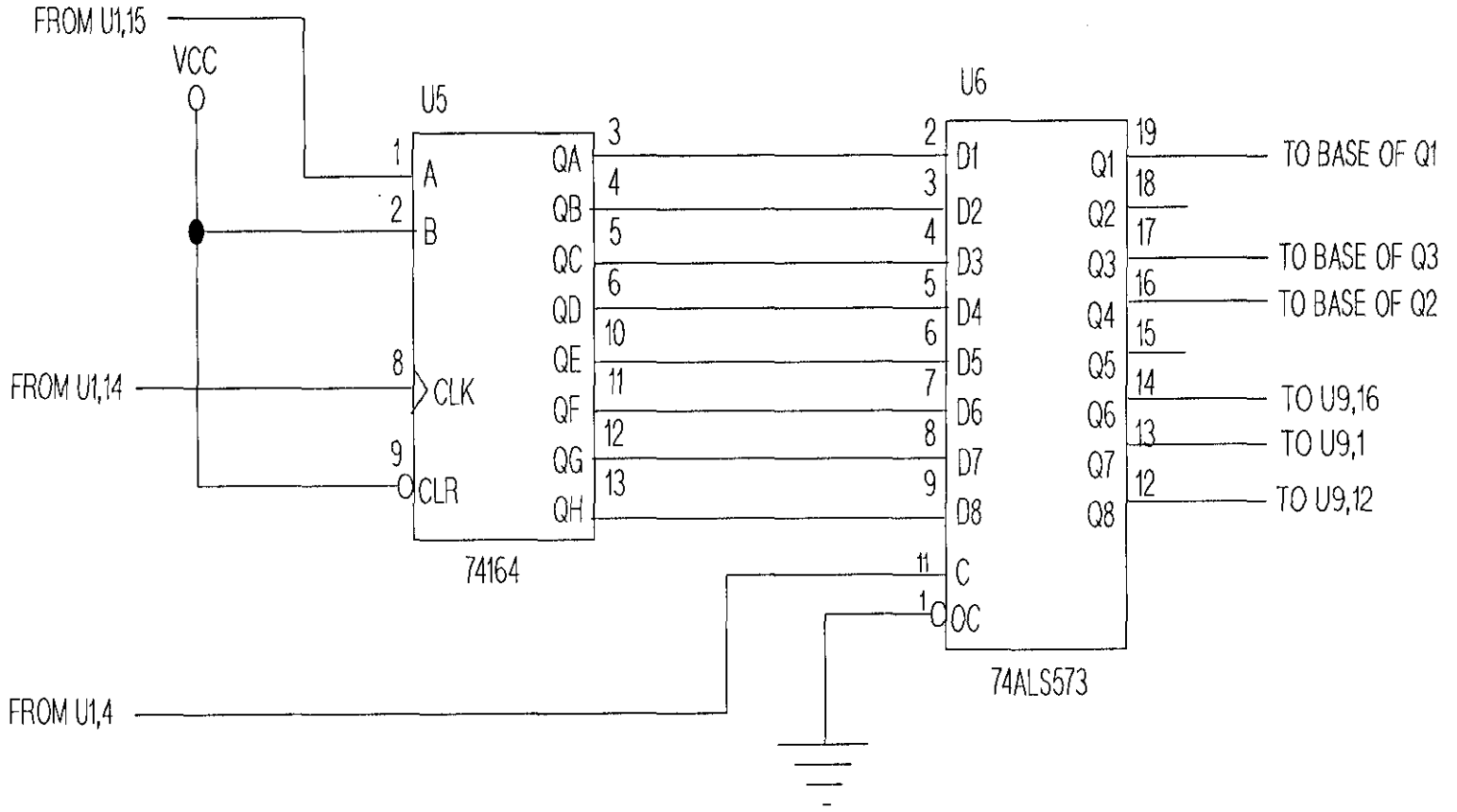


FIGURE 3.5 THE SERIAL-TO-PARRELLEL CONVERTER

3.5.2 Mode of operation.

The simplest method of overcoming this problem was to incorporate a serial-to-parallel converter with latched outputs.

The 74LS164 8-bit Parallel-out Serial-in Shift Register was used in conjunction with the 74LS573 Octal D-Type latch to obtain this.

3.5.3 Software control.

The three remaining I/O lines were defined in the software as follows;

```
#define  ENABLE P1_3_bit      /*  Define enable bit     */
#define  CLK  P3_4_bit        /*  Define clocking bit   */
#define  DATA P3_5_bit        /*  Define data bit       */
```

The subroutine (VOID CONTROL listed below) is then responsible for converting the hex value supplied by the main program, into a valid latched output, which in turn controls the operation of the relays, RING/MESS. indicator and handshaking signals of the modem.

```
void control(unsigned char serial)  /*   Output control byte  */
{
    for (w=0;w<=7;w++)              /* Repeat process eight   */
        {                           /*         times          */
        h=serial;                   /* Set serial equal to h  */
        h=(h&0x01);                 /*   And h with 01H       */
        if (h==0x00)                /* If h=0, clear data bit */
            {
            clear_bit(DATA);
            }
        else                        /*    else, set data bit  */
            {
            set_bit(DATA);
            }
#   delay(1);                      /*        Delay           */
    set_bit(CLK);                   /*  Clock serial to       */
##  delay(1);
    clear_bit(CLK);                 /*   parallel converter   */
    serial=(serial>>1);             /*    Shift serial right  */
    }
    set_bit(ENABLE);                /*  Latch output of serial */
### delay(1);
    clear_bit(ENABLE);              /*  to parallel converter  */
}
```

Note:
Delay allows steady state to be reached before clocking.

Delay increases clock pulse width.

Delay increases latch pulse width.

Further details pertaining to this sub-routine can be found in Section 4.1.7.

3.5.4 Component specifications:

74LS164

Supply voltage	4.5 to 5.5	V
Supply current	30 to 54	mA
High-level input voltage	2	V
Low-level input voltage	0.8	V
Maximum clock frequency	36	Mhz

74LS573

Supply voltage	4.5 TO 5.5	V
Supply current	24 TO 40	mA
High-level input voltage	2	V
Low-level input voltage	0.8	V
Minimum LE Pulse Width	15	nS

3.6 THE KEYBOARD

Refer to figure 3.6 (Page 27)

3.6.1 Interfacing the Keyboard.

A standard IBM compatible 101-keyboard, in the AT mode, has been used for the Voiceless Telephone. The keyboard derives its power from the main power supply of the unit and the 'clock' and 'data' lines are connected directly to the external interrupts 0 and 1 of the processor.

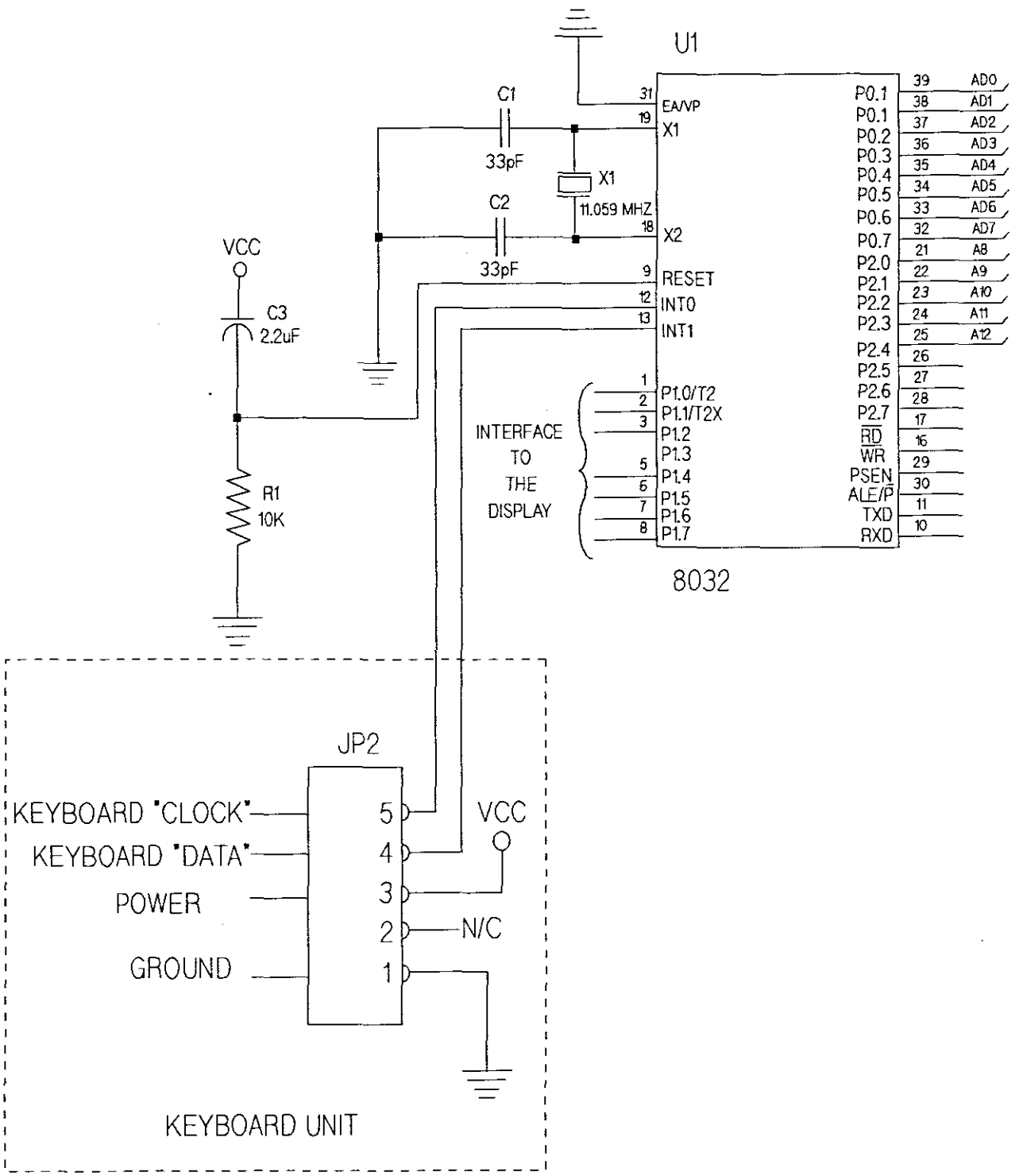


FIGURE 3.6 INTERFACING THE KEYBOARD

3.6.2 The IBM Keyboard.

On power-up, the LEDs on the keyboard will switch on for 300 to 500 milliseconds, while the BAT (Basic Assurance Test) is being conducted. This test is internal to the keyboard and has no effect on the rest of the unit. The keyboard makes use of scan codes to determine which key has been pressed.

The IBM keyboard supports three different scan codes. Scan code set 2 was chosen as this is the default setting and prevents having to set-up the keyboard on power-up.
In scan code set 2, each key is assigned a unique 8-bit make scan code, which is sent when the key is pressed. Each key also sends a break code when the key is released. The break code consists of 2 bytes, the first of which is the break code prefix, hex F0; the second byte is the same as the make scan code for that key.
Refer to Table 3-4 (page 29) for a list of scan codes (set 2) and figure 3.7 (page 30) for the layout of the keyboard and corresponding key numbers.

3.6.3 Keyboard Data Output.

The keyboard and processor communicate over the 'clock' and 'data' lines. When no communication is taking place, the 'data' line is held active (high) by the keyboard. During transmission, the 'clock' line provides the clocking signal used to latch the serial data from the keyboard into the processor. The data stream consists of 11 bits. Refer to table 3-5 for the function of each bit.

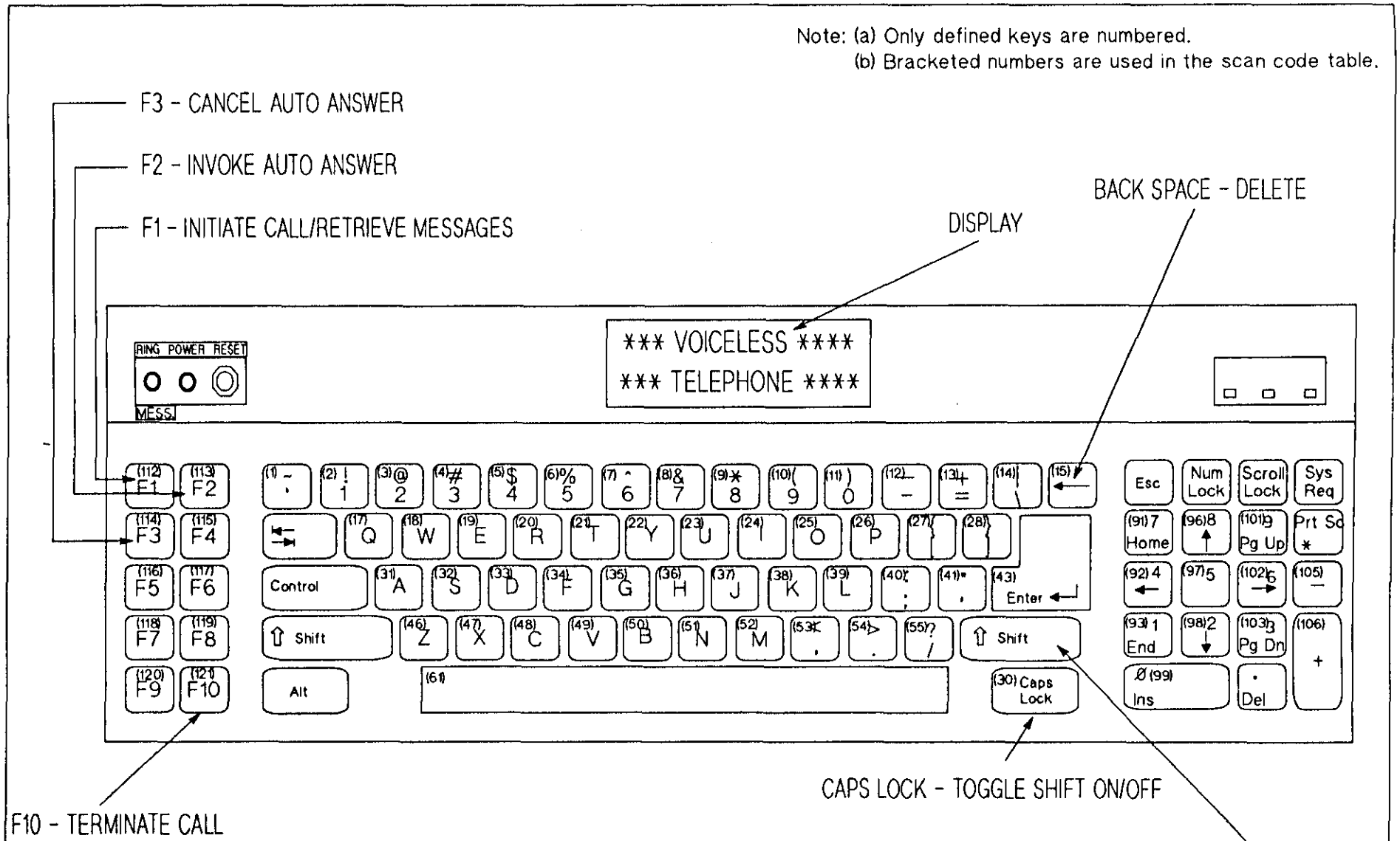
KEY NUMBER	MAKE CODE	BREAK CODE	KEY NUMBER	MAKE CODE	BREAK CODE
1	0E	F0 0E	66	--	-- --
2	16	F0 16	67	--	-- --
3	1E	F0 1E	68	--	-- --
4	26	F0 26	69	--	-- --
5	25	F0 25	70	--	-- --
6	2E	F0 2E	71	--	-- --
7	36	F0 36	72	--	-- --
8	3D	F0 3D	73	--	-- --
9	3E	F0 3E	74	--	-- --
10	46	F0 46	75	**	** **
11	45	F0 45	76	**	** **
12	4E	F0 4E	77	--	-- --
13	55	F0 55	78	--	-- --
14	--	-- --	79	**	** **
15	66	F0 66	80	**	** **
16	0D	F0 0D	81	**	** **
17	15	F0 15	82	--	-- --
18	1D	F0 1D	83	**	** **
19	24	F0 24	84	**	** **
20	2D	F0 2D	85	**	** **
21	2C	F0 2C	86	**	** **
22	35	F0 35	87	--	-- --
23	3C	F0 3C	88	--	-- --
24	43	F0 43	89	**	** **
25	44	F0 44	90	77	F0 77
26	4D	F0 4D	91	6C	F0 6C
27	54	F0 54	92	6B	F0 6B
28	5B	F0 5B	93	69	F0 69
29	5D	F0 5D	94	--	-- --
30	58	F0 58	95	--	-- --
31	1C	F0 1C	96	75	F0 75
32	1B	F0 1B	97	73	F0 73
33	23	F0 23	98	72	F0 72
34	2B	F0 2B	99	70	F0 70
35	34	F0 34	100	7C	F0 7C
36	33	F0 33	101	7D	F0 7D
37	3B	F0 3B	102	74	F0 74
38	42	F0 42	103	7A	F0 7A
39	4B	F0 4B	104	71	F0 71
40	4C	F0 4C	105	7B	F0 7B
41	52	F0 52	106	79	F0 79
42	5D	F0 5D	107	--	-- --
43	5A	F0 5A	108	E0 5A	E0 F0 5A
44	12	F0 12	109	--	-- --
45	61	F0 61	110	76	F0 76
46	1A	F0 1A	111	--	-- --
47	22	F0 22	112	05	F0 05
48	21	F0 21	113	06	F0 06
49	2A	F0 2A	114	04	F0 04
50	32	F0 32	115	0C	F0 0C
51	31	F0 31	116	03	F0 03
52	3A	F0 3A	117	0B	F0 0B
53	41	F0 41	118	83	F0 83
54	49	F0 49	119	0A	F0 0A
55	4A	F0 4A	120	01	F0 01
56	--	-- --	121	09	F0 09
57	59	F0 59	122	78	F0 78
58	14	F0 14	123	07	F0 07
59	--	-- --	124	--	-- --
60	11	F0 11	125	7E	F0 7E
61	29	F0 29			
62	E0 11	E0 F0 11			
63	--	-- --			
64	E0 14	E0 F0 14			
65	--	-- --			

** These keys send a code dependent on the state of the various shift keys and Num Lock. This feature is not used.

-- These keys are not used.

IBM KEYBOARD SCAN CODES (SET 2)

TABLE 3 - 4



THE KEYBOARD LAYOUT

FIGURE 3.7

Bit	Function
1	Start bit (always 0)
2	Data bit 0 (least significant)
3	Data bit 1
4	Date bit 2
5	Data bit 3
6	Data bit 4
7	Data bit 5
8	Data bit 6
9	Data bit 7 (most significant)
10	Parity bit (odd parity)
11	Stop bit (always 1)

BIT FUNCTIONS

TABLE 3 - 5

The parity bit is either 1 or 0, and the 8 data bits, plus the parity bit, always have an odd number of 1's (Odd parity).

The IBM keyboard supports various other commands which pass to and from the keyboard itself, but these features have not been utilised.

3.6.4 Keyboard Encoding.

The keyboard routine is divided into two section. The first section, written in assembler for timing purposes (refer to section 4.3), is responsible for reading the data stream from the keyboard, stripping off the break-code and then storing the following scan code in Timer 0 (high). A flowchart and associated explanation of this routine can be found in the software section.

The second section is responsible for converting the keyboard scan code, now in TH0, into the corresponding Ascii character. This is done by mapping the Ascii characters to the U.S. English keyboard layout.

Refer to source code below, and table 3-6 for the mapping table. An example of the above, is given in appendix I.

```
while(1)
  {
  if(read_bit_and_clear(F0_bit))        /*  On KB interrupt  */
        {
        output(IE,0x00);                /* Disable interrupts */
        clear_bit(RS);                  /* Select cont        */
        clear_bit(RW);                  /* Write to display   */
        c=input(TH0);                   /* Let c=KB input(TH0)*/
        chchk(c);                       /* Check for func. key*/
        if (s==1)                       /* If shift key -     */
        c=c+0x80;                       /* add 80H to "c"     */
        if (ignore==0)                  /* If non-function    */
              {                         /*          key -     */
              tbuf[tcnt]=dec[c];        /* Place input in     */
              tcnt++;                   /*  transmit buffer   */
              display(dec[c]);          /*    and display.    */
              scrnset();                /*   Set cursor       */
              if (tcnt==78)             /*    If buffer = 78  */
                    buffull();          /* indicate bufferfull*/
              }
        }

  }
```

The above routine adds hex 80H to the value of 'C' if the Caps Lock key has been previously pressed (s=1), and determines whether or not the key pressed was a special function key (eg.F1, F2, Back Space, etc.). If the key pressed was not a special function key, the character is displayed on the screen, written into the transmit buffer(TBUF), and the cursor is advanced. If the transmit buffer contains 78 characters, ie. the buffer is full, the message "Buffer full.."is displayed.

```
0x7f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,
0x3f,0x2c,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x71,0x31,0x3f,0x3f,0x32,
0x7a,0x73,0x61,0x77,0x32,0x3f,0x3f,0x63,0x78,0x64,0x65,0x34,0x33,
0x3f,0x3f,0x20,0x76,0x66,0x74,0x72,0x35,0x3f,0x3f,0x6e,0x62,0x68,
0x67,0x79,0x36,0x3f,0x3f,0x3f,0x6d,0x6a,0x75,0x37,0x38,0x3f,0x3f,
0x2c,0x6b,0x69,0x6f,0x30,0x39,0x58,0x3f,0x2e,0x2f,0x6c,0x3b,0x70,
0x5f,0x3f,0x3f,0x3f,0x2c,0x3f,0x5b,0x3d,0x3f,0x3f,0x3f,0x3f,0x2a,
0x5d,0x3f,0x3f,0x3f,0x3f,0x1f,0x3f,0x3f,0x1e,0x3f,0x3f,0x3f,0x3f,
0x3f,0x31,0x10,0x34,0x37,0x3f,0x3f,0x3f,0x30,0x3f,0x32,0x35,0x36,
0x38,0x3f,0x3f,0x3f,0x2b,0x33,0x2d,0x2a,0x39,0x3f,0x3f,0x00,0x3f,
0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,
0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x51,0x21,0x3f,0x3f,0x3f,0x5a,0x53,
0x41,0x57,0x40,0x3f,0x3f,0x43,0x58,0x44,0x45,0x24,0x23,0x3f,0x3f,
0x20,0x56,0x46,0x54,0x52,0x25,0x3f,0x3f,0x4e,0x42,0x48,0x47,0x59,
0x5e,0x3f,0x3f,0x20,0x4d,0x4a,0x55,0x26,0x2a,0x3f,0x3f,0x3c,0x4b,
0x49,0x4f,0x29,0x28,0x3f,0x3f,0x3e,0x3f,0x4c,0x3a,0x50,0x2d,0x3f,
0x3f,0x2c,0x22,0x69,0x7b,0x2b,0x39,0x3f,0x3f,0x3f,0x3f,0x7d,0x3b,
0x3f,0x2d,0x3f,0x3f,0x3f,0x2c,0x3f,0x5b,0x3d,0x3f,0x3f,0x3f,0x31,
0x34,0x34,0x37,0x3f,0x3f,0x3f,0x30,0x2e,0x32,0x35,0x36,0x38,0x3f,
0x3f,0x3f,0x2b,0x33,0x2d,0x2a,0x39,0x3f,0x3f,0x30,0x3f,0x32,0x35,
0x36,0x38,0x3f,0x3f,0x3f,0x2b,0x33,0x2d,0x2a,0x39,0x3f,0x3f,0x3f,
0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,
0x3f,0x3f
```

MAPPING TABLE

TABLE 3 - 6

If however, the key pressed was a special function key, control is handed to the character check subroutine (CHCHK). Details of this routine can be found in section 4.1.17.

3.6.5 Specifications:

Keyboard

Supply voltage	+4.5 to +5.5 V
Low-level(inactive) voltage	0 to +0.7 V
High-level(active) voltage	+2.4 to +5.5 V

3.7 LCD DISPLAY

3.7.1 THE LTN221 Liquid Crystal Display.

The LTN221 is a 5 X 7 dot, 20 character, 2-line dot matrix liquid crystal display module, with driver and controller LSI IC mounted on a single printed circuit board. The controller incorporates a ROM-based character generator (160 characters) and RAM display data with 80 characters. The module operates from an extensive instruction set. Refer to table 3-7 for the instruction set and figure 3.8 for the interface connections.

3.7.2 Interfacing the Display.

The LCD display is connected to port 1 of the processor. P1.0, P1.1 and P1.2 provide the control signals while P1.4 to P1.7 act as the data bus (4-bit mode).

3.7.3 Display Control.

Several routines have been written to control the display, write to the display and obtain the status of the display. These are cover in detail in section 4.1.

3.7.4 Specifications:

LTN221 LCD Module

Supply voltage	4.75 to 5.25 V
Supply current(logic)	1.5 to 2.4 mA
High-level input voltage	2.2 to 5.0 V
Low-level input voltage	-0.3 to 0.6 V
High-level output voltage	2.4 (min) V
Low-level output voltage	0.4 (max) V
Internal oscillating frequency	250 Khz

CODES											
FUNCTION	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	REMARKS
CLEAR	0	0	0	0	0	0	0	0	0	1	Clear LCD and memory. Cursor returns home.
HOME	0	0	0	0	0	0	0	0	1	*	Cursor returns home but LCD and memory unchanged.
SET ENTRY MODE	0	0	0	0	0	0	0	1	I/D	S	I/D=1 Cursor moves right. I/D=0 cursor moves left. S=1 display moves in opposite direction and cursor stays.
ON/OFF CONTROL	0	0	0	0	0	0	1	D	C	B	D=1/0 for LCD on/off. C=1/0 for cursor on/off. B=1 for blinking cursor.
SHIFT	0	0	0	0	0	1	S/C	R/L	*	*	S/C=1/0 for display/cursor shift. R/L=1/0 for right/left shift.
FUNCTION SET	0	0	0	0	1	DL	N	F	*	*	DL= 1/0 for 8/4 bits. N=1/0 for 2/1 row/s. F= 1/0 for 5X10/5X7 dots.
CG RAM	0	0	0	1	(ADDRESS FOR CG RAM)						Set CG RAM address. R/W to character generator RAM after this command.
DD RAM	0	0	1	(DISPLAY DATA RAM ADDRESS)							Set DD RAM address. R/W to display data after this command.
STATUS	0	1	BF	(ADDRESS COUNTER)							BF=Busy flag. 1-Internally busy. 0-can accept instructions.
WRITE DATA	1	0	(DATA TO BE WRITTEN INTO MODULE)								Write into DD or CG RAM
READ DATA	1	1	(DATA READ FROM MODULE)								Read from DD or CG RAM.

TABLE 3 - 7 THE LCD INSTRUCTION SET

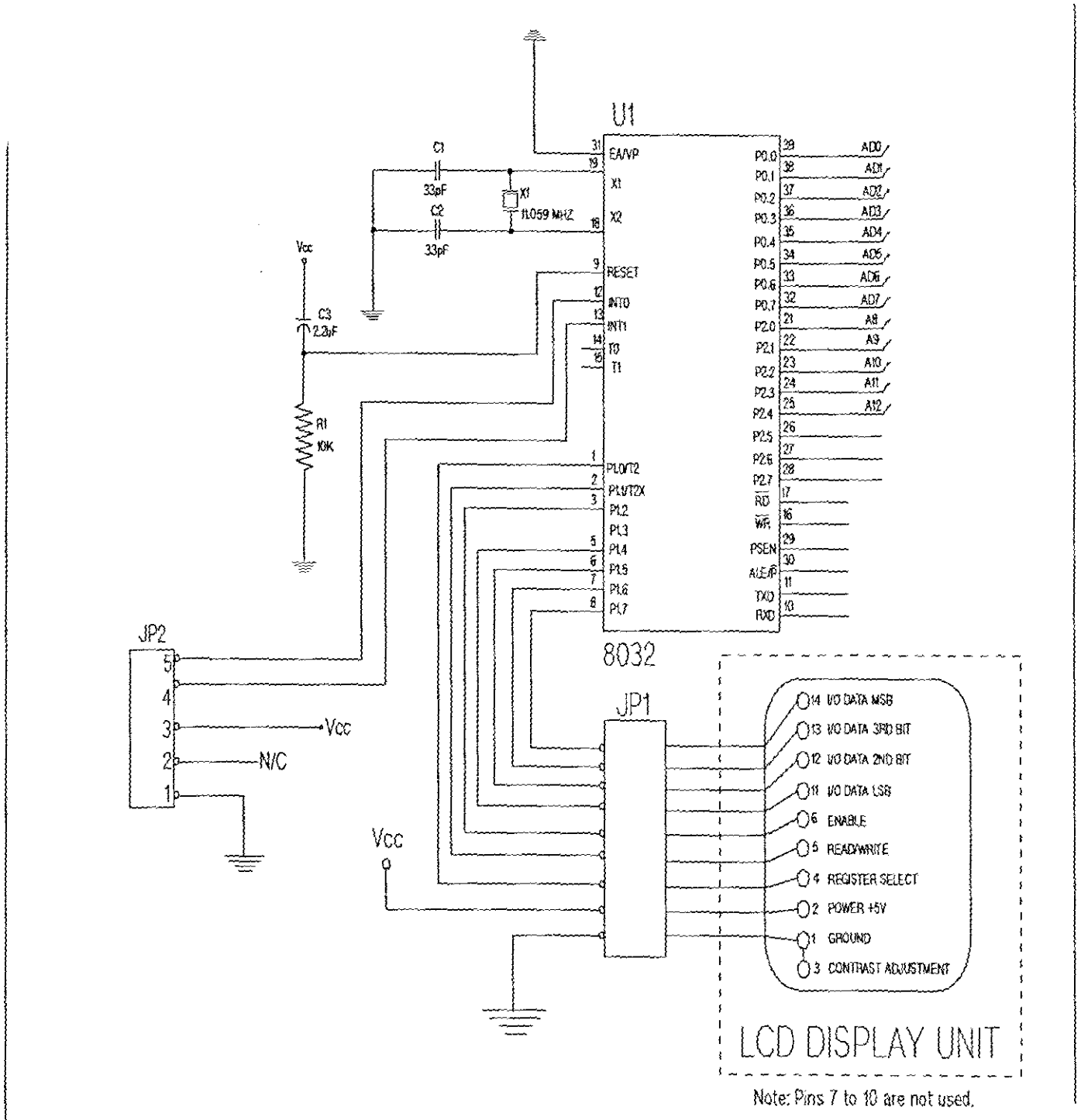


FIGURE 3.8 THE DISPLAY INTERFACE

3.8 THE POWER SUPPLY

3.8.1 The Supply Design.

The power supply for the Voiceless Telephone is a relatively simple one, consisting only of a 9 volt transformer with two windings, two bridge rectifiers, a positive and negative 5 volt regulator and the necessary smoothing capacitors. Further smoothing is provided by decoupling the more sensitive devices with 10uF capacitors.

3.8.2 Specifications:

Transformer

Voltage Rating	9 V (Dual tapping)
Current Rating	1 A

Bridge rectifiers

Maximum Voltage(RMS)	50 V
Maximum Current	3 A

LM7805 Positive Regulator

Maximum Input Voltage	35 V
Output Voltage	5 V
Supply Current	1 A
Output Noise Voltage	40 uV

LM7905 Negative Regulator

Maximum Input Voltage	-35 V
Output Voltage	-5 V
Supply Current	1 A
Output Noise Voltage	150 uV

4.0 SOFTWARE DEVELOPMENT

The application program was written in 'C' with the exception of the keyboard routine. This was written in assembler using the ASM51 compiler to reduce the execution time. A explanation of each module follows.

4.1 Program modules.

4.1.1 Main routine. (Refer to page 67 for routine listing and flowchart on page 39.)

As with all 'C' programs, program execution starts at the MAIN routine. The purpose of this routine is initialise the unit from power-up, and then display the LOGO while testing for a keyboard interrupt and ringing tone. Should a key have been pressed, the unit will clear the display and depending on which key was pressed, respond as follows.

(1) F1 pressed: Disable keyboard and pass control to the outgoing call routine (OGCALL).

(2) F2 pressed: Display "Auto answer enabled" and test for ringing. If any key is pressed at this stage, auto answer will be disabled.

(3) F3 pressed: Disable keyboard and pass control to the automatic answering routine (AUTOA).

Any other key: Disable keyboard, display "Inactive key" and return to the MAIN routine.

On returning from any of the above routines, the keyboard is re- enabled, and the LOGO displayed.

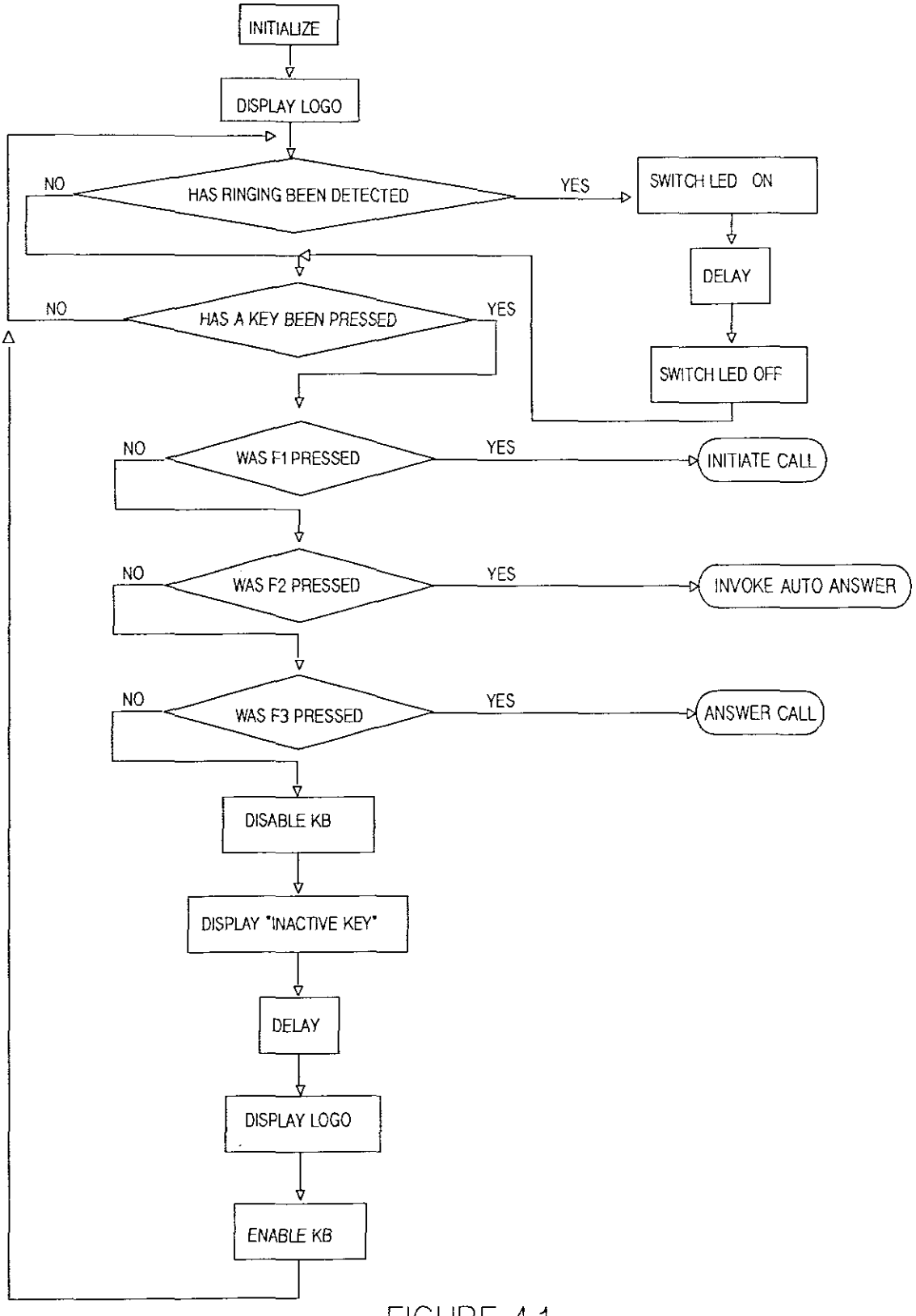


FIGURE 4.1

FLOWCHART - MAIN ROUTINE

4.1.2 LCD routine. (Refer to page 53 for routine listing.)

This routine is used in conjunction with the display routine to display the text, specified in parenthesis, on the screen.

4.1.3 Get_status routine. (Refer to page 54 for program listing.)

This routine is used to determine the status of the display (ie. read the busy flag). This is achieved by setting the RW bit, clearing the RS bit and then latching the instruction register of the display in two nibbles (4-bit mode). The most significant bit acts as the busy flag (1 = busy, 0 = ready). Depending on the value of the busy flag, the routine returns a 1 (if busy) or 0 (if ready) to the application program.

4.1.4 Delay routine. (Refer to page 54 for program listing.)

This routine generates a delay, the length of which is specified by the variable 'c'.

4.1.5 Outc routine. (Refer to page 54 for routine listing.)

This routine converts the control nibble passed on by the application program from the four least significant data lines to the four most significant and then latches it out to the display using the enable (E) bit.

4.1.6 Outd routine. (Refer to page 54 for routine listing.)

This routine is similar to the OUTC routine except that it first clears bit RW to indicate this is text data as opposed to control data. It converts the text nibble passed on by the

application program from the four least significant data lines to the four most significant and then latches it out to the display using the enable (E) bit.

4.1.7 Control routine. (Refer to page 55 for routine listing.)

This routine controls the latched output of the serial-to-parallel converter. This is done by determining the value of the least significant bit of the control byte, passed on by the application program (unsigned char serial), and then clocking in the appropriate 1 or 0 into the shift register. The control byte is then rotated making the second LSB the least significant bit and the process is repeated for all 8 bits of the control byte. On completion of this cycle, the output of the shift register is then latched to the peripherals using the ENABLE bit.

4.1.8 Init routine. (Refer to page 55 for routine listing).

This routine initialises the hardware and is responsible for the following:

- Disable all unnecessary interrupts;
- switching DTR,RTS and RING off;
- disconnecting the line;
- clearing the buffers (SBUF,RBUF and TBUF);
- setting the baud-rate and control registers;
- setting the display to 4-bit mode;
- switching on the display and cursor; and
- positioning the cursor (top row, first character).

Note: Before setting the display to 4-bit mode, it was found that one must first set it to 8-bit mode. No explanation for this is given in the documentation.

4.1.9 Display routine. (Refer to page 56 for routine listing.)

This routine checks the status of the display, and while it is not busy, converts the text data byte passed to it into two nibbles. It then uses the OUTD routine to write these two nibbles to the display (most significant nibble first).

4.1.10 Rint routine. (Refer to page 57 for routine listing.)

This routine acts as the receive interrupt. On reception, this routine writes the data received from the SBUF to the receive buffer (RBUF) and increments the receive counter (RCNT) until such time as a carriage return (0DH) is detected. The carriage return is used as an end of text (EOT) signal (refer to transmit section) and on detection sets the PX1 bit used in the TXRX routine.

4.1.11 Buffull routine. (Refer to page 57 for routine listing.)

This routine displays the message, "Buffer full..",on the opposite line to which the user is busy, whenever the transmit buffer (TBUF) is full. The buffer has been declared to have a length of 78 characters, and any further characters will be lost once this range has been exceeded.

4.1.12 Logo routine. (Refer to page 57 for routine listing.)

This routine simply displays the unit's LOGO, one line at a time.

```
**** VOICELESS *****
**** TELEPHONE *****
```

4.1.13 Tonetst routine. (Refer to page 58 for routine listing.)

This routine is used to sample the output of the tone detection circuit and return a value representative of a particular tone, eg. dial tone, engaged tone, etc. This is achieved by testing the value of the TONE bit 20 times, at regular internals, and incrementing a variable 'v' if high. The tones and corresponding values are shown below.

Dial tone - v>=13	Ring tone - v>=1 and v<=7
Line fault- v=0	Engaged - V>=8 and v <=12

4.1.14 Ogcall routine. (Refer to page 58 for routine listing and flowchart on page 44/45.)

This routine is responsible for the initial stages of placing an outgoing call (Refer to figure 4.2). It starts by displaying the message; "Initiating call", runs the routine TONETST, and depending on the value returned (ie. whether dial tone was detected or not), passes control over to the CONT routine or displays, "No dial tone". If dial tone is not detected, the call is automatically cancelled and the unit is re-initialised to the power-up state.

4.1.15 Autoa routine. (Refer to page 59 for routine listing.)

This routine handles the automatic answering of an incoming call. It can be initialised by ring current when in the Auto Answer mode, or by pressing F3 while the telephone is ringing (manual answering). The first function in this routine is to connect the telephone line to the unit. This is followed by switching on DTR and RING handshaking signals on the modem which will start a sequence to generate an answer tone. After switching on RTS, a test for the carrier frequency is performed. If carrier is detected, control is passed

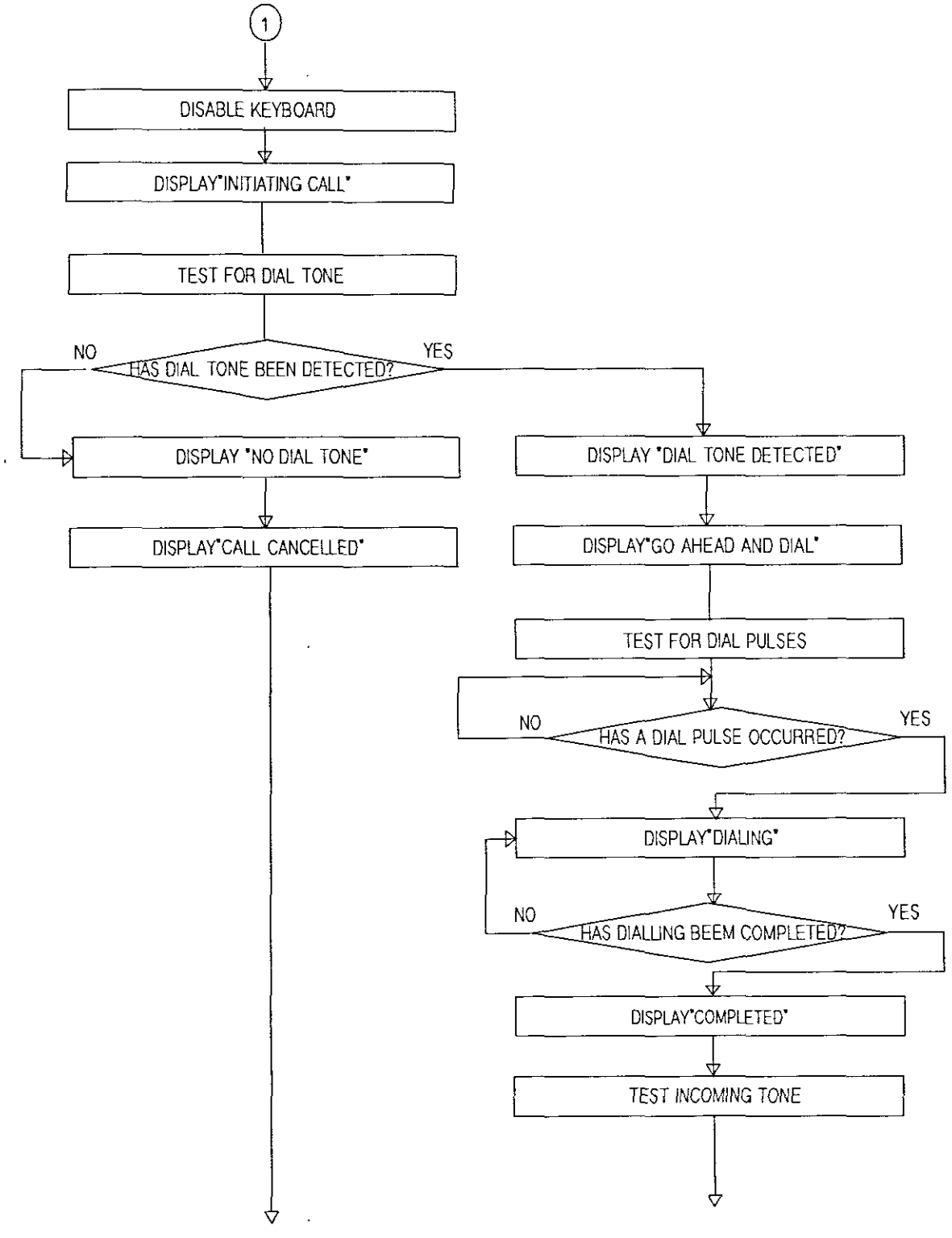


FIGURE 4.2 (a) FLOWCHART - OGCALL AND CONT ROUTINES

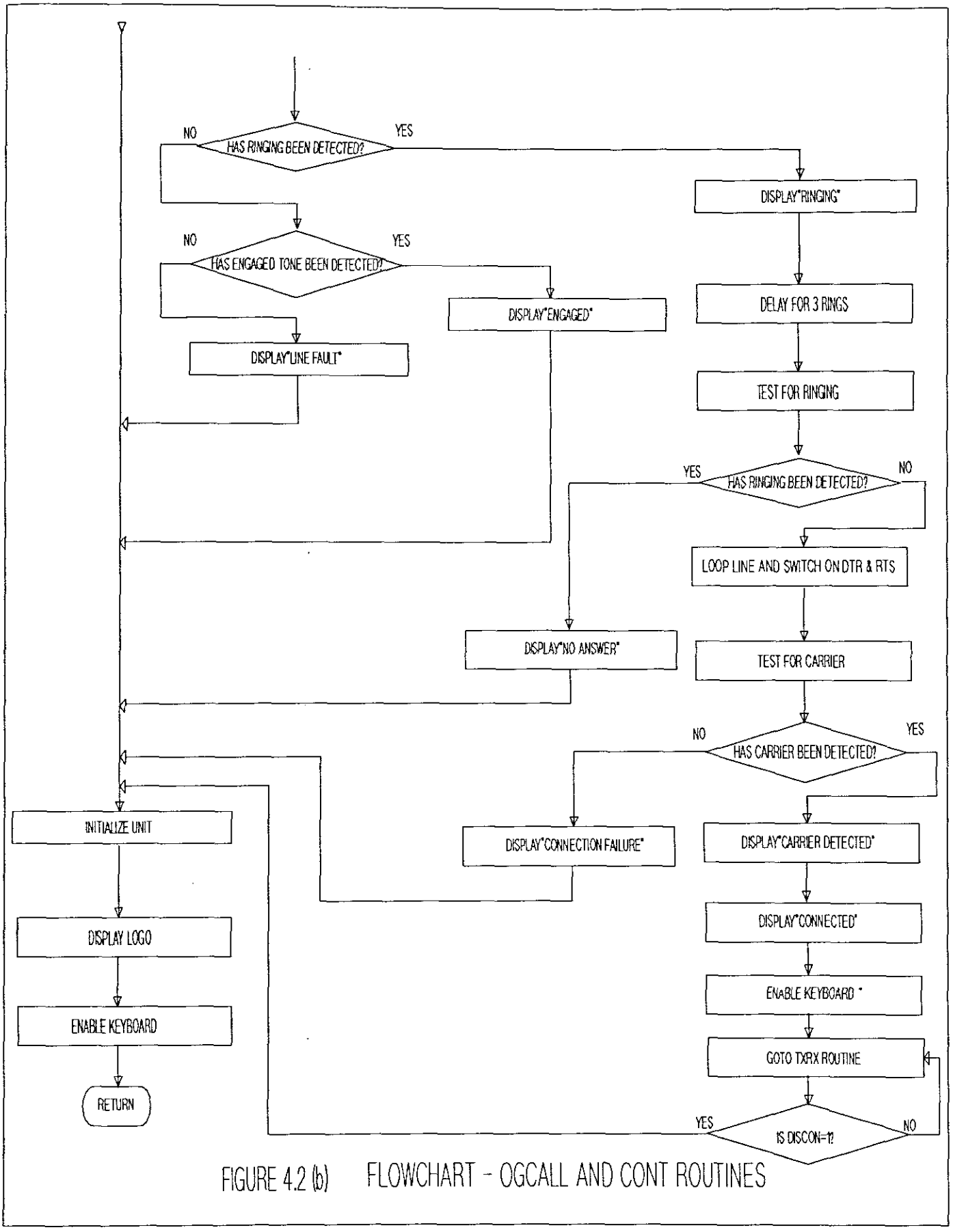


FIGURE 4.2 (b) FLOWCHART - OGCALL AND CONT ROUTINES

to the transmission (TXRX) routine after displaying, "Carrier detected" followed by ,"Connected". Should the carrier test fail, the unit will be re-initialised and the message, "Connection failure" will be displayed.

4.1.16 Scrnset routine. (Refer to page 60 for routine listing.)

This routine handles the wrap around feature of the display. This is achieved by displaying the bottom row on the top row and positioning the cursor at the beginning of the second row when the end of the bottom row is reached (chcnt=20). When the end of the top row is reached, the cursor is moved to the first position of the second row and the rowflag is set to 'b' (bottom).

4.1.17 Chchk routine. (Refer to page 61 for routine listing.)

This routine checks the input from the keyboard for special functions. Should a special function be detected, the IGNORE flag is set to one indicating that the character should not be displayed. The response to a special function differs according to the following. If;

- Caps Lock is pressed, it toggles the variable, 's', used in the transmission routine to display upper-case or lower-case characters.
- Enter is pressed, it causes the characters in the transmit buffer (TBUF) to be transmitted, and TBUF is cleared ready for the next message.
- F1 is pressed, the received message is displayed in single-line mode and rotated left until the entire message has been displayed. The MESS(AGE) LED is switched off and if present, the message being edited, is re-displayed.

- F10 is pressed, the variable 'DISCON' is set equal to one, which is used in the transmission routine to disconnect the line.

- Back Space is pressed, the previous character is deleted, and the cursor repositioned. How this is achieved, is determined by the position of the cursor at the time. Refer to the program listing for further detail.

4.1.18 Cont routine. (Refer to page 64 for routine listing and flowchart on page 44/45.)

This routine handles the out-going call once dial tone has been detected (Refer to figure 4.2). After displaying "Dial tone detected", the following message is displayed, "Go ahead and dial", on the second line. The program then performs a loop waiting for dial pulses. Once the first pulse is detected, a second loop is entered into, which will wait for the pulses to stop for two seconds. The program assumes that the dialing is then complete, and displays, "Dialing complete". The line is then tested for ringing. If ringing tone has been detected, execution is delayed for two seconds, and the line retested. Should ringing tone still be present, the program again assumes that the remote end has not answered the call, and the call is terminated after displaying, "No answer". If no ringing is detected, the modem is connected to line and the same sequence, as used in the AUTOA routine, is used to complete the connection.

Should the call be engaged, or a fault detected during this routine, the unit is re-initialised after displaying the appropriate informative message.

4.1.19 Txrx routine. (Refer to page 66 for routine listing and figure 4.3 on page 49.)

Once a call has successfully been established, control is passed to the transmission (TXRX) routine which handles message editing, transmission and reception. This routine consists of a loop which continuously tests for a keyboard and receive interrupt. On receiving a keyboard interrupt, the following checks and/or functions occur.

- The scan code is checked for special functions (see CHCHK routine).
- 80H is added to the scan code value if the Caps Lock key (s=1) has been pressed.
- The input character is stored in TBUF and displayed (ignore=0).
- The cursor is repositioned (see SCRNSET routine).
- The buffer is checked for overflow (see BUFFULL).
- Carrier from the remote unit is tested.

On receiving a message (the PX1 bit set), a control byte is sent to the serial to parallel converter, to switch the RING/MESS(AGE) LED on.

4.1.20 Latch routine. (Refer to page 55 for routine listing)

The LATCH routine provides a 150uS delay after each nibble of control data has been latched to the display using the OUTC routine.

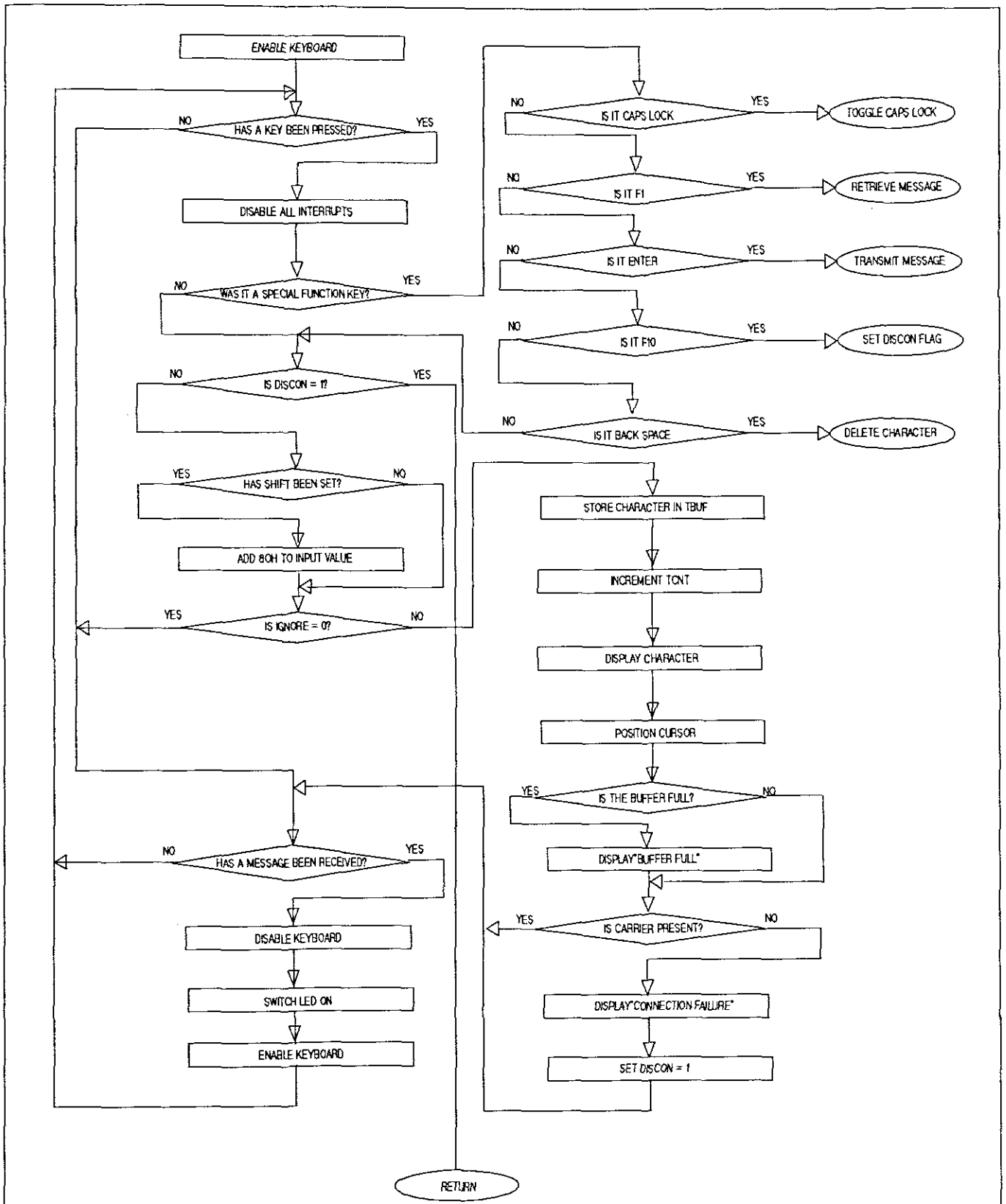
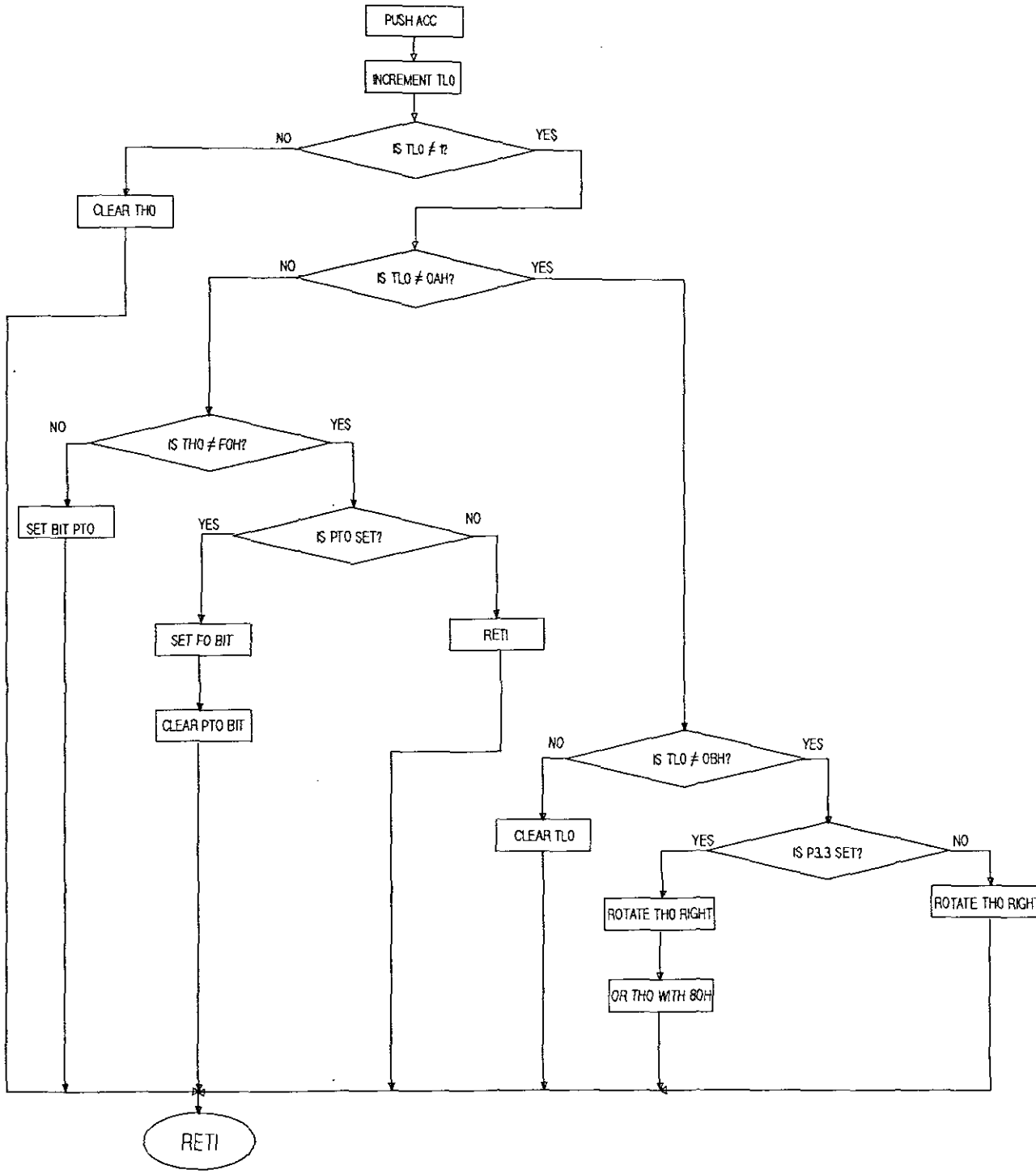


FIGURE 4.3 FLOWCHART - TRANSMISSION ROUTINE (TXRX)

4.1.21 Ex0int routine. (Refer to page 70 for program listing and the flowchart on page 51.) This routine is responsible for storing the make scan code in TH0 until the break code is detected. This is done by testing the value of pin 3.3 each time External interrupt 0 is triggered. Depending on whether a '1' or '0' is detected, the contents of TH0 are either rotated (right) or ORed with 80H and then rotated. Once 10 (TL0=0AH) bits have been sampled, the routine checks the value of TH0. If it is not equal to F0H (break code), the PT0 flag is set indicating a scan code has been stored. If however, TH0 is equal to F0H, and the PT0 flag has already been set, the main 'C' routine is notified by the setting of the F0 flag.

TL0 acts as a bit counter and is reset to zero once it has reached 11 (0BH).



FLOWCHART - EXOINT ROUTINE

FIGURE 4.4

4.2 'C' PROGRAM LISTING

```
/*****************************************************************/
/* This is the application program for the Voiceless Telephone */
/*****************************************************************/

#include "c:\51\io51.h"                 /* Include external program */
#define  RS      P1_0_bit               /*   Define read-status bit */
#define  RW      P1_1_bit               /*   Define read-write bit  */
#define  E       P1_2_bit               /*   Define E bit,,,        */
#define  ENABLE  P1_3_bit               /*   Define enable bit      */
#define  RING    P2_5_bit               /*   Define ring detect bit */
#define  TONE    P2_6_bit               /*   Define tone detect bit */
#define  DCD     P2_7_bit               /*   Define DCD detect bit  */
#define  CLK     P3_4_bit               /*   Define clocking  bit   */
#define  DATA    P3_5_bit4              /*   Define data bit        */
/* data declarations */
char a,k,q,d,f,s,i,p,g,w,v;             /*    Declare variables     */
char rowflag='b';                       /*    Set rowflag = bottom  */
char chcnt=0;                           /*    Set chara. count = 0  */
char discon=0;                          /*    Set disconnect = off  */
char count=0;                           /*    Set count = 0         */
char fin=0;                             /*    Set fin = 0           */
char ignore=0;                          /*    Set ignore = 0        */
char oa=0;                              /*    Set oa = 0            */
unsigned char h,c,z;                    /*    Declare unsigned vari.*/

const char txt0[]={"Transmitting message"};/*  Define system   */
const char txt1[]={"Message received    "};/*     messages     */
const char txt2[]={"Buffer full..    "};   /*                  */
const char txt3[]={"**** Voiceless *****"};/*   Define system  */
const char txt4[]={"**** Telephone *****"};/*      messages    */
const char txt5[]={"Dial tone detected.."};/*                  */
const char txt6[]={"Dialing.."};   /*                         */
const char txt7[]={"Ringing.."};   /*        Define system    */
const char txt8[]={"Engaged.."};   /*          messages       */
const char txt9[]={"No answer.."}; /*                         */
const char txt10[]={"Testing line.."};/*      Define system   */
const char txt11[]={"No dial tone.."};/*        messages      */
const char txt12[]={"Connected.."};/*                         */
const char txt13[]={"Disconnected.."};/*                      */
const char txt14[]={"Number unavailable"};/*   Define system  */
const char txt15[]={"Auto answer enabled "};/*   messages      */
const char txt16[]={"Inactive key"};/*                        */
const char txt17[]={"Initiating call"};/*                     */
const char txt18[]={"Call cancelled"}; /*     Define system   */
const char txt19[]={"F3 after dialing"};/*       messages     */
const char txt20[]={"Carrier detected"};/*                    */
const char txt21[]={"Connection failure"};/*                  */
const char txt22[]={"Answering call"};/*      Define system   */
const char txt23[]={"Auto answer disabled"};/*   messages     */
const char txt24[]={"Line fault"}; /*                         */
```

```
const char txt25[]={"Go ahead and dial.."};/*                    */
const char txt33[]={"Dialing"};     /*                           */
const char txt34[]={"complete"};    /*                           */

const char dec[]={
0x7f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,
0x2c,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x71,0x31,0x3f,0x3f,0x32,0x7a,0x73,
0x61,0x77,0x32,0x3f,0x3f,0x63,0x78,0x64,0x65,0x34,0x33,0x3f,0x3f,0x20,
0x76,0x66,0x74,0x72,0x35,0x3f,0x3f,0x6e,0x62,0x68,0x67,0x79,0x36,0x3f,
0x3f,0x3f,0x6d,0x6a,0x75,0x37,0x38,0x3f,0x3f,0x2c,0x6b,0x69,0x6f,0x30,
0x39,0x58,0x3f,0x2e,0x2f,0x6c,0x3b,0x70,0x5f,0x3f,0x3f,0x3f,0x2c,0x3f,
0x5b,0x3d,0x3f,0x3f,0x3f,0x3f,0x2a,0x5d,0x3f,0x3f,0x3f,0x3f,0x1f,0x3f,
0x3f,0x1e,0x3f,0x3f,0x3f,0x3f,0x3f,0x31,0x10,0x34,0x37,0x3f,0x3f,0x3f,
0x30,0x3f,0x32,0x35,0x36,0x38,0x3f,0x3f,0x3f,0x2b,0x33,0x2d,0x2a,0x39,
0x3f,0x3f,0x00,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,
0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x51,0x21,0x3f,0x3f,0x3f,
0x5a,0x53,0x41,0x57,0x40,0x3f,0x3f,0x43,0x58,0x44,0x45,0x24,0x23,0x3f,
0x3f,0x20,0x56,0x46,0x54,0x52,0x25,0x3f,0x3f,0x4e,0x42,0x48,0x47,0x59,
0x5e,0x3f,0x3f,0x20,0x4d,0x4a,0x55,0x26,0x2a,0x3f,0x3f,0x3c,0x4b,0x49,
0x4f,0x29,0x28,0x3f,0x3f,0x3e,0x3f,0x4c,0x3a,0x50,0x2d,0x3f,0x3f,0x2c,
0x22,0x69,0x7b,0x2b,0x39,0x3f,0x3f,0x3f,0x3f,0x7d,0x3b,0x3f,0x2d,0x3f,
0x3f,0x3f,0x2c,0x3f,0x5b,0x3d,0x3f,0x3f,0x3f,0x31,0x34,0x34,0x37,0x3f,
0x3f,0x3f,0x30,0x2e,0x32,0x35,0x36,0x38,0x3f,0x3f,0x3f,0x2b,0x33,0x2d,
0x2a,0x39,0x3f,0x3f,0x30,0x3f,0x32,0x35,0x36,0x38,0x3f,0x3f,0x3f,0x2b,
0x33,0x2d,0x2a,0x39,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,
0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f
};
char *ptr;                          /*    Define pointer         */
void lcd(char *);                   /*    Define subroutine      */
void display(unsigned char);        /*  Output data to display   */
void control(unsigned char);        /*    Define subroutine      */
void txrx();                        /*    Define subroutine      */
void cont();                        /*    Define subroutine      */
void delay(unsigned char);          /*    Define subroutine      */
char get_status();                  /*    Define subroutine      */
void latch(unsigned char);          /*    Define subroutine      */
char rbuf[78];                      /*    Define recv. buffer    */
char tbuf[78];                      /*    Define trans. buffer   */
char rcnt=0;                        /*    Define recv. counter   */
char tcnt=0;                        /*    Define trans. counter  */
unsigned char serial=00;            /*    Define control byte    */
extern void ex0int();               /*    Define ext subroutine  */

/*****************************************************************/
/* This routine displays the text specified in parenthesis       */
/* on the screen.                                                */
/*****************************************************************/

void lcd(char *text)                /*    Define subroutine      */
{
   while(*text)                     /* While text is available   */
        {
        display(*text);             /*    display text and       */
```

```
        text++;                         /*    increment pointer     */
        }
}
/*****************************************************************/
/*    This routine determines the status on the LCD display      */
/*****************************************************************/
char get_status()                       /*   Read status of display */
{
    unsigned char up,lo;                /*   Define "up" and "lo"   */
    delay(3);                           /*         Delay            */
    clear_bit(RS);                      /*   Clear read-status flag */
    set_bit(RW);                        /*   Set read mode          */
    set_bit(E);                         /*   Enable display         */
    up=input(P1);                       /*   Input high nibble      */
    clear_bit(E);                       /*   Disable display        */
    delay(3);h                          /*   Delay - min = 1us      */
    set_bit(E);                         /*   Enable display         */
    lo=input(P1);                       /*   Input low nibble       */
    clear_bit(E);                       /*   Disable display        */
    if(up>=0x80) return(1);             /*   If busy - return "1"   */
    else return(0);                     /*     else return "0"      */
}
/*****************************************************************/
/*   This routine provides a delay specified in parenthesis      */
/*****************************************************************/
void delay(unsigned char c)             /*   Presettable delay      */
{
    unsigned char d;                    /*   Define variable "d"    */
    for(d=0;d<c;d++);                   /*   Loop for "c"           */
}
/*****************************************************************/
/*   This routine latches control data to the display in         */
/*   four-bit mode.                                              */
/*****************************************************************/
void outc(unsigned char c)              /*   Output control nibble  */
{
    c=((c<<4)&0xf8);                    /*  Shift c & AND with F8H  */
    output(P1,c);                       /*   Output to port 1       */
    set_bit(E);                         /*   Enable display         */
    clear_bit(E);                       /*   Disable display        */
}
/*****************************************************************/
/*   This routine latches the text into the display buffers in   */
/*   four-bit mode.                                              */
/*****************************************************************/
void outd(unsigned char c)              /*    Output data nibble    */
{
    c=((c<<4)&0xf8);                    /*  Shift c & AND with F8H */
    output(P1,c);                       /*     Output to port 1    */
    clear_bit(RW);                      /*     Set write mode      */
    set_bit(RS);                        /*                         */
    set_bit(E);                         /*     Enable display      */
    clear_bit(E);                       /*     Disable display     */
```

```
/*****************************************************************/
/*  This routine provides a delay between each nibble of data  */
/*  latched to the display using the OUTC routine              */
/*****************************************************************/
void latch(unsigned char c)          /*   Output control data    */
{
   outc(c);                          /*      Output data         */
   delay(0);                         /*      Delay - 150 us      */
}
/*****************************************************************/
/*  This routine clocks the control byte into the serial to    */
/*  parallel converter and then latches the output to          */
/*  control the two relays, the ring LED and the handshak-     */
/*  ing signals for the modem.                                 */
/*****************************************************************/
void control(unsigned char serial) /*   Output control nibble  */
{
 for (w=0;w<=7;w++)                /*  Repeat process eight    */
  {                                /*        times             */
  h=serial;                        /*  Set serial equal to h   */
  h=(h&0x01);                      /*    And h with 01H        */
  if (h==0x00)                     /*  If h=0, clear data bit  */
   {
   clear_bit(DATA);
   }
  else                             /* else, set data bit       */
   {
   set_bit(DATA);
   }
  delay(1);                        /*          Delay           */
  set_bit(CLK);                    /*   Clock serial to        */
  delay(1);clear_bit(CLK);         /*    parallel converter    */
  serial=(serial>>1);              /* Shift serial right       */
  }
  set_bit(ENABLE);                 /*   Latch output of serial */
  delay(1);
  clear_bit(ENABLE);               /*   to parallel conv.      */
}
/*****************************************************************/
/*  This routine initialises the screen, presets the latched   */
/*  value of the serial to parallel converter, clears the      */
/*  transmit and receive buffers and disables the unnecces-    */
/*  sary interrupts on the processor.                          */
/*****************************************************************/
void init()                          /*   Initz. display & proc. */
{
 serial=0x0e;                        /*   Lower DTR &            */
 control(serial);                    /*   disconnect line        */
 output(IE,0x11);                    /*   Disable all interrupts */
 q=input(SBUF);                      /*   Clear SBUF(out)        */
 for(i=0;i<78;i++)
 {                                   /*  Clear transmit and      */
 rbuf[i]=0x20;                       /*      receive buffers     */
```

```
 tbuf[i]=0x20;
 }
 output(TH0,0x00);                    /*    Set timer 0 = 0000H    */
 output(TL0,0x00);
 output(TH1,0x50);                    /*    Set timer 1 = 5050H    */
 output(TL1,0x50);
 output(TMOD,0x20);                   /*    Set TMOD = 20H         */
 output(TCON,0xc1);                   /*   Set TCON = C1H          */
 output(SCON,0x50);                   /*   Set SCON = 50H          */

 clear_bit(RS);                       /*    Select control         */
 clear_bit(RW);                       /*    Set write mode         */
 outc(0x03);                          /*    Set 8 bit mode         */
 clear_bit(E);                        /*    Enable display         */
 delay(30);                           /*    Delay ~ 5 ms           */
 latch(0x03);                         /*                           */
 latch(0x03);                         /*                           */
 latch(0x02);                         /*                           */
 latch(0x02);                         /*    Set 4-bit mode         */
 latch(0x0c);                         /*    Set 2 row mode         */
 latch(0x00);                         /*    Cursor & display on.   */
 latch(0x0f);                         /*    Set blinking cursor.   */
 latch(0x00);                         /*    Home cursor            */
 latch(0x01);                         /*      and clear display.   */
 while(get_status());                 /*                           */
 latch(0x08);                         /*    Set top row            */
 latch(0x00);                         /*        first character.   */
 q=input(SBUF);                       /*   Clear SBUF (input)      */
 output(IE,0x81);                     /*   Enable serial interrupt*/
 discon=0;                            /*      Set discon = 0       */
}

/*****************************************************************/
/*  This routine outputs text to the display one nibble at a     */
/*  time - most significant nibble first.                        */
/*****************************************************************/

void display(unsigned char c)         /*   Output data to display */
{
 while(get_status());                 /*   While not busy,        */
 outd((c&0xf0)>>4);                   /*      output MSN          */
 delay(0);                            /*       delay - min 1us    */
 outd(c&0x0f);                        /*          output LSN.     */
 ignore=0;                            /*  Reset ignore flag (Off) */
void t0int()                          /*                          */
{
}
void ex1int()                         /*                          */
{
}
void t1int()                          /*                          */
{
}
```

```
/*****************************************************************/
/*  This routine copies the data received into a buffer called */
/*  RBUF until a carriage return (0DH) is received.          */
/*****************************************************************/


void rint()                          /*    Receive interrupt     */
{
 if(read_bit(RI_bit))                /*    If RI bit set,        */
 {
 clear_bit(RI_bit);                  /*      clear RI bit.       */
 if((rbuf[rcnt]=input(SBUF))!=0x0d)/*   Place SBUF into RBUF   */
  rcnt++;                            /*     until CR received    */
 else set_bit(PX1_bit);              /*   Then set PX1 bit       */
 }
}


/*****************************************************************/
/* This routine clears the screen and displays "Buffer full.." */
/*****************************************************************/


void buffull()                       /*   Buffer full message    */
{
delay(10);                           /*         Delay            */
if (rowflag=='t')                    /* If cursor is on top      */
{                                    /* row, set cursor to first */
latch(0x0c);                         /* chara. of second row     */
latch(0x00);
}
else                                 /*  else set cursor to the  */
{                                    /*   first position on the  */
latch(0x08);                         /*         top row          */
latch(0x00);
}
lcd(&txt2);                          /* Display "Buffer full.."  */
}
/*****************************************************************/
/* This routine displays the logo on the screen.                 */
/*****************************************************************/

logo()
{
 latch(0x00);                        /*       Set cursor to      */
 latch(0x01);                        /*         first line.      */
 lcd(&txt3);                         /*       Display text 3.    */
 latch(0x0c);                        /*        Set cursor to     */
 latch(0x00);                        /*         second line      */
 lcd(&txt4);                         /*       Display text 4.    */
 delay(100);                         /*         Delay            */
}
```

```
/*****************************************************************/
/* This routine samples the output of the frequency detection    */
/* circuit (centre frequency 400Hz) and from this determines     */
/* the type of tone.                                             */
/*                                                               */
/* Dial tone    - v>= 13 Ring tone4   - v>=1 and <=74            */
/* Line fault   - v = 04  Engaged     - v>=8 and >=12            */
/*****************************************************************/

void tonetst()
{
 v=0;                                /*  Reset v=0                */
 latch(0x00);                        /*  Clear screen             */
 latch(0x01);                        /*    and set cursor         */
 lcd(&txt10);                        /* Display "Testing line.." */
 serial=(serial|0x80);               /* Switch in tone detection */
 control(serial);                    /*      circuitry           */
 for (g=0;g<20;g++);                 /*  Take 20 samples         */
  {
  set_bit(TONE);                     /* Set bit 2_6 to input mode*/
  for (i=0;i<25;i++)                 /*      Delay               */
  delay(10);
  z=input(P2);                       /*      Read port 2         */
  z=(z&0x40);                        /*  And 'z' with hex 40H    */
  if(z!=0);                          /*    If z=1, increment     */
   {                                 /*     the variable "v"     */
   v=v+1;
   }
  for(i=0;i<23;i++)                  /*      Delay               */
  delay(10);
  }
 delay(10);
 serial=(serial&0x7f);               /*    Switch out tone       */
 control(serial);                    /*   detection circuitry.   */
}



/*****************************************************************/
/* This routine controls the initialisation of an outgoing       */
/*  call.                                                        */
/*****************************************************************/
void ogcall()
{
 lcd(&txt17);                        /* Display "Initiating call"*/
 for (i=0;i<50;i++)                  /*      Delay               */
 delay(100);
 tonetst();                          /*   Test for dial tone     */
 latch(0x00);                        /*   Clear display          */
 latch(0x01);                        /*    and set cursor        */
```

```
 if(v>=10)                          /*  If dial tone detected,  */
  {
  cont();                           /*  continue with call.     */
  }
 else                               /* If no dial tone was      */
  {                                 /*     detected,            */
  lcd(&txt11);                      /* Display "No dial tone"   */
  for(i=0;i<100;i++)
  delay(100);                       /*      Delay               */
  latch(0x00);                      /*   Clear screen           */
  latch(0x01);                      /*    and set cursor.       */
  lcd(&txt18);                      /*  Display"Call cancelled" */
  for(i=0;i<100;i++)
  delay(100);                       /*      Delay               */
  }
 init();                            /*  Re-initialise unit      */
 delay(100);                        /*           Delay          */
 logo();                            /*   Display logo & return  */
}

/*********************************************************************/
/* This routine handles the automatic answering of a call.           */
/*********************************************************************/
void autoa()
{
    serial=(serial|0x10);           /*   Switch in transformer  */
    control(serial);                /*         circuit.         */
          delay(100);               /*           Delay.         */
    serial=(serial&0xfb);           /*   Switch on DTR.         */
    control(serial);                /*                          */
          delay(250);               /*            Delay         */
    serial=(serial&0xfd);           /*      Switch on RI        */
    control(serial);                /*                          */
    for(i=0;i<50;i++)
          delay(100);               /*           Delay.         */
    serial=(serial|0x02);           /*    Switch off RI         */
    control(serial);                /*                          */
    serial=(serial&0xf7);           /*   Switch on RTS          */
    control(serial);                /*                          */
    for(i=0;i<150;i++)              /*                          */
        {
        delay(250);                 /*         Delay            */


    set_bit(DCD);                   /*  Convert P2-7 to input   */
    delay(200);                     /*         Delay            */
    z=input(P2);                    /*  Input from port 2       */
    z=(z&0x80);                     /*   And with hex 80H       */
    if(z!=0)                        /*   If carrier detected    */
        {
        latch(0x00);                /*   Clear screen           */
        latch(0x01);                /*     and set cursor       */
```

```
        lcd(&txt20);                    /*   Display "CD"             */
        for (i=0;i<50;i++)              /*      Delay                 */
             delay(100);
        latch(0x00);                    /*   Clear screen and         */
        latch(0x01);                    /*       set cursor           */
        lcd(&txt12);                    /*   Display "Connected"      */
        for (i=0;i<50;i++)              /*      Delay                 */
             delay(100);
        latch(0x00);                    /*   Clear screen and         */
        latch(0x01);                    /*         set cursor         */
        discon=0;                       /* Set disconnect = 0 (Off)   */
        txrx();                         /* Enter transmission mode    */
        }
    elseh                               /*    If no carrier           */
        {                               /*        is detected,        */
        init();                         /*   Initialise  unit         */
        delay(10);                      /*        Delay               */
        lcd(&txt21);                    /* Display - "Connection"     */
        for(i=0;i<100;i++)              /*     failure"               */
          delay(250);                   /*           Delay            */
        }
}

/*****************************************************************/
/* This routine handles the wrap-around feature of the display.*/
/*****************************************************************/
void scrnset()                          /*   Set cursor position    */
{
        chcnt++;                        /*   Increment chara. count */
        if (chcnt==20)                  /* If end if row is reached */
            {
            chcnt=0;                    /*  Set chara. count = 0    */
            if (rowflag=='b')           /*   If on bottom row,      */
                {
                latch(0x00);            /*    Clear display,        */
                latch(0x01);            /*     home cursor,         */
                ptr=((tbuf+tcnt)-20);/*  Display the last 20       */
                while(*ptr)             /*    of the message        */
                    {                   /*                          */
                    display(*ptr);      /*      being edited on     */
                    ptr++;              /*         top line.        */
                    }
                  latch(0x0c);          /*    Set cursor at first   */
                  latch(0x00);          /*     pos. on bottom row.  */
                }


            else                        /*      If on top row,      */
                {
                latch(0x0c);            /*    Set cursor at first   */
                latch(0x00);            /*     pos. on bottom row,  */
                rowflag='b';            /*  and set rowflag=bottom  */
                }
```

```
        }
}

/*****************************************************************/
/*   This routine checks the value in parenthesis (scan code   */
/*   from the keyboard for special functions; eg. F1,F2,Back   */
/*   Space,etc.                                                */
/*****************************************************************/
void chchk(c)                          /*  Check for function keys */
{
 ignore=0;                             /*  Set ignore = 0 (Off)    */
 switch(c)                             /*                          */
  case 0x58:                           /* If Caps Lock is pressed, */
  clear_bit(EX0_bit);                  /*     Disable keyboard     */
  ignore=1;                            /*  Set ignore = 1 (On)     */
  if(s==1)                             /*    If caps lock on,      */
  s=0;                                 /*       switch off.        */
  else                                 /*     If caps lock off,    */
  s=1;                                 /*       switch on.         */
  clear_bit(F0_bit);                   /*    Clear int. bit        */
  set_bit(EX0_bit);                    /*    Enable keyboard       */
  break;                               /*  Break from routine      */

  case 0x5a:                           /*   If Enter is pressed,   */
  clear_bit(EX0_bit);                  /*   Disable keyboard       */
  latch(0x00);                         /*    Clear display,        */
  latch(0x01);                         /*     home cursor,         */
  chcnt=0;                             /*    Set chara. count = 0  */
  rowflag='t';                         /*    Set rowflag to top,   */
  lcd(&txt0);                          /*   Display "Transmitting  */
  for(i=0;i<=(tcnt-1);i++)             /*  Transmit message        */
   {
   output(SBUF,tbuf[i]);
   while(!(read_bit(TI_bit)));
   clear_bit(TI_bit);
   }
  output(SBUF,0x0d);                   /*   Transmit CR            */
  while (!(read_bit(TI_bit)));
  clear_bit(TI_bit);
  delay(100);                          /*      Delay               */
  latch(0x00);                         /*   Clear display          */
  latch(0x01);                         /*     and home cursor.     */
  ignore=1;                            /*   Set ignore = 1 (On)    */
  tcnt=0;                              /*   Set tcnt = 0           */
  chcnt=0;                             /*   Set chcnt = 0          */
  clear_bit(F0_bit);                   /*   Clear interrupt bit    */
  set_bit(EX0_bit);                    /*    Enable keyboard       */

  break;                               /*    Break from routine    */

  case 0x05:                           /*   If F1 is pressed,      */
  clear_bit(EX0_bit);                  /*   Disable keyboard       */
```

```
delay(10);                          /*        Delay                */
serial=(serial&0xdf);               /*   Switch off LED            */
control(serial);                    /*                             */
latch(0x02);                        /*  Convert to one-line        */
latch(0x04);                        /*     mode                    */
delay(100);                         /*             Delay           */
latch(0x00);                        /*    Clear display            */
latch(0x01);                        /*     and home cursor         */
ptr=&rbuf;                          /*      Display received       */
for (i=0;i<rcnt;i++)                /*         message             */
 {
 display(*ptr);
 ptr++;
 }
for (i=0;i<rcnt;i++)
rbuf[i]=0x20;                       /*       Clear receive         */
for (i=0;i<50;i++)                  /*   Delay for 2 seconds       */
delay(200);
for (i=0;i<rcnt;i++)
 {                                  /*   Rotate message            */
latch(0x01);                        /*  until the whole            */
latch(0x08);                        /*   message has been          */
for (d=0;d<15;d++)                  /*         displayed.          */
delay(100);                         /*                             */
}
delay(10);                          /*           Delay             */
latch(0x02);                        /* Convert back to two-        */
latch(0x0c);                        /*          line mode          */
delay(10);                          /*         Delay               */
latch(0x00);                        /*    Home cursor              */
latch(0x01);                        /*     and clear display.      */
if (tcnt!=0)                        /* If the transmit buffer      */
 {                                  /*         is not empty;       */
 ptr=((tbuf+tcnt)-chcnt);
 for (i=0;i<chcnt;i++)              /*   display one line          */
  {                                 /*     of the  message         */
  display(*ptr);                    /*      being edited on        */
  ptr++;                            /*         the top line        */
  }
 }
rcnt=0;                             /* Set receive count = 0       */
rowflag='t';                        /* Set rowflag to top line     */
ignore=1;                           /* Set ignore = 1 (On)         */
clear_bit(F0_bit);                  /*    Clear interrupt bit      */
set_bit(EX0_bit);                   /*    Enable keyboard          */
delay(10);                          /*          Delay              */
break;                              /*  Break from routine         */

case 0x09:                          /*  If F10 is pressed,         */
delay(10);                          /*           Delay             */

discon=1;                           /*   Set discon = 1            */
```

```
  ignore=1;                          /*   Set ignore = 1 (On)     */
  break;                             /*  Break from routine       */

  case 0x66:                         /* If Back Space is pressed */
  clear_bit(EX0_bit);                /*    Disable keyboard      */
  ignore=1;                          /* Set ignore = 1 (on)      */
  if (tcnt>=1)                       /*  If transmit buffer is   */
   {                                 /*  not empty;              */
   tcnt=tcnt-1;                      /*     Decrement tcnt       */
   chcnt=chcnt-1;                    /*   Decrement chara. count */
   if (chcnt!=0xff)                  /*  If not at beginning     */
    {                                /*      of row,             */
    latch(0x01);                     /*  Shift cursor one        */
    latch(0x00);                     /*    position left         */
    display(0x20);                   /*  output a space (20H)    */
    latch(0x01);                     /*    shift cursor one      */
    latch(0x00);                     /*   position left again    */
    }
   else                              /*  If at the beginning     */
    {                                /*        of a row,         */
    if (rowflag=='t')                /*     If on top row,       */
     {
     latch(0x00);                    /*  Clear display and       */
     latch(0x01);                    /*  home cursor.            */
     ptr=((tbuf+tcnt)-18);
     for (i=0;i<19;i++)
     {                               /* display message          */
     display(*ptr);                  /*  being edited on         */
     ptr++;                          /*    top line              */
     }
     latch(0x09);                    /* Set cursor at last       */
     latch(0x03);                    /* position on top line     */
     chcnt=19;                       /*  Set chcnt to 19         */
     rowflag='t';                    /* Set rowflag to top       */
     }
    else                             /* If on bottom row,        */
     {
     latch(0x09);                    /* Pos. cursor on last      */
     latch(0x03);                    /*  chara. on top row,      */
     display(0x20);                  /* Display a space(20H)     */
     latch(0x01);                    /*  Shift cursor one        */
     latch(0x00);                    /* position left.           */
     chcnt=19;                       /* Set chara.count=19       */
     rowflag='t';                    /*  Set rowflag to top      */
     }
    }
   }
  ignore=1;                          /* Set ignore = 1 (On)      */
  clear_bit(F0_bit);                 /* Clear interrupt bit      */
  set_bit(EX0_bit);                  /*   Enable keyboard        */
  break;                             /*  Break from routine      */
          }
}
```

```
/*****************************************************************/
/*    This routine handles an out-going call once dial tone      */
/*    has been detected.                                         */
/*****************************************************************/
void cont()
{
    lcd(&txt5);                    /*  Display "DT detected"    */
    delay(100);                    /*         Delay             */
    latch(0x0c);                   /*   Clear display           */
    latch(0x00);                   /*     and home cursor.      */
    delay(100);                    /*           Delay           */
    lcd(&txt25);                   /* Display "Go ahead and.." */
    output(IE,0x11);               /*Disable serial interrupts */
    k=0;                           /*        Set k = 0          */
    while(k==0)                    /*  Wait for dialing         */
        {
        set_bit(RING);             /*      Set P2_5 to input    */
        delay(10);                 /*         Delay             */
        z=0;                       /*        Set z = 0.         */
        z=input(P2);               /*        Read port 2.       */
        z=(z&0x20);                /*   And with hex 20H.       */
        k=z;                       /*        Set k = z.         */
        }                          /*                           */
    latch(0x00);                   /*   Clear display           */
    latch(0x01);                   /*      and home cursor.     */
    delay(100);                    /*           Delay.          */
    lcd(&txt33);                   /*     Display "Dialing"     */
    count=0;                       /*    Set count = 0          */
    fin=0;                         /*    Set fin = 0            */
    while(fin!=10)                 /* Test for end of dialing   */
        {                          /*                           */
        set_bit(RING);             /*  Set 2_5 to input mode    */
        delay(10);                 /*           Delay           */
        z=0;                       /*        Set z = 0          */
        z=input(P2);               /*     Read port 2.          */
        z=(z&0x20);                /*  And input with 20H       */
        delay(1);                  /*        Delay              */
        if(z!=0)                   /*  If dialing detected,     */
            {                      /*                           */
            count=0;               /*  Reset count to 0.        */
            }                      /*                           */
        else                       /*  If no dialing            */
            {                      /*                           */
            count=(count+1);       /*  Increment 'count'        */
            }                      /*                           */
        if(count==250)             /*  If 'count' = 250,        */
            {                      /*                           */
            fin=(fin+1);           /*  increment 'fin'.         */
            }                      /*                           */
        }                          /*                           */
        latch(0x08);               /*   Set cursor on space     */
        latch(0x08);               /*     after "Dialing".      */
        delay(100);                /*          Delay            */
```

```
lcd(&txt34);                /* Display "completed"      */

for(i=0;i<150;i++)          /*      Delay               */
    delay(10);              /*                          */
tonetst();                  /* Test for ringing         */
latch(0x00);                /*   Clear screen           */
latch(0x01);                /*  and home cursor.        */
if(v>=1 && v<=7)            /*   If ring detected       */
    {                       /*                          */
    lcd(&txt7);             /* Display "Ringing"        */
    for(i=0;i<250;i++)      /*                          */
        delay(150);         /*    Delay                 */
    tonetst();              /* Test for ringing again.  */
    if (v>=1);              /* If still ringing,        */
        {                   /*                          */
        latch(0x00);        /*     Set cursor and       */
        latch(0x01);        /*   home cursor.           */
        delay(100);         /*  Delay                   */
        lcd(&txt9);         /* Display "No answer".     */
        for(i=0;i<150;i++)  /*      Delay               */
            delay(10);      /*                          */
        init();             /*   Re-initialise unit.    */
        }
    else                    /*  If the call has been    */
        {                   /*  answered.               */
        serial=(serial|0x10);/* Switch in transformer   */
        control(serial);    /*    circuit               */
        delay(100);         /*        Delay             */
   serial=(serial&0xfb);    /*    Switch on DTR         */
        control(serial);    /*                          */
        delay(250);         /*        Delay             */
        serial=(serial&0xf7);/* Switch on RTS           */
        control(serial);    /*                          */
        for(i=0;i<250;i++)  /*                          */
            delay(100);     /*    Delay                 */
        set_bit(DCD);       /*  Convert I/O to input    */
        delay(200);         /*          Delay           */
        z=0;                /*            z = 0.        */
        z=input(P2);        /*  Input from port 2       */
        z=(z&0x80);         /*      Check for carrier   */
        if(z!=0)            /*      If carrier detected */
            {               /*                          */
            latch(0x00);    /*   Clear screen           */
            latch(0x01);    /*   and home cursor.       */
            delay(10);      /*            Delay         */
            lcd(&txt20);    /*   Display "CD"           */
            for (i=0;i<50;i++) /*Delay                  */
                delay(100);/*                           */

            latch(0x00);    /*   Clear screen           */
            latch(0x01);    /*   and home cursor.       */
            lcd(&txt12);    /*   Display "Connected"    */
            discon=0;       /*                          */
```

```
                for (i=0;i<50;i++) /* Delay                    */
                    delay(100);/*                               */
                latch(0x00);   /*   Clear screen              */
                latch(0x01);   /*   and home cursor.          */

            output(IE,0x91);   /*  Enable serial interrupt    */
                txrx();        /*  Enter transmission mode    */
                lcd(&txt13);   /* Display "Disconnected"      */
                for (i=0;i<50;i++)/*   Delay                  */
                    delay(100);/*                               */
                clear_bit(EX0_bit);/*    Enable keyboard      */
                }
            else               /*     If no carrier is       */
                {              /*         detected           */
                latch(0x00);   /*   Clear screen              */
                latch(0x01);   /*   and home cursor.          */
                delay(10);     /*          Delay              */
                lcd(&txt21);   /*  Display "Connection        */
                for (i=0;i<50;i++) /*  failure" and delay.     */
                    delay(100);
                }
            }
        }
        if(v>=8 && v<=12)          /*   If engaged tone is     */
            {                      /*       detected,          */
            lcd(&txt8);            /* Display "Engaged"        */
            for (i=0;i<50;i++)     /*      Delay               */
            delay(100);
            latch(0x00);           /*   Clear screen and       */
            latch(0x01);           /*        home cursor.      */
            }
        if(v==0 || v>=13)          /* If an unrecognisable tone*/
            {                      /*        is detected,      */
            lcd(&txt24);           /* Display "Line fault"     */
            for (i=0;i<50;i++)     /*      Delay               */
            delay(100);            /*                          */
            latch(0x00);           /*   Clear screen and       */
            latch(0x01);           /*   home cursor.           */
            }
}

/*****************************************************************/
/* This routine is used to edit messages, transmit messages,     */
/* and indicates when a message has been received from the       */
/* remote caller.                                                */
/*****************************************************************/
void txrx()                        /*  Transmission routine     */
{
   output(IE,0x91);                /*   Enable keyboard.        */
   while(1)                        /*   Loop while connected.   */
        {
        if(read_bit_and_clear(F0_bit))/*  If key pressed,       */
            {
```

```
                output(IE,0x00);      /*   Disable all interrupts */
                clear_bit(RS);        /*    select cont           */
                clear_bit(RW);        /*    write to display      */
                c=input(TH0);         /*    c= KB input           */
                chchk(c);             /*  Check for function key  */
                if (s==1)             /*    If shift key -        */


                    c=c+0x80;         /*      add 80H to "c"      */

                if (ignore==0)        /*    If non-function       */
                {                     /*         key -            */
                tbuf[tcnt]=dec[c];    /*    Place input in        */
                tcnt++;               /*      transmit buffer     */
                display(dec[c]);      /*        and display.      */
                scrnset();            /*    Setup display         */
                if (tcnt==78)         /*    If buffer = 78  -     */
                      buffull();      /*    indicate buffer full. */
                set_bit(DCD);         /*  Set bit 2_7 to input.   */

                delay(100);           /*         Delay.           */
                z=input(P2);          /*       Read port 2        */
                z=(z&0x80);           /*   AND with hex 80H.      */

                if(z==0)              /*   If no carrier is       */
                     {                /*       detected           */
                     latch(0x00);     /*   Clear screen and       */
                     latch(0x01);     /*        home cursor.      */
                     lcd(&txt21);     /* Display "Conn. failure"  */
                     for (i=0;i<150;i++)/*        Delay           */
                       delay(20);     /*                        */
                     discon=1;        /*      Set discon = 0.   */
                     }                /*                        */
                }                     /*                        */
                if (discon==1)        /*    If disconn = 1      */
                    break;            /*     break from routine.*/
              }                       /*                        */
            output(IE,0x91);          /*  Enable interrupts     */
            if(read_bit(PX1_bit))     /*  If a message has      */
              {                       /*       been received,   */
              serial=(serial|0x20);   /*   Switch on LED        */
              control(serial);        /*                        */
              clear_bit(PX1_bit);     /*  Clear message received*/
             }                        /*            flag.       */
        }
}

/*****************************************************************/
/* This routine displays the LOGO and then waits for a keyboard*/
/* entry or ringing  tone before passing  control over to  the */
/* respective sub-routine.                                       */
/*****************************************************************/
void main()                           /*     Main routine       */
```

```
{
   delay(100);                          /*    Provide a delay 17 ms */
   init();                              /*    Initialise unit       */
   logo();                              /*        Display LOGO      */
   clear_bit(F0_bit);                   /*    Clear interrupt flag  */
   while(1)                             /*     Loop continuously    */
        {                               /*                          */
        set_bit(RING);                  /*   Set bit 2_7 to input.  */
        delay(10);                      /*           Delay.         */
        z=0;                            /*         Set z = 0        */
        z=input(P2);                    /*          Read port 2     */
        z=(z&0x20);                     /*   AND with hex 20H.      */
        if(z!=0)                        /*    If ringing has        */
            {                           /*       been detected.     */
             serial=(serial|0x20);      /*    Switch on LED         */
             control(serial);           /*                          */
             delay(150);                /*          Delay           */
             serial=(serial&0xdf);      /*   Switch off LED         */
             control(serial);           /*                          */
             }                          /*                          */
        if(read_bit_and_clear(F0_bit))/* If a key has              */
            {                           /*     been pressed.        */
             latch(0x00);               /*    Clear screen and      */
             latch(0x01);               /*       home cursor.       */
             c=input(TH0);              /*    c = KB input          */
             switch(c)                  /*        Check key         */
                {                       /*                          */
                 case 0x05:             /*   If F1 is pressed,      */
                     clear_bit(EX0_bit);/* Disable keyboard        */
                     ogcall();          /*  Goto OGCALL subroutine  */
                     clear_bit(F0_bit);/*    Clear int. flag       */
                     set_bit(EX0_bit);/*      Enable keyboard       */
                     break;             /*  Break from routine      */
                 case 0x04:             /*   If F3 is pressed,      */
                     clear_bit(EX0_bit);/* Disable keyboard        */
                     lcd(&txt22);       /*  Display  -  "Answering  */
                     for(i=0;i<50;i++)/*          call"           */
                          delay(100);/*          Delay             */
                     autoa();           /*  Goto AUTOA  subroutine  */
                     clear_bit(F0_bit);/*  Clear int. flag         */
                     set_bit(EX0_bit);/*      Enable keyboard       */
                     init();            /*   Initialise unit        */
                     delay(10);         /*          Delay           */
                     logo();            /*        Display LOGO      */
                     break;             /*  Break from routine      */
                 case 0x06:             /*   If F2 is pressed,      */
                     lcd(&txt15);       /*   Display "AA enabled"   */
                     a=0;               /*         Set a = 0        */
                     while((!(read_bit_and_clear(F0_bit)))&&a==0)
                        {               /*   Check for ringing.     */
                        set_bit(RING);/*  Set bit 2_5 to input      */
                        delay(10);      /*         Delay            */
                        z=0;            /*         Set z = 0        */
```

```
                    z=input(P2);/*       Read port 2.       */
                    z=(z&0x20);/*   AND with hex 20H.      */
                    if(z!=0)    /*  If ringing has          */
                        {       /*        been detected.   */
                        clear_bit(EX0_bit);/*  Disable KB  */
                        serial=(serial|0x20);/*  Switch on */
                        control(serial);/*     LED          */
                        latch(0x00);/* Clear display, and  */
                        latch(0x01);/*   home cursor.       */
                        lcd(&txt7);/* Display "Ringing"    */
                        for(i=0;i<250;i++)/*  Delay         */
                        delay(150);/*                       */
                        serial=(serial&0xdf);/* Switch off */
                        control(serial);/*     LED          */
                        autoa();/*  Goto AUTOA subroutine  */
                        a=1;    /*        Set a = 1.       */
                        clear_bit(F0_bit);/*Clear int. flag*/
                        set_bit(EX0_bit);/* Enable keyboard*/
                        init();/*    Initialise unit        */
        }
                    }
                    if (discon==0)/*     If discon has      */
                        {       /*        been set to 0     */
                        latch(0x00);/* Clear display and    */
                        latch(0x01);/*    home cursor.       */
                        lcd(&txt23);/* Display"Auto answer */
                        for(i=0;i<100;i++)/*   disabled"    */
                        delay(150);/*         Delay         */
                        }
                    delay(100);/*         Delay             */
                    logo();     /*    Display LOGO         */
                    break;      /*   Break from routine    */
                default:        /*         Else             */
                    clear_bit(EX0_bit);/*   Disable keyboard  */
                    lcd(&txt16);  /*  Display "Inactive key"  */
                    for (i=0;i<30;i++)/*         Delay        */
                        delay(200);/*                        */
                    logo();       /*        Display LOGO    */
                    clear_bit(F0_bit);/*  Clear int. flag    */
                    set_bit(EX0_bit);/*   Enable keyboard    */
                    break;        /*   Break from routine    */
                }
            }
        }
}
```

4.3 ASSEMBLER LISTING

```
            NAME EXINTERR

            PUBLIC ex0int
            RSEG    PSTART

ex0int:                         ;On keyboard interrupt
      push acc                  ;Save PSW
      inc tl0                   ;Increment Timer 0 (low)
ptart:                          ;
      jnb p3.2,ptart            ;Wait for INT0 to go low
      mov a,tl0                 ;Move contents of TL0 to ACC.
      cjne a,#01,next           ;If ACC not = 1; goto NEXT;
      mov th0,#00h              ;else move 00H to TH0
      ljmp outof                ;and goto OUTOF
next:                           ;
      mov a,tl0                 ;Move contents of TL0 to ACC.
      cjne a,#0ah,next1         ;If ACC not = 0AH, goto NEXT1.
      mov a,th0                 ;Else move contents of TH0 to ACC.
      cjne a,#0f0h,next5        ;If ACC not = F0H, goto NEXT5.
      setb PT0                  ;Else set bit PT0
      ljmp outof                ;and goto OUTOF.
next5:                          ;
      jnb PT0,outof             ;If PT0 is not set, goto OUTOF.
      setb F0                   ;Else set bit F0,
      clr PT0                   ;clear bit PT0,
      ljmp outof                ;and goto OUTOF.
next1:                          ;
      mov a,tl0                 ;Move contents of TL0 to ACC.
      cjne a,#0bh,next2         ;If ACC not = 0BH, goto NEXT2
      mov tl0,#00h              ;Else clear TL0
      ljmp outof                ;and goto OUTOF.
next2:                          ;
      jnb p3.3,zero             ;If P3_3 is not set, goto ZERO.
      mov a,th0                 ;Else move contents of TH0 to ACC
      rr a                      ;and rotate accumulator right.
      orl a,#80h                ;Logically OR ACC with 80H.
      mov th0,a                 ;Move contents of ACC to TH0
      ljmp outof                ;and goto OUTOF.
zero:                           ;
      mov a,th0                 ;Move the contents of TH0 to ACC
      rr a                      ;Rotate ACC right and
      mov th0,a                 ;move the contents of ACC to TH0
outof:                          ;
      pop acc                   ;Restore PSW
      reti                      ;Return from interrupt.

      END


      ACC = Accumulator

      PSW = Program Status Word
```

5.0 PROJECT SYNTHESIS AND RECOMMENDATIONS.

The objective of this project was to develop a prototype unit that will allow the deaf and mute society of South Africa to cost effectively communicate via the existing Public Switched Telephone Network (PSTN). The "Voiceless Telephone" has achieved this, and has been met with great enthuisiasm by the Institutes and Societies providing various services to the handicapped. When demonstrated to the Institute for the Deaf in Worcester, Miss Ula Dedekind, an audiologist at the institute, said that; "The unit is simple in operation, making it suitable for all age groups, and provides telephone communication between hearing impaired people that would otherwise be isolated when on thier own." Miss Dedekind went on to say that; "PC'sare currently being used to perform a similar function, but because of the costs involved, its use was limited to the more privileged. If the unit, (the Voiceless Telephone), was to be marketed in the region of a few hundred rands, it would overcome this problem.

Mr Alan Tot, Head of the Science Department said; "This unit would be most useful in the case of an emergency."

When asked what improvements or additional functions could be added to the Voiceless Telephone, the following suggestions were made :-

An answering machine facility would greatly enhance the unit;

and the dialing mechanism should be included in some versions

of the Voiceless Telephone for those homes where all members

of the family have seriously impaired hearing.

Both of these features could be incorporated into the unit at minimal cost because ,besides the software modifications, only the memory would need to be extended and a small alteration made to the relay circuitry.

6.0 BIBLIOGRAPHY

Intel Corporation. 1990. **8-Bit Embedded Controllers.** Mt. Prospect, Intel Literature Sales.

Intel Corporation. 1988. **Memory Components Handbook.** Santa Clara, Intel Literature Sales.

Motorola Semiconductors.1977. **LS TTL Low Power Schottky.** Fairchild Camera and Instrument Corporation.

IBM Personal Computer Reference Library. 1985. **Technical Reference Vol.2** Section 4.

General Instruments. 1983. **Optoelectronic Products.** Optoelectronics Division. Page 7.

National Semiconductors. 1982. **Linear Databook.** 3-257 to 3-259.

United Microelectronics Corporation. 1990/91. **UMC Communication IC's.**Page 5-23.

Orcad Systems Corporation. 1990. **Schematic Design Tools.**

Texas Instruments. 1985. **TTL Databook.**

SGS Thomson Microelectronics. 1989. **Modem Databook and Applications.**

7.0 ACKNOWLEDGEMENTS

I wish to thank my colleagues in Saponet's Research and Development Centre, especially Adriaan Botha, for all their support and advice over the past several months.

Many thanks, to all at the Institute for the Deaf, namely Mrs Ula Dedekind, Mr Louis Barnard and Mr Alan Tot and the staff of the Cape Town branch of Lifeline, for their assistance in evaluating the unit.

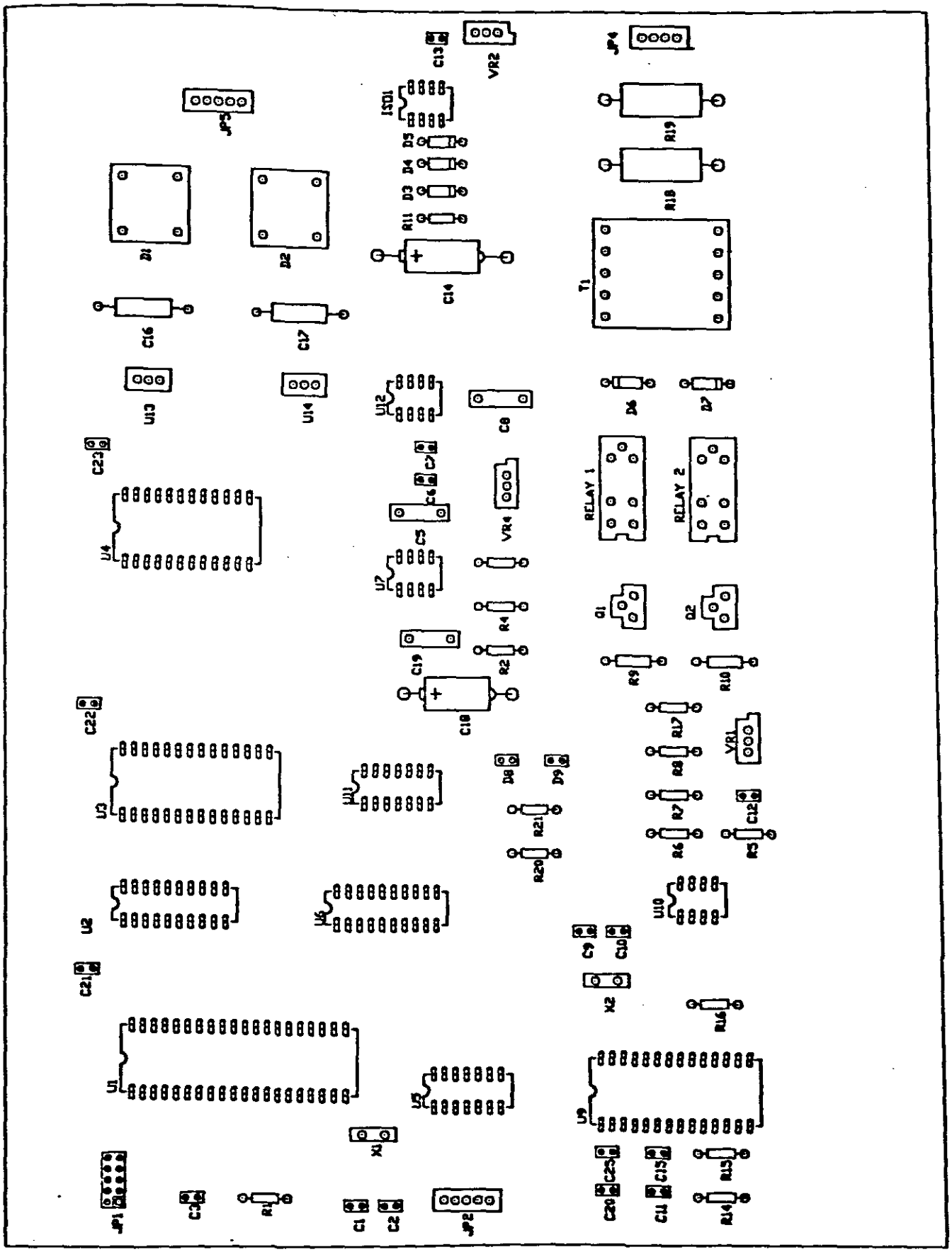
A special word of thanks to my wife, Fiona, for her encouragement throughout, especially when "things went wrong".

APPENDIX A

VALUE	FUNCTION	PURPOSE
01H	OR	SWITCH $\overline{RTS}$ OFF
F7H	AND	SWITCH $\overline{RTS}$ ON
02H	OR	SWITCH RING INDICATOR OFF
FDH	AND	SWITCH RING INDICATOR ON
04H	OR	SWITCH $\overline{DTR}$ OFF
FBH	AND	SWITCH $\overline{DTR}$ ON
08H	OR	NO FUNCTION
F7H	AND	NO FUNCTION
10H	OR	SWITCH IN RELAY ONE
EFH	AND	SWITCH OUT RELAY ONE
20H	OR	NO FUNCTION
DFH	AND	NO FUNCTION
40H	OR	NO FUNCTION
BFH	AND	NO FUNCTION
80H	OR	SWITCH IN RELAY TWO
7FH	AND	SWITCH OUT RELAY TWO

VALUES USED BY THE SERIAL TO PARALLEL CONVERTER

APPENDIX B



RELATIVE COMPONENT LAYOUT

APPENDIX C

VOICELESS TELEPHONE — Revised: September 12,1991

Bill of Materials — September 23,1991 — 12:51:29 — Page 1

Item	Quantity	Reference	Part
1	1	U1	8032
2	2	U2,U6	74ALS573
3	1	U3	2764
4	1	U4	6116
5	1	U5	74164
6	1	U7	741
7	1	U9	AM7911
8	1	ISO1	OPTO ISOLATOR
9	1	T1	TRANSFORMER
10	1	U10	LF353
11	1	R4	100K
12	2	C1,C2	33PF
13	3	R1,R2,R16	10K
14	1	VR1	47K
15	2	R5,R8	600
16	3	R6,R7,R17	22K
17	3	R9,R10,R13	2K2
18	2	Q1,Q2	BC107B
19	1	K1	RELAY 1
20	1	K2	RELAY 2
21	2	VR3,R15	1M
22	1	U11	74LS14
23	1	U12	LM567
24	1	VR4	3K9
25	1	U14	LM7805

COMPONENT LIST (a)

APPENDIX C

VOICELESS TELEPHONE — Revised: September 12,1991

Bill of Materials — September 23,1991 — 12:51:29 — Page 2

Item	Quantity	Reference	Part
26	1	U13	LM7905
27	2	D1,D2	BRIDGE
28	1	X1	11.059 MHZ
29	1	X2	2.45 MHZ
30	1	R14	100
31	2	C9,C10	27pF
32	1	C11	2.2nF
33	1	C12	0.1uF
34	1	C3	2.2uF
35	1	C5	0.01uF
36	7	C13,C20,C21,C22,C23,C24,C25	10uF
37	1	D5	1N4148
38	2	R18,R19	10R 5W
39	2	D6,D7	4.7V
40	1	C15	47nF
41	2	C16,C17	470uF
42	1	JP1	DISPLAY
43	1	JP2	KEYBOARD
44	1	JP4	TEL.LINE
45	1	TB1	70V
46	3	C8,C18,C19	0.47uF
47	1	C6	1uF
48	2	C7,C14	2.2uF
49	1	R11	5K6

COMPONENT LIST (b)

APPENDIX C

VOICELESS TELEPHONE — Revised: September 12,1991

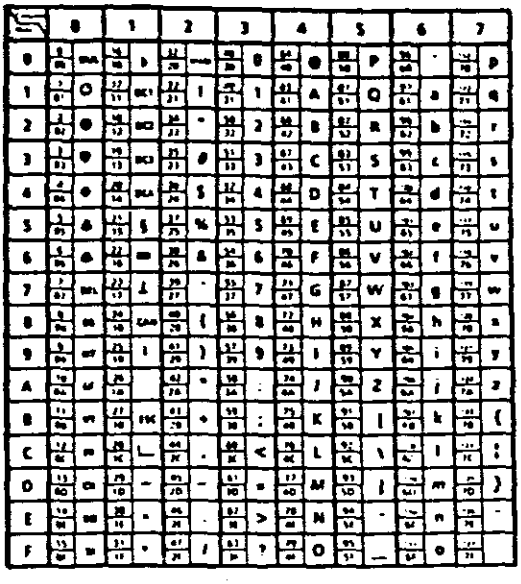
Bill of Materials — September 23,1991 — 12:51:29 — Page 3

Item	Quantity	Reference	Part
50	2	D3,D4	27V
51	1	JP5	POWER
52	2	R20,R21	1K
53	2	D8,D9	LED
54	1	JP6	AA STRAP

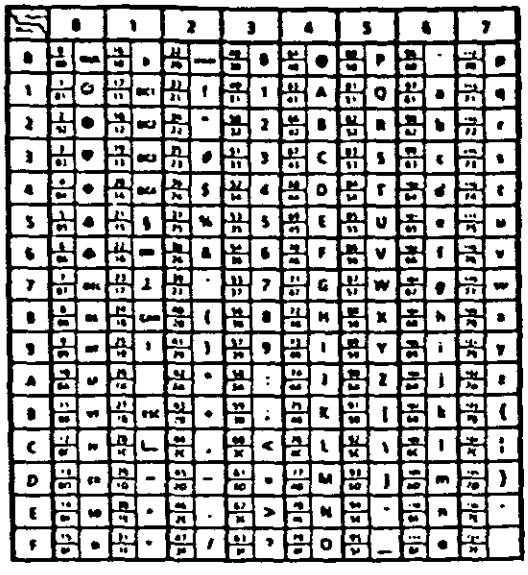
COMPONENT LIST (c)

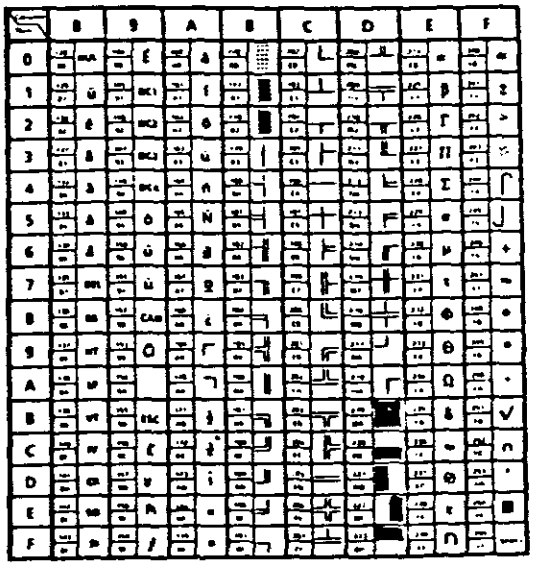
APPENDIX D

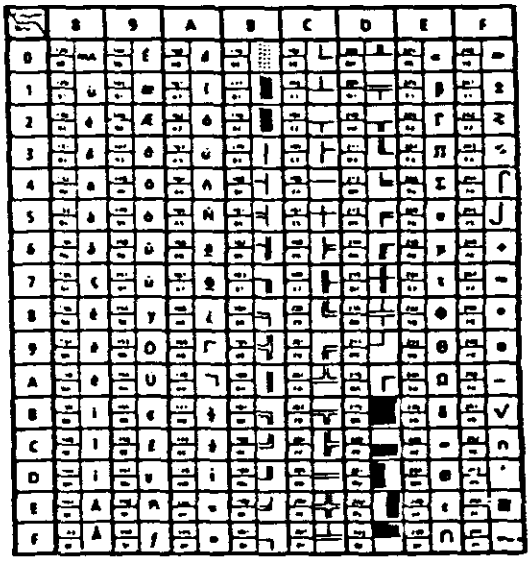
Character set 1



Character set 2







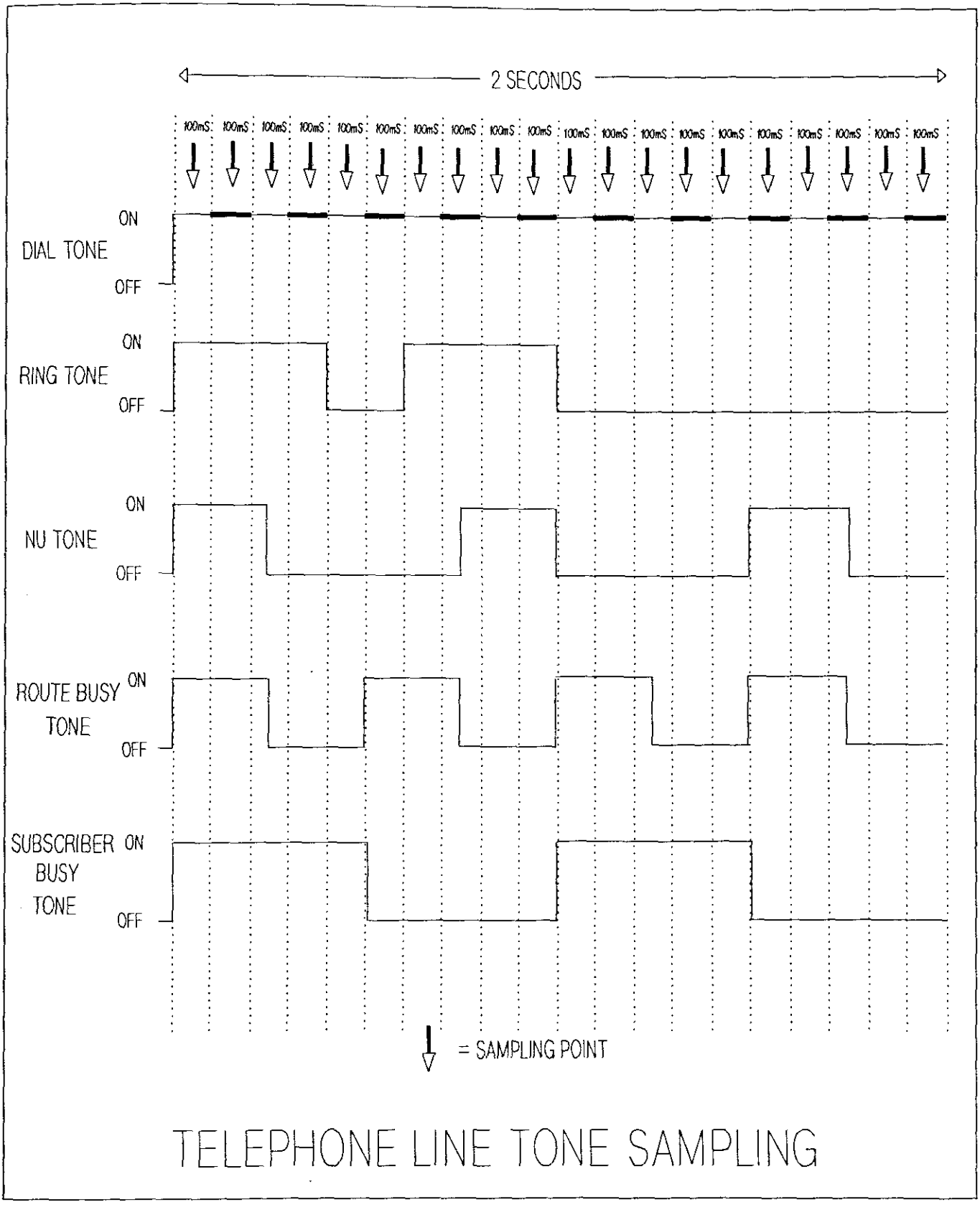
ASCII CHARACTER SET

APPENDIX E

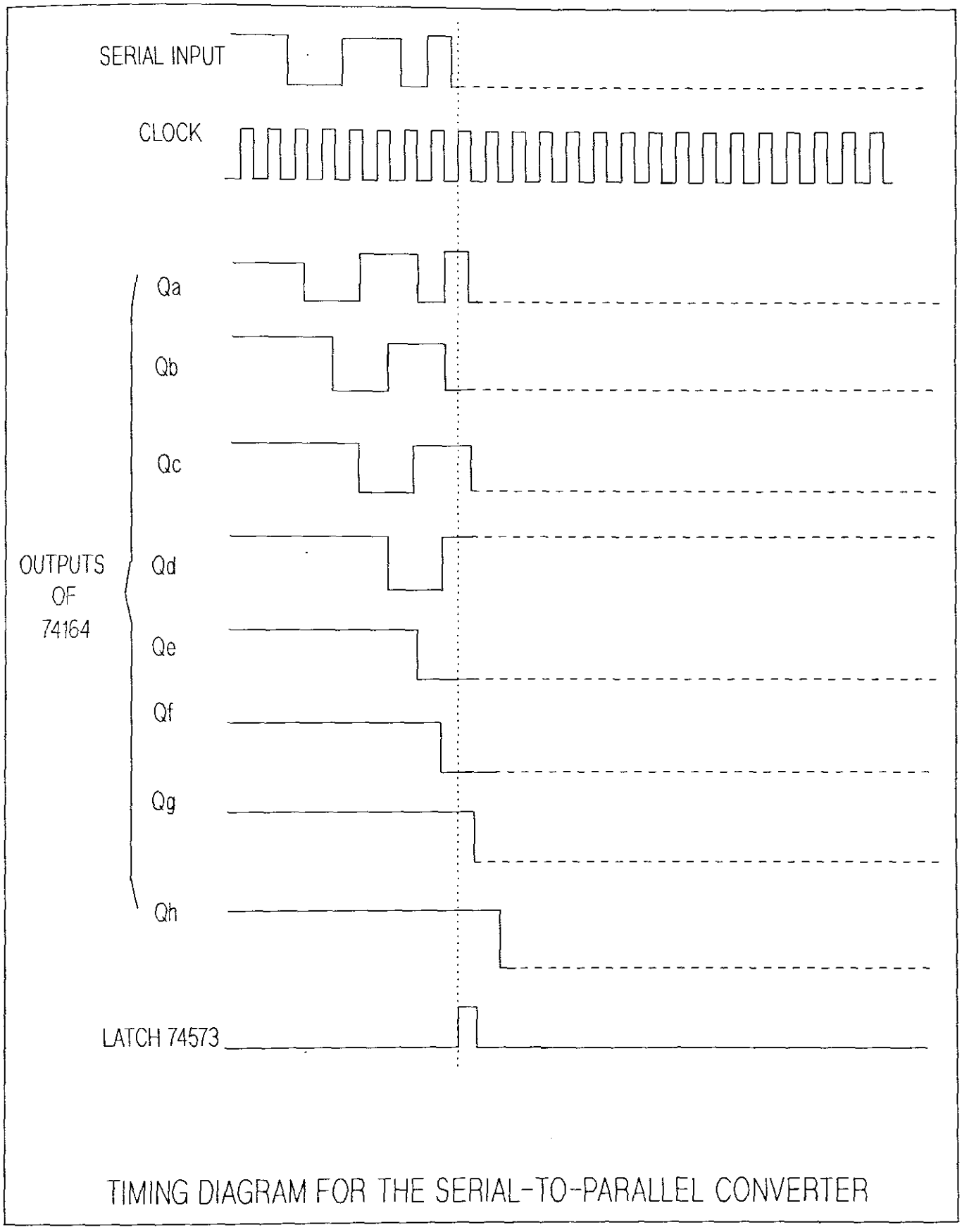
PIN NO.	SYMBOL	NAME AND FUNCTION
1	Vss	Ground
2	Vdd	Power supply (logic) = 5V
3	Vo	Contrast adjustment voltage = 0V
4	RS	Register select - L = instruction code input H = Data input
5	R/W	Read/write - L = Write data to LCD H = Read data from LCD
6	E	Enable
7	D0	I/O data LSB - not used
8	D1	I/O data 2nd bit - not used
9	D2	I/O data 3rd bit - not used
10	D3	I/O data 4th bit - not used
11	D4	I/O data 5th bit - LSB 4-bit mode - used
12	D5	I/O data 6th bit - 2nd bit in 4-bit mode - used
13	D6	I/O data 7th bit - 3rd bit in 4-bit mode - used
14	D7	I/O data MSB in both 4 and 8-bit modes - used

LCD PIN DESCRIPTION

APPENDIX F



TELEPHONE LINE TONE SAMPLING



TIMING DIAGRAM FOR THE SERIAL-TO-PARALLEL CONVERTER

APPENDIX H

(MSB) (LSB)

GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
Timer 1				Timer 0			

TMOD

(Timer/Counter Mode Control Register)

GATE When set, timer/counter X is enabled only when INTx pin is high and TRx control pin is set. When cleared, timer/ counter x is enabled whenever TRx is eabled.

C/T Timer or Counter Selector cleared for timer operation (input from internal system clock), set for counter operation (input from Tx input pin).

MODE 0: MCS-48 Timer
1: 16-Bit timer/counter. THx and TLx are cascaded.
2: 8 Bit-auto-reload timer/counter.
3: (Timer 0) TL0 and TH0 form two 8-bit timers. (Timer 1) Timer-counter 1 stopped.

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	1E0	IT0

TCON

(Timer/Counter Cntrol Register)

TF1 Timer 1 overflow flag.
TR1 Timer 1 run control bit.
TF0 Timer 0 overflow flag.
TR0 Timer 0 run control bit.

IE1 - Interrupt 1 edge flag.
IT1 - Interrupt 1 type control bit.
IE0 - Interrupt 0 edge flag.
IT0 - Interrupt 0 type control bit.

(MSB) (LSB)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SCON

(Serial Port Control Register)

SM0	SM1	Mode	
0	0	0	Shift register
0	1	1	8-bit UART
1	0	2	9-bit UART(fixed baud rate).
1	1	3	9-bit UART(variable baud rate).

SM2 - *Enables communication in mode 2 and 3.*

REN - Enables serial reception.
TB8 - Contains 9th data bit transmitted in modes 2 and 3.
RB8 - Contains the 9th data bit received in modes 2 and 3.
TI - Transmit interrupt flag.
RI - Receive interrupt flag.

(MSB) (LSB)

EA	X	ET2	ES	ET1	EX1	ET0	EX0

IE

(Interrupt Enable Register)

EA - Disables all interrupts.
X - Reserved.
ET2 - Enables/disables Timer 2 overflow interrupt.
ES - Enables/disables serial port interrupt.

ET1 - Enables/disables Timer 1 overflow interrupt.
EX1 - Enables/disables external interrupt 1.
ET0 - Enables/disables Timer 0 interrupt.
EX0 - Enables/disables external interrupt 0.

SPECIAL FUNCTION REGISTERS

APPENDIX I

STEP ONE: A key is pressed {Example: key 4 - Ascii character "3" - refer to figure 3.7 on page 30}.

KEY NUMBER	MAKE CODE	BREAK CODE
1	0E	F0 0E
2	16	F0 16
3	1E	F0 1E
4	26	F0 26
5	25	F0 25
6	2E	F0 2E
7	36	F0 36
8	3D	F0 3D
9	3E	F0 3E

SCAN CODE FOR KEY 4

PART OF THE IBM KEYBOARD SCAN CODES (SET 2)

FIGURE A

STEP TWO: The scan code for that key, 26H (see figure A), is sent to the processor and stored in the TH0 by the EX0INT routine (refer to page 70).

STEP THREE: The variable 'C' is made equal to the value of the TH0 register (ie. the scan code value).

STEP FOUR: The scan code is then checked for special function operations (eg. F1,F2,etc.) by the CHCHK routine.

STEP FIVE: If the shift flag,'S' ,is equal to 1, 80H is added to the value of 'C'.

STEP SIX: If the scan code DOES NOT perform a special function, ie. IGNORE = 0, the scan code is converted to the corresponding Ascii character (33H), which is determined by adding the offset (26H), to the start position of the mapping table (figure C).

STEP SEVEN: The Ascii character is then written to the transmit buffer and the LCD display.

```
if(read_bit_and_clear(F0_bit))/* If key pressed,        */
    {
    output(IE,0x00);          /*  Disable all interrupts */
    clear_bit(RS);            /*    select cont          */
    clear_bit(RW);            /*    write to display     */
    c=input(TH0);             /*    c= KB input          */
    chchk(c);                 /* Check for function key  */
    if (s==1)                 /*    If shift key -       */
    c=c+0x80;                 /*    add 80H to "c"       */
    if (ignore==0)            /*  If non-function        */
      {                       /*          key -          */
      tbuf[tcnt]=dec[c];      /*    Place input in       */
      tcnt++;                 /*    transmit buffer      */
      display(dec[c]);        /*      and display.       */
      scrnset();              /*    Setup display        */
      if (tcnt==78)           /*    If buffer = 78  -    */
          buffull();          /*    indicate buffer full */
      }
    }
```

PART OF TXRX ROUTINE

FIGURE B

```
CONST CHAR DEC[ ]={
0x7f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,
0x3f,0x2c,0x3f,0x3f,0x3f,0x3f,0x3f,0x3f,0x71,0x31,0x3f,0x3f,0x32,
0x7a,0x73,0x61,0x77,0x32,0x3f,0x3f,0x63,0x78,0x64,0x65,0x34,0x33,
0x3f,0x3f,0x20,0x76,0x66,0x74,0x72,0x35,0x3f,0x3f,0x6e,0x62,0x68,
0x67,0x79,0x36,0x3f,0x3f,0x3f,0x6d,0x6a,0x75,0x37,0x38,0x3f,0x3f,
0x2c,0x6b,0x69,0x6f,0x30.....}
```

00H

1AH

26H

PART OF MAPPING TABLE

FIGURE C

EXAMPLE OF KEYBOARD INPUT

APPENDIX J

PRELIMINARY RESEARCH

Before commencing work on the "Voiceless Telephone", extensive research was undertaken to establish what research had been done in the field of communication apparatus for the handicapped; or if anything similar was, or had been investigated.

Searches were made on the SABINET, CSIR, HSCR and DIALOG databases under the following catagories;

- Deaf Aids
- Deaf Apparatus
- Communication for the Deaf
- Communication - General
- Telephone and telephone equipment

The research proved fruitless, as no reference was made to anything closely related to that of the "Voiceless Telephone".