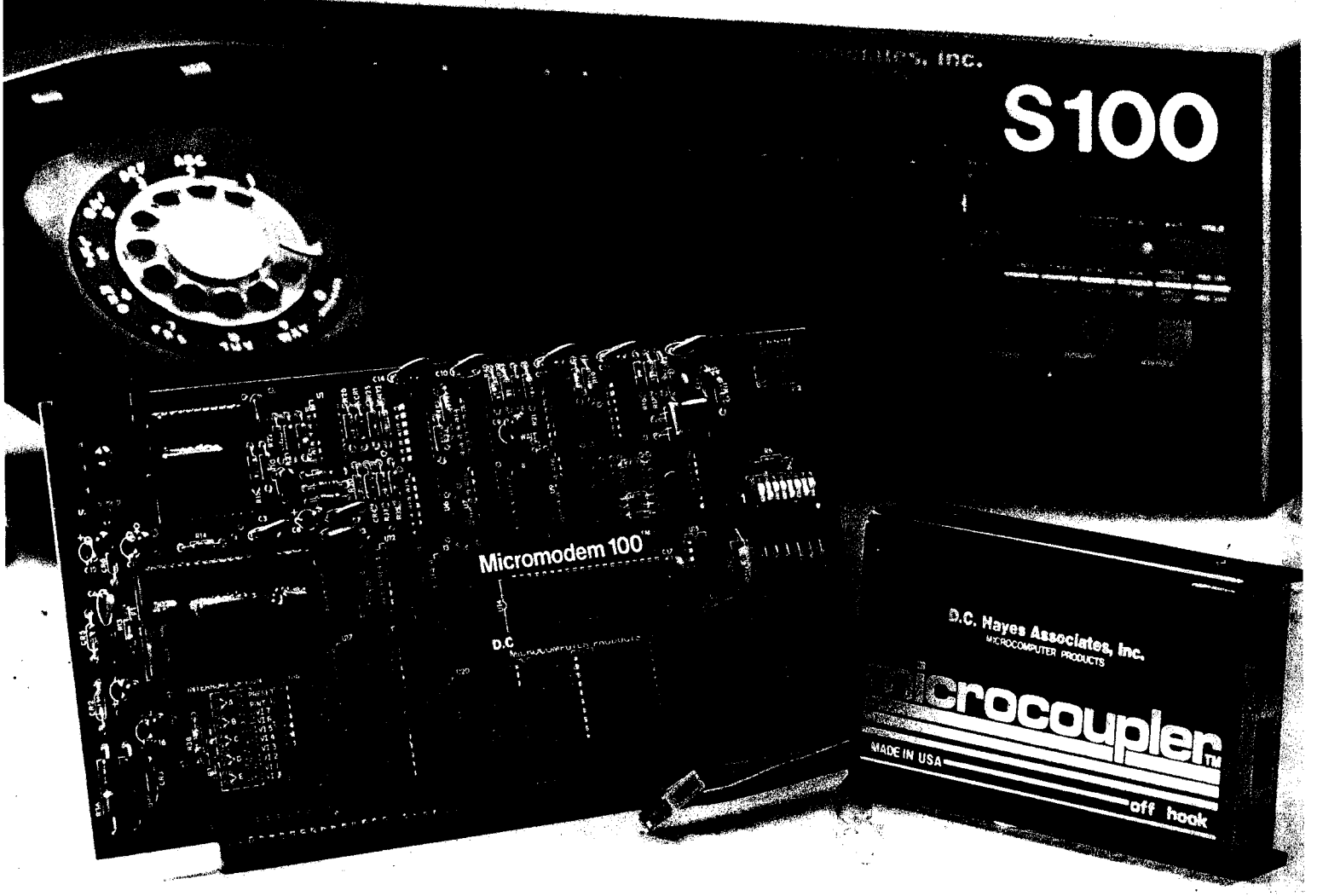
D.C. Hayes Associates, Inc.

MICROCOMPUTER PRODUCTS

MICROMODEM 100™

with the MICROCOUPLER™



OWNER'S MANUAL

MICROMODEM 100 OWNERS MANUAL

Written by Dale A. Heatherington and Donald J. Hyde

CONTENTS

Printed in USA

NOTE:

1.0 BACKGROUND INFORMATION

The information in this chapter is not essential to using the Micromodem 100. But I believe that you will find it to be helpful and possibly even interesting. Data communications is a fairly complex topic, combining as it does aspects of several related technologies, each of which is a complex and interesting subject in itself.

Data communications is a very confusing subject for most people, including data processing professionals. The subject in inherently complex, and the terminologies are often misleading. This is partly because of the mix of data-processing, electronic, and telephone technologies. Frequently the same words mean different things in the different fields, and when words from the different fields are combined to describe concepts which combine the different fields, they often carry a built-in confusion factor.

I have chosen several of the most important and most confusing topics and have attempted to describe them in plain English. I hope that I have succeeded at least to some extent.

1.1 Compatibility With the Bell System 103 Modem

The Micromodem 100 is designed to be completely compatible with the transmission frequencies and modulation techniques of the Bell System (Western Electric) model 103 low-speed modem. The Bell System 103 (and its various equivalents) is by far the most widely-used modem in North America. It is used by virtually all time-sharing systems and dial-up data access systems as their standard mode of access. This popularity is due partly to the relative simplicity of the 103's FSK modulation technique and the reasonable cost of the circuitry required to implement it and partly due to the fact that there are so many other 103-compatible modems already installed to talk to.

1.2 What IS a modem anyway?

Early in the history of computing (back in the dim distant days of the 1950's),when computers were still huge like dinosaurs and people were just beginning to discover their data processing power, it occurred to someone that you could do a lot of neat things if you had two computers with a wire between them, or a computer and a typewriter if they had a wire connecting them. When they started thinking about wires that ran across the country, they soon began to think about how nice it would be if they could use all the phone wires that were already there.

Unfortunately, as we all know, computers talk digital and telephones are designed to carry the human voice. And the Bell System has lots of smart people who've been working for years and years building black-magic widgets of all kinds that mash and tear up those voice signals in all sorts of inconceivable ways to get them into long-distance wires in the most efficient possible way.

They spent decades researching the human voice so that they knew exactly how much they could mash and distort it and still have it come out recognizable at the other end of the line. If you could send digital data on the phone, who knows what all those widgets might do to it before it got to the other end of the line.

The phone company was worried that digital signals might hurt their various widgets, or might interfere with normal voice signals, so they weren't very encouraging at first.

But the problem did get some study, and some experiments were carried out (mostly by the Bell System). It seems that one thing all the widgets are careful not to disturb is the frequencies of the tones making up a voice. They may distort the amplitude or the phase, but they are careful not to distort frequencies. So a device was built which encoded digital data consisting of ones and zeroes into a signal containing different frequencies for ones and zeroes. This was called a modulator. That was pretty easy. The hard part was building something to un-do the modulating, and turn the tones back into digital ones and zeroes. Suffice it to say that they did figure out a way. They called the un-doer a demodulator.

It turned out that the modulator and the demodulator worked pretty good (better than most people expected). So the phone people put one of each into a box, and started to sell it as data transmission service. Like most engineering types, they weren't at their best with words, so they just took pieces of the words for the parts and stuck them together to make a word for the box. MOdulator + DEModulator = MODEM, thus the modem was born.

Well, a lot of things have happened since those distant days when dinosaurs first spoke to each other over the phone. A lot of research was done on modems. They got faster. And they got bigger and more expensive. In fact, the speed and price have tracked very closely, and there is a common rule of thumb that modems cost about a dollar a baud. But that didn't matter much because the computers they worked with were even bigger and more expensive.

Things happened even faster when the Supreme Court decided to allow competition in the modem business.

At first the phone company insisted that they had a legal monopoly in the modem business because modems were part of the telephone system and they had a legal monopoly to run the phones. Nobody argued. After all, it was pretty obvious that a modem had to be wired up to the phone line, and that certainly made it part of the phones didn't it? But what about the computer wired to the modem? Certainly it wasn't a telephone and the phone company couldn't claim a monopoly on making computers, too. Then along came the acoustic coupler, a modem that wasn't wired up to the phone line. It talked into the phone just like a human, making noises into the mouthpiece, and listening to the noises coming from the earpiece.

The phone company insisted that an acoustic coupler was also a violation of their protected monopoly. The Carterfone company, which built an acoustically-coupled device for use with two-way mobile radios, went to court

after the phone company put them out of business by telling their customers that the acoustic coupler was illegal and threatening to disconnect their telephones if they didn't stop using it.

The court eventually decided in favor of Carterfone (awarding them considerable damages, which helped them to get back in business), and ruled that acoustic couplers were legal. More recent decisions have broadened the rules for interconnect, giving the FCC the power to license devices for use on the telephone network in much the same way that they have long licensed radio transmitters, and limiting the phone company's protected monopoly to running the network which connects the phones together.

Today the dinosaurs, though still very much alive and very much still with us, are giving up center stage to the much smaller, more advanced microcomputers. Small, inexpensive computers need small inexpensive modems, so the old reliable 103-style modem is now even more popular than ever.

1.3 Baud Rates

The term baud often confuses people. And with good reason. It means different things depending on who is using the term, and when. In its narrowest technical definition, a baud is defined as being a measure of the rate at which signals are transmitted through a communications channel, with one baud corresponding to a rate of one signal element per second.

Well, that sounds pretty straightforward, but what's a signal element? With digital data it seems pretty reasonable that a signal element should be the same thing as a bit so that 300 baud would be the same thing as 300 bits per second. Right!? Well, sometimes.

Baud rate is concerned with the stuff (like tones) going down the communications channel (in this case a phone line), and NOT with the stuff (like ones and zeroes) we're turning it into at the other end. In a 103-style modem, each possible change from one frequency to the other is a signal element. Since that is how we code a single bit, one baud equals one bit per second. But the phone line has a very finite rate at which it can carry signal elements. The rate is related to the frequency range the line can accomodate and the modulation technique (as well as the error rate one is willing to tolerate). In general, 1200 baud is about the highest the phones will accomodate with any degree of reliability. So the fastest that a modem can be is 1200 baud (unless you're using some other kind of telephones).

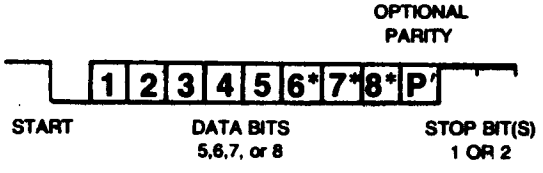
Well where do 4800-baud modems come from then? Well, the answer is that they aren't 4800-baud modems at all. They are 4800 bit-per-second modems. They do it by very cleverly encoding 4 bits of digital data onto a single signal transition (corresponding to a single cycle of a 1200 Hz carrier tone). As you can imagine, it is quite a trick encoding them that way, and even more of one to get them back out at the other end. That's why they're so expensive -- it ain't easy.

But that's not the only confusing part. As data communications users, you

and I don't care how many funny little bauds go down the phone line. In fact we really don't care about the bits even. We are trying to get some bytes moved around, but we'll probably settle for moving ASCII characters.

So how fast can I send characters with a 300 bit-per-second modem? Usually about 30 characters per second, but it can vary. You see, in order to be able to pick the characters out at the receiving end, we have to put on some extra bits to tell where the character begins and ends, etc. So our 7-bit ASCII character is accompanied by one start bit, one stop bit, an optional parity bit for error-checking, and an optional extra stop bit (I don't quite know why, it just slows things down). Thus our ASCII character could get from 2 to 4 extra bits as travelling companions.

The most common usage is one each of start bits, stop bits, and parity bits. This works out nicely because that means 10 bits per character which makes the character rate at 300 baud simple to calculate as 30 characters per second.



Normally, there is no good reason to transmit any slower than you have to (you just get a bigger phone bill), except that it won't work if you send faster than whatever's at the other end can receive it. There are an awful lot of model 33 Teletypes still kicking around. They can send and receive only at 110 baud. Between all those TTY's and the machines built to talk to them, there are still a lot of 103-type modems connected to things that only run at 110 baud.

1.4 Half- and Full- Duplex

This is another very confusing aspect of data communications, and like baud rates is mostly confusing because the terms have been used so loosely and with so little regard for their original very narrow technical meaning.

A communications link which carries data from one point to another in only one direction is said to be a simplex link. Most of the devices (such as radio) which are used for communication are basically simplex devices. To get a two-way communication link, we have to use two simplex links, one going in each direction. This is called a duplex communications link. The telephone is a duplex communications link. The people at both ends of the phone can hear each other. They can even both talk at the same time (except on long-distance sometimes).

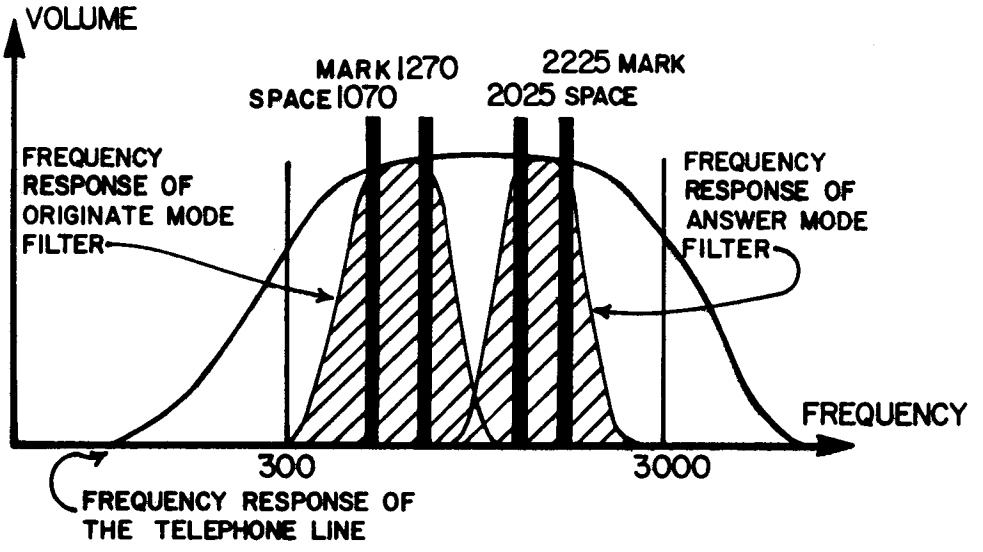
If we have a simplex link which can be turned around, then we can have two-way communications with only one communication path. This is called half-duplex, because it has the effect of a duplex link but with only half as much stuff. CB radio is half-duplex. When one person finishes talking, he has to say "over" or "10-4" or something so the person at he other end knows it's his turn to talk. If they both try to talk at the same time, neither one can hear anything.

To have a full-duplex radio channel, each person would have to have a separate transmitter and receiver, and both people's transmitters would have to run all the time. In order for the two transmitters not to interfere with each other, they would have to use different channels. So full-duplex takes two channels.

Full-duplex communications is easier to use than half-duplex (at least for people), because it is more like normal face-to-face communication. So, even though it takes more stuff to make it work, the phones are built to work full-duplex. The phone in your house has a clever transformer called a hybrid which allows the two signal paths going in opposite directions to share the same pair of wires to the central office. But once it gets to the office, there are more circuits that sort them out, and they are kept that way until they leave another central office on their way to someone's phone.

Long-distance phone circuits always occur in pairs -- one circuit going in each direction. On calls that are over a few hundred miles, there is a problem with a full-duplex link. It takes the signals a finite length of time to get from one end to the other. When they arrive, part of the signal goes back into the phone and is sent back where it came from. You get an echo. To fix that, there are gizmos called echo suppressors in the lines. An echo suppressor is a voice-controlled switch that makes the phone line really work half-duplex, but it seems to be full-duplex because the echo suppressors turn around in the space between two syllables.

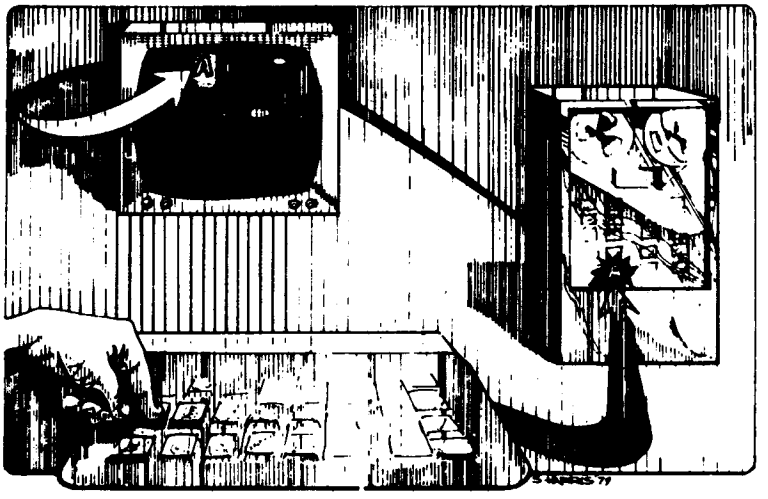
Well, the 103-type modem was designed to be able to take advantage of all those full-duplex communication paths. The designers of the 103 divided the frequency band of the telephone into two narrower bands, and assigned one for sending in one direction and the other for sending in the other. This gave them two channels to work with (like using two CB channels). It worked. Data could go in both directions at the same time. It worked on long distance, and the modems weren't at all bothered by the echos, because the modems were designed to talk on one frequency and listen on the other. They didn't care about the echos because they couldn't hear them.



The only problem was those echo suppressors. They kind of messed things up. Fortunately, someone had already thought of that and put a special disable circuit into the echo suppressors, that made them turn off if they heard a tone in a certain band. One of the tones used by the 103 just happens to correspond to this signal. So as soon as an echo suppressor hears the carrier from a 103 modem, it turns itself off.

Not all modems are full-duplex. In fact, until recently the 103-type was the only full-duplex modem. Most of the faster modems are half-duplex. This makes them more complicated to use beause then the computers at both ends have to say "over" or "10-4" or something and all that. That, of course means a lot of complicated programming.

One nice thing about a full-duplex modem is that you can echo the characters. Virtually all time-sharing and data access systems echo each character back as it is received, and the character is not displayed or printed on the terminal until it has been for a full round trip to the distant computer and back. This gives the person typing a clear indication whether what he typed was received correctly. What he sees is what the computer sees. If a character gets garbled in the phone line, it shows up garbled on his terminal. If it gets lost completely, then nothing shows up on his terminal at all. This technique has a very catchy name, echo-plex.



It is because of this practice that most terminals have a switch marked full- or half-duplex. If the terminal is connected to a half-duplex modem, echo-plex won't work because the computer can't send back the characters it receives. Some systems don't use echo-plex even when they can because it introduces a slight delay which some people find objectionable. In this case, the terminal must display the characters that it sends to the computer. This is all a full-/half-duplex switch does on any terminal. Often the same switch appears on modems (especially acoustic couplers) mostly because it's cheap and looks impressive.

1.5 Ringing and Dialing

Ringing and dialing are two aspects of what the telephone company calls signaling. Signaling is the process by which a connection is established on the switched (dial) telephone network. As is the case with many commonplace things, telephone signalling is much more complicated than it looks.

First let's consider what has to happen when you place a call. The dialog looks something like this:

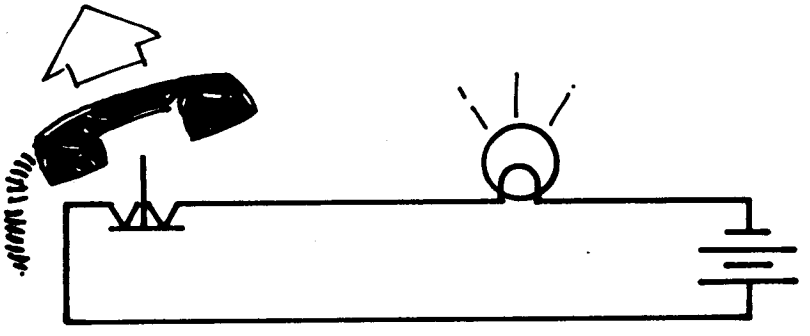
YOU	PHONE COMPANY
I want to make a call.	
	OK. Who do you want to call?
My girlfriend.	
	OK. I'll let her know you want to talk.
	I'm sorry, she's talking to someone else.

Now, that's not such a complicated dialog, but there are a few difficult limitations. For one thing, as we all know, there is seldom a person at the phone company, but some kind of a machine.

Everything that we say to the phone company has to be simple and exact enough for a machine, and everything the phone company says to us has to be simple enough that a machine can make the sounds. Whatever we do, we have to do it with just the two wires that connect us to the central office. And most limiting of all, we have to do it all with technology that was avaialable in 1894, because that's when the dial telephone was invented.

Let's start simple. How does the phone company find out that we've picked up the phone to make a call? Your telephone has a switch on which the receiver normally rests when not in use (the switch hook). When you lift the receiver, this switch establishes a connection between the two wires and allows current to flow from a battery at the central office. This current can be detected at the central office by a light bulb in series with the line, or by a relay.

fig 11

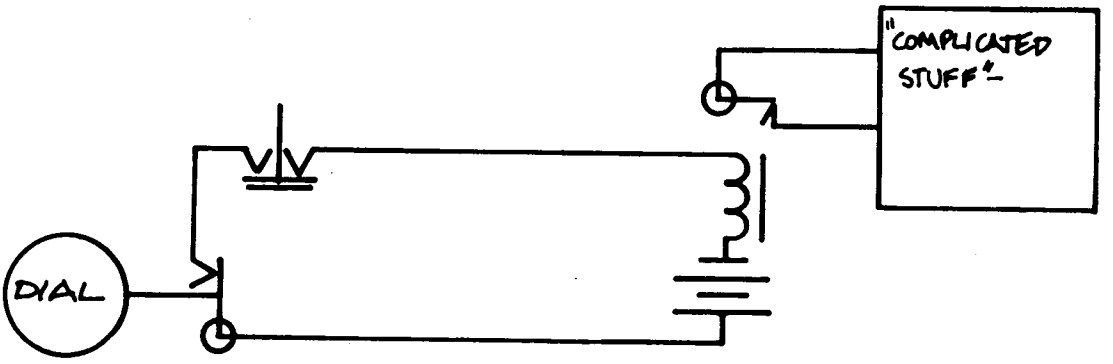


To tell you that it is listening, the central office machinery connects the line to a generator which produces an audible dial tone in your receiver.

To tell the machine who you want to talk to, you dial that person's telephone number. The dial in your phone has a switch that is in series with the switch hook. This switch momentarily breaks the connection between the two wires. When the connection breaks, it causes the relay at the central office to drop out momentarily.

The dial makes a series of these pulses very rapidly. The number of pulses corresponds to the digit you have dialed, from 1 to 10 pulses for the digits 1 to 0. At the central office these pulses are counted by Strowger relays and other more recent devices.

fig 12

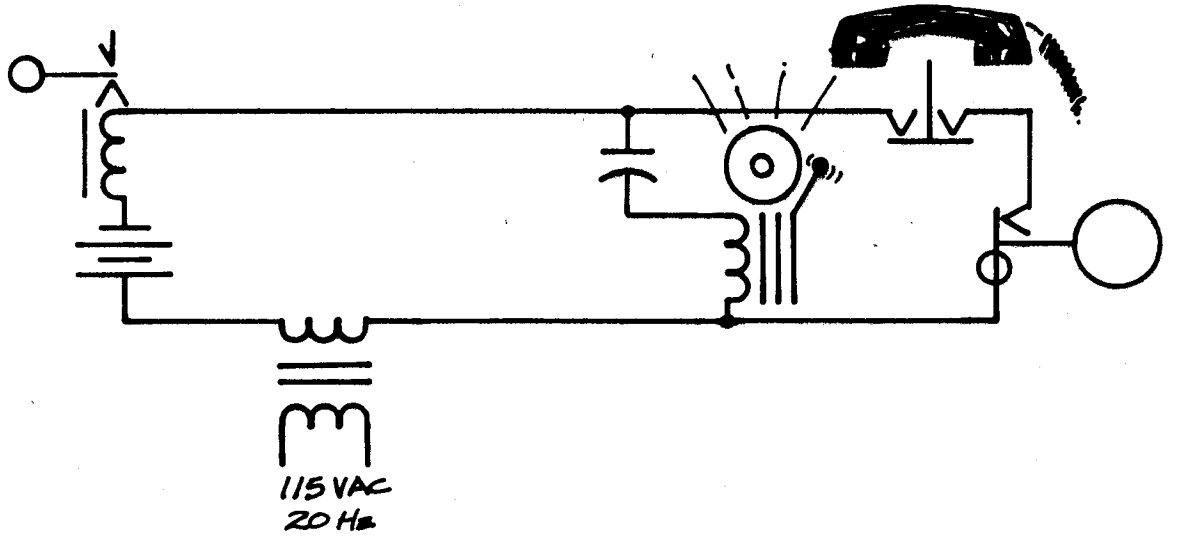


Once you have dialed, a whole bunch of complicated machines that I won't attempt to describe find the pair of wires corresponding to the number you have dialed.

If the line is busy, its phone will be off hook and the relay connected to it will be closed. The machine will then connect your line to another generator which makes a busy signal.

If the line is not busy, then it will connect that line to yet another generator. This one puts out 115 volts AC at 20 Hertz. Your friend's telephone has a bell which is connected to the line via a capacitor. The capacitor prevents the DC current from the battery from flowing through the bell, but allows the AC ring signal to pass through, thus making a loud noise.

When your friend picks up his phone, his switch hook makes connection, drawing current from the battery and activating the relay connected to his line. This causes the ring generator to be disconnected and stops the ringing.



Over the years, many refinements and variations have been added, including tone dialing, in which tones are used instead of the current pulses I have described. Tone dialing may eventually replace pulse dialing, but now and for many years to come, all automatic telephone exchanges accept pulses even if they are designed for tone dialing. Many older exchanges are able to accept only pulse dialing, so for now, pulse dialing is the only universally usable technique.

Well, if you understand everything this far, you can consider yourself to be a minor expert on telephones and data communications. And as the owner of a Micromodem 100, you are now equipped to use that knowledge to advance the state of the art by discovering and developing new applications. Good luck.

2.0 FUNCTIONAL DESCRIPTION

The Micromodem 100 is an S-100 bus compatible device which combines the functions of a serial interface, modem, and automatic dialer on one card. Included with the Micromodem 100 is the Microcoupler. This FCC registered device couples the Micromodem 100 directly to the telephone line without the use of a phone company supplied data access arrangement (DAA). Operation of the Microcoupler is detailed in section 3.

The Micromodem 100 has three main functional blocks. They are the S-100 bus interface, serial interface, and modem. They function together to provide the user with a quality low speed data communication system capable of communicating with other Bell 103 style modems at baud rates from 30 to 300. Operation of the Micromodem 100 is controlled with software. Baud rate selection, dialing, answering calls, originate/answer mode, character formats, etc. are all set by writing data to the Micromodem 100 control registers. See section 5 for details on these registers.

2.1 S-100 INTERFACE

The Micromodem 100 uses the low 8 address lines to decode four I/O addresses from a possible 256. A2 through A7 are used to select the base address while A0 and A1 select 1 of 4 registers for input or output operations. PDBIN is anded with SINP and the output of the base address decoder to provide a read strobe. Data is gated onto the bus during this time. The data source is determined by address line A0. When A0 is low modem received data is read. When A0 is high data from the status register is gated onto the bus. PWR is anded with SOUT and the output of the base address decoder to provide a write strobe. A0 and A1 are used to determine which of the 4 registers will be written into.

		READ ONLY REGISTERS	WRITE ONLY REGISTERS
A0	A1		
0	0	receive data	transmit data
0	1	status	control reg. 1
1	0	not used	control reg. 2
1	1	not used	control reg. 3

2.2 TIMER

A 50 millisecond hardware timer is provided so that software timing loops can be eliminated. It consists of a 555 timer chip wired as a non-retriggerable one-shot. Any output to control register 3 will start the timer. Its status may be read in the status register as bit 5. This bit goes to zero when the timer is started and returns to a one after 50 MS have elapsed. This timer bit is also avalable at the interrupt option socket as a low true open collector signal. This interrupt signal can be enabled or disabled under software control by changing bit 5 in control register 1. A 1 enables timer interrupts while a 0 disables them.

2.3 SERIAL INTERFACE

Serial to parallel and parallel to serial conversion is done with the popular AY-5-1013A Universal Asyncronous Receiver Transmitter (UART). The transmit and receive data registers as well as the low 5 bits of control register 1 and the status register are physically located in the UART. This 40 pin chip handles the job of serial-parallel conversion, adding start and stop bits, and generating and checking parity. The serial baud rate is controlled by the baud rate generator.

2.4 BAUD RATE GENERATOR

The baud rate generator provides a square wave clock at 16 times the desired baud rate for the UART. The UART internally divides this clock by 16. The frequency is derived from a 4 MHZ crystal oscillator divided down by U9 and U10. The division ratio of U9 can be varied to provide any baud rate from 30 to 300 baud. 300 baud or a lower user defineable baud rate can be selected under software control by changing bit 0 in control register 2. See section 4.5 for details.

2.5 POWER SUPPLY

The power supply provides 3 regulated voltages for the digital and analog circuits. Two 12 volt zener diodes, CR2 and CR3 , supply +12 and -12 volts for the analog circuits. The 7805, a 3 terminal voltage regulator, supplies +5 volts for the digital logic. The Micromodem 100 pulls about 25 MA from the S-100 +16 and -16 volt busses and 250 MA from the 8 volt bus. The Microcoupler is powered from the 5 volt supply and draws about 55 MA.

2.6 MODEM

There are four main parts in the modem section. They are: modulator/demodulator, transmit filter, receive filter, and carrier detect circuit.

The modulator/demodulator is a MC14412 CMOS chip. Serial data from the UART is converted to frequency shift keyed (FSK) data by the modulator. It is a digitally synthesized sine wave. When in ORIGINATE mode a logic 1 from the UART produces a 1270 HZ tone and a logic 0 produces a 1070 HZ tone. When in ANSWER mode a 1 produces a 2225 HZ tone and a 0 produces a 2025 HZ tone.

The demodulator section of the MC14412 takes the received FSK square wave from the limiter and converts it to serial mark/space data for the UART. In ORIGINATE mode a 2225 HZ tone is converted to a logic 1 and a 2025 HZ tone is converted to a logic 0. In ANSWER mode a 1270 HZ tone is converted to a 1 and 1070 HZ is converted to a 0.

In self-test mode the demodulator frequencies are switched to match the modulator. With an external connection, data can be looped back from the transmitter to the receiver for testing purposes.

The transmit filter is a 20 pin hybrid module which contains a 6 pole active band pass filter. Its purpose is to clean up the noise, harmonics, and modulation sidebands present in the synthesized sine wave produced by the modulator before it is sent to the phone line. In ORIGINATE mode it passes frequencies from 950 to 1400 HZ. In ANSWER mode frequencies from 1900 to 2350 are passed.

The receive filter ia a 40 pin hybrid module which contains a 10 pole active band pass filter. Analog signals from the phone line pass through the Microcoupler and are sent to this filter. The desired band of frequencies are amplified by about 20 DB (10 times) and the unwanted signals (noise and transmit carrier) are attenuated by 50 DB (300 times). The receive filter has 2 outputs, a limiter output and an analog output. The analog output is simply the input signal after being filtered and amplified. The limiter output is a 5 volt square wave with the same frequency as the input signal. It is sent to the demodulator where it is converted to serial mark/space data for the UART.

In ANSWER mode the receive filter passes frequencies from 1000 HZ to 1350 HZ . In ORIGINATE mode frequencies from 1950 to 2300 HZ are passed.

The purpose of the carrier detect circuit is to provide the program with a means of determining the presence or absence of a receive data carrier. This circuit is implemented with a quad OP-AMP, U1, and half of U2, a dual timer chip. In operation it "listens" to signals passed by the receive filter and sets the carrier detect bit in the status register and lights the LED if those signals are louder than -50 DBM. Noise pulses lasting less than 2 seconds will not activate this circuit because of the action of timer, U2. Only 2 seconds or more of steady tone will register as a valid carrier.

LOST CARRIER DISCONNECT

The lost carrier circuit uses the output of the carrier detect circuit and the on/off hook control to provide an indication that the carrier has been lost for 10 seconds. The main components in the circuit are filp-flop, U6, and timer, U2. The flip-flop "remembers" that a carrier has been present since going off-hook and enables the 10 second timer to start when the carrier is lost. If the carrier returns within 10 seconds the timer is reset, otherwise it times out and activates the lost carrier signal. This signal can be used to reset the CPU if the program crashes and can't poll the carrier detect status bit.

3.0 MICROCOUPLER

The Microcoupler is an FCC registered protective coupler. It's purpose is to protect the computer user and the telephone network from the harmful effects of high voltage spikes and excessive signal levels. It also insures that the telephone billing equipment will properly register and time the calls. The Microcoupler circuity is proprietary and, due to FCC regulations, not user serviceable. For these reasons no schematic is provided. However, to aid the user in trouble shooting his system the following functional description is provided.

The Microcoupler uses only the middle two wires in the four conductor modular telephone cable. These are the red and green wires and are called tip and ring. During the idle condition the telephone system supplies about 50 volts DC between these wires. During ringing 90 volts AC at 20 HZ is superimposed on the 50 volts DC. When a call is in progress the tip to ring voltage will be somewhere between 2 and 16 volts DC at 15 to 120 MA. The DC resistance measured between tip and ring on the Microcoupler is 137 ohms when off-hook (in use) and infinity when on-hook (idle). This measurement is made with the phone line disconnected.

A reed relay controls the on-off hook states and accomplishes the dialing function by pulsing on and off at a 10 HZ rate. The 90 volt AC ringing signal is detected with an optoisolator. The ring detect circuit requires at least 50 volts AC at any frequency between 15 and 70 HZ for 1 second or more.

The Microcoupler has an automatic transmit level control that maintains a constant -10 DBM output into 600 ohms with input levels ranging from -7 to +10 DBM. Inputs below -7 DBM will be passed with a constant 3 DB loss. The automatic level control has no effect on received signals. All transmit signals are sent through a 3 KHZ low pass filter with a 12 DB per octave rolloff.

Signals received from the phone line are passed through a duplexer. The purpose of the duplexer is to seperate the received signals from the transmitted signal. About 10 DB of transmiter rejection is achieved. The duplexer has an output impedance of 600 ohms on both the user and phone line sides. The receive gain is 0 DB when connected to a 600 ohm load or 6 DB when driving a high impedance load such as the Micromodem 100.

The interface between the Micromodem 100 and the Microcoupler is simple and straight-forward. All power, control, and audio signals are sent via a 10 conductor ribbon cable. Signals received from the phone line appear on pin 1. Signals to be transmitted to the phone line are applied to pin 3. Transmitted signals will also appear mixed with the received signals at pin 1 but they will be about 10 DB weaker because of the action of the duplexer. A TTL low level on pin 7 will cause the Microcoupler to go off-hook. Dialing is accomplished by pulsing this line at a 10 HZ rate. Pin 5 is the ring detector output. It is normally high and goes low during the telephone ringing periods. It will drive 1 LS-TTL load. A single 5 volt power source runs all the digital and analog

circuits. Pin 10 is +5 volts, pin 8 is ground. The Microcoupler has a timer which inhibits the transmit and receive audio for 2 seconds after going off-hook. This is for the benefit of the phone company billing equipment.

3.1 MICROCOUPLER INTERFACE CONNECTOR

PIN	NAME	FUNCTION
1	Receive data	Audio signals from the phone line appear on this pin. The output impedance is 600 ohms and the level depends on telco loss. -30 dbm typical.
2	not used	Key
3	Transmit data	Audio signals to be sent to the phone line are applied to this pin. It has a 47 K ohm input impedance. The maximum input level is +10 DBM. Minimum input level is -7 DBM for a -10 DBM level on the phone line.
4	not used	
5	Ring detect	Goes low during the telephone ringing signal. It will drive 1 LS-TTL load.
6	not used	
7	Off-hook	A TTL low level on this pin will cause the Microcoupler to go off-hook or answer an incoming call. Audio will be switched through on pins 1 and 3 2 seconds after this line is brought low. It presents 1 TTL load.
8	Ground	Logic and power supply ground
9	not used	
10	+5 volts	+5 volt power supply input. Current required is 55 MA.

3.2 MICROCOUPLER ELECTRICAL CHARACTERISTICS

FCC registration number	BI986H-62226-PC-E	
Ringer equivalance number	.4B	
On-hook impedance	DC	open circuit
	AC	25 K ohms at 20 HZ
off-hook impedance	DC	137 ohms
	AC	600 ohms
Maximum loop current	130 DC MA	
Maximum output level	-10 DBM into 600 ohms	
Frequency response	+/- 3 DB 300 to 3500 HZ	
Ring detector	frequency	15 to 70 HZ
	voltage	50 to 200 VAC
	resp. time	1 second
Power	5 volts DC at 55 MA	

INSTALLATION

4.1 Identifying the Parts

Your Micromodem 100 is packed with several necessary items. You should take them out of the box and verify that they are all present and in good condition. They are:

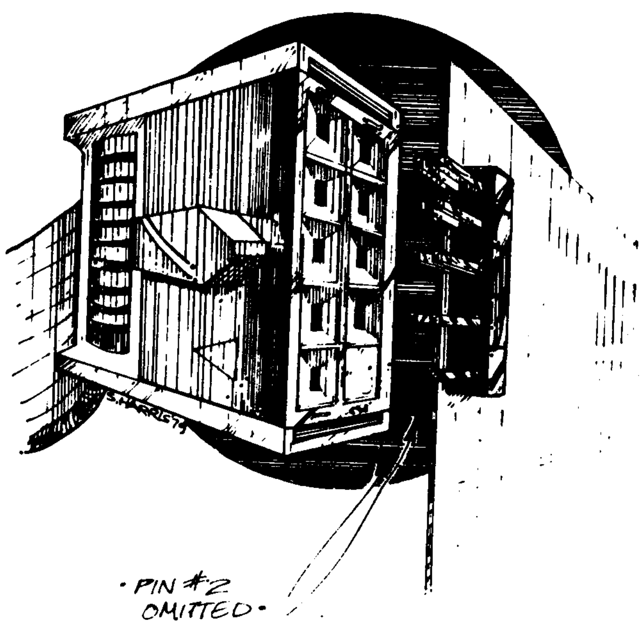
1) Micromodem 100 printed circuit board.
2) Microcoupler for attachment to the telephone line.
3) Ribbon cable to connect Micromodem 100 to Microcoupler.
4) Modular telephone cable to connect Microcoupler to telephone line.

fig 1

4.11 Installing the Micromodem 100 in the Computer

!!
------- MAKE SURE THE COMPUTER IS TURNED OFF! -------
!!

The Micromodem 100 may be installed in any S-100 slot. Before inserting the board in the computer you should set up the various options that you may need for your application. See section 4.2 for details. Once you have set the options and plugged the Micromodem into your computer you can connect the ribbon cable between the Microcoupler and the Micromodem 100. This 10 conductor cable is terminated on each end with identical female connectors. Pin 2 is blocked with a plastic plug. The male plugs on the Micromodem 100 and Microcoupler have pin 2 cut off. This missing pin and plugged hole arrangement form a key which makes it difficult to plug in backwards.



Both ends of the ribbon cable are the same, and are interchangeable. Pick one end and plug it into the Micromodem 100 board. If you have any difficulty plugging it in, try turning it over -- it might be backward. When the connector is properly mated, all of the pins will fit easily into their corresponding holes, and the connector body will be flush with the printed circuit board.

Run the ribbon cable out through one of the cable slots in the back wall of the case. You may now replace the cover.

The other end of the ribbon cable connects to the Microcoupler. On one end of the Microcoupler you will see two connectors. In the center of the end wall is a rectangular hole about an inch long. Just peeking through this hole you will see another set of pins just like the ones on the Micromodem 100 board. They are also keyed in the same fashion.

Plug the remaining end of the ribbon cable into this connector. Again, if you have any difficulty, check that it is turned the right way.

You are now ready to connect your computer to the national telephone network.

4.12 CONNECTING TO THE TELEPHONE NETWORK

There are a few details that you need to take care of now before you can legally connect your computer to the telephones. Keep in mind that the Microcoupler is the only part which is FCC registered. The Micromodem 100 and the computer are of no concern to the FCC or the telephone company.

Legal and Technical Details

First you CANNOT legally connect your Microcoupler to either a party line or a pay phone.

Next you need a place to plug it in. If your home or office is already equipped with modular telephone jacks, then you may already have this. You can recognize a modular telephone jack by the small squarish hole (about 3/8") where the modular plug goes in. If you have jacks with four round holes about 3/4" inch apart, then you don't have modular jacks, but an older type of connector. Keep in mind that even if you do have what appears to be a modular jack, it may not be compatable with the Microcoupler. The Microcoupler requires a USOC-RJ11W or USOC-RJ11C modular jack.

4.13 Notifying the telephone company

Before you plug anything into the telephone line you must notify the telephone company. One reason this is necessary that they normally test all their lines on a regular basis. To do this they need to know what's attached to each line, and how many ringers there are on the line. The test measures how much current is drawn on the line. If it draws too much or too little, then they will know that there is something broken or that someone has been tampering with the line. This is how they find illegal extension phones.

You must call the telephone company business office (their telephone number is probably printed on your phone bill) and tell them that you are preparing to install an FCC registered device and wish to notify them. They will need to know what line you are connecting it to (the phone number), the FCC registration number (BI986H-62226-PC-E), the ringer equivalence number (0.4B) and that it needs to be connected to a modular jack USOC-RJ11W or USOC-RJ11C. If you plan to move your computer around, you are allowed to give the phone company a list of phone numbers. Then you won't have to notify them again as long as you move it to one of the lines on your list.

If your phone company has any additional questions about the Microcoupler, ask them to call us in Atlanta.

You are also required to notify the phone company whenever you permanently remove the Microcoupler.

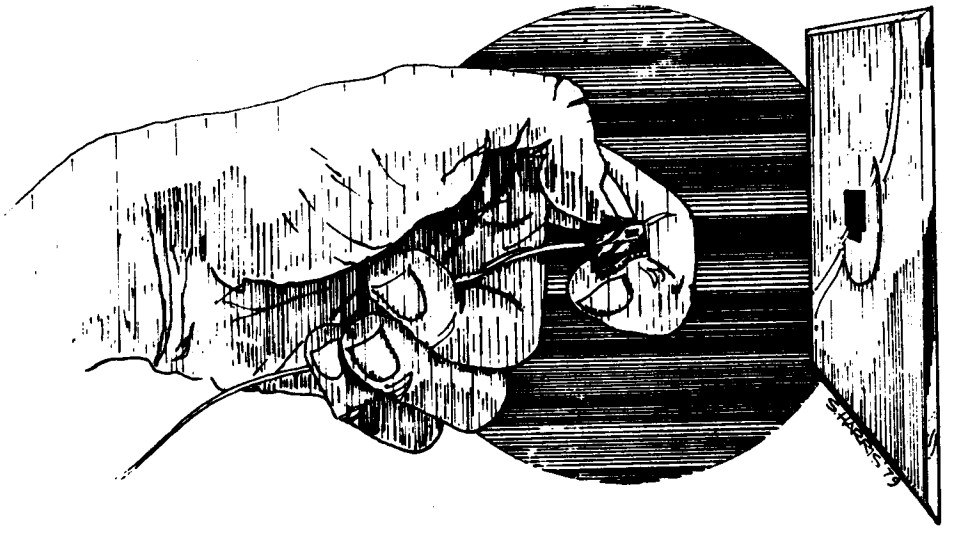
If you experience trouble with your telephone line, you must first disconnect the Microcoupler and make sure that it is not causing the problem. The phone company cannot be responsible for problems caused by your equipment connected to their lines.

In the unlikely event that your Microcoupler causes trouble with your telephone line, you must disconnect it and not use it until it has been repaired. In order to keep FCC approval in force (and thereby be legal), all repairs must be performed by D.C. Hayes Associates, Inc. or our authorized agents. We can repair any problem with your Microcoupler or your Micromodem 100 very promptly. Please see your warranty for full details.

One more legal technicality and you can plug it in. We are obliged to inform you that the telephone company has the right to change their equipment and is under no obligation to make sure that their new equipment is compatible with your Microcoupler. It is very likely that they will change their equipment, but it is extremely unlikely that those changes will render your Micromodem 100 or your Microcoupler unusable. The interface between the Microcoupler and the telephone line is functionally identical to the interface between a standard dial phone and the telephone line. Since the phone companies have millions of dial phones operating today, they are very unlikely to install any new equipment that will render them all obsolete. It would cost too much to replace all those telephones.

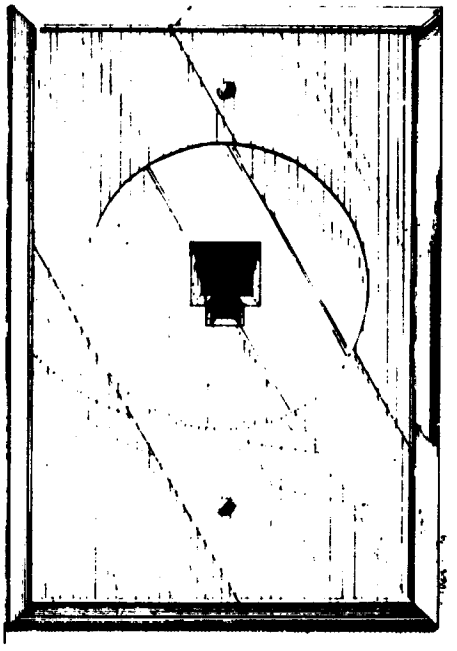
Plugging in to the Telephone Line

Pick up the modular telephone cable. You will see that there is a small plug on each end. The plug is molded of clear plastic and has 4 gold contacts on one side and a small plastic tab on the other. The plastic tab forms a latch which will hold the plug in the jack. Insert one of the plugs into the modular jack on the Microcoupler. It will only go in one way, and when you put it all the way in, it will make a small snap as the latch takes hold. Pull on the cord. It will not unplug unless you press on the plastic tab.



Now you can plug the other end of the modular cable into the telephone jack on the wall.

This completes the installation of your Micromodem 100.



4.2 USER OPTIONS

All Micromodem 100 boards are shipped from our factory with the following options installed.

PORT ADDRESS	=	80 Hex	see section 4.3
WAIT STATES	=	NONE	see section 4.4
INTERRUPTS	=	NONE	see section 4.7
AUTODISCONNECT	=	OFF	see section 4.75
CARRIER DET. LEVEL	=	-50 DBM	see section 4.6
BAUD RATES	=	300/110	see section 4.5

Most users will find these options well suited for their application. However, if you want to change any options please read the following text.

4.3 SELECTING AN I/O PORT ADDRESS

The Micromodem 100 occupies 4 consecutive I/O port addresses on the S-100 bus. The first address (base) is set by the first 6 switches in the 8 pole dip switch (S1) on the right side of the board. Switches 1-6 correspond to address bits A7-A2. The desired port address is selected by turning these switches on or off to match the address lines. An ON switch will match a zero and an OFF switch will match a one.

Examples:

Address bits	A7	A6	A5	A4	A3	A2	
Switch #	1	2	3	4	5	6	
port 80 Hex	off	on	on	on	on	on	(standard)
port 00	on	on	on	on	on	on	
port 04	on	on	on	on	on	off	
port 0C	on	on	on	on	off	off	

The Micromodem 100 is shipped set to port 80 Hex. This is the standard address used by all D.C. Hayes Associates, Inc. software. If this conflicts with another board in your system you will need to set the Micromodem 100 to an unused port address and change the port equates in the software.

4.4 WAIT STATES

The Micromodem 100 will work in most systems at full speed with no wait states. However, some 4 MHZ Z80 or 5 MHZ 8085 based systems may provide a PWR (processor write) pulse which is less than the 300 nanoseconds required by the Micromodem 100. If you install the Micromodem 100 in such a system you should install the wait state jumper. This will insert one CPU wait state during outputs and extend the PWR pulse by one clock period (250 NS at 4 MHZ). No wait states are required for input operations.

Installing a jumper between the pads marked "P3" will activate the wait state circuit.

4.5 BAUD RATES

The baud rate of the Micromodem 100 can be switched under software control between 300 baud and a lower user defined baud rate. Units are shipped with the low baud rate set to 110. Any baud rate from 30.6 to 300 may be selected with an error of less than 2% by installing from 1 to 8 diodes in the baud rate select area. If frequent baud rate changes are required we suggest that you put a 16 pin IC socket in the diode area and wire up several 16 pin headers with diodes for the baud rates you will need.

4.51 BAUD RATE CALCULATION

The 8 diode positions in the baud rate select area form an 8 bit binary number which sets the divide ratio on the baud rate counter. The presence of a diode equals a logic 1 while the absence of a diode equals a logic 0.

Baud rate formulas:

N=Divider ratio B=Baud rate

N=7812.5/B or B=7812.5/N

110 baud example:

N=7812.5/110 = 71.0227

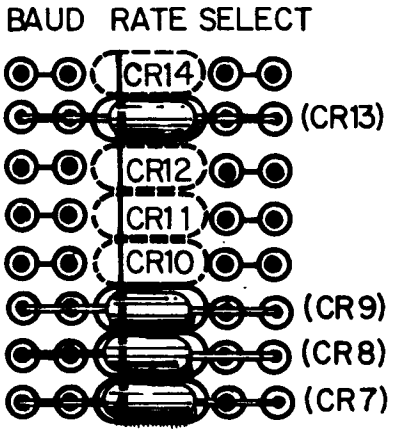
rounded off N=71 decimal or 47 hex or 01000111 binary.

Since one diode is required for each binary 1, four diodes are needed. The actual baud rate obtained will be:

B=7812.5/71 = 110.0352

This is only .032% high.

The diodes are installed as shown in the diagram below. They are 1N4148 or equivilent silicon switching diodes. Note that no baud rate greater than 300 may be selected.



4.52 VALUE OF N FOR COMMON BAUD RATES

Baud rate	N decimal	N binary
45.5	172	10101100
50	156	10011100
75	104	01101000
110	71	01000111
134.5	58	00111010
150	52	00110100
200	39	00100111

BIT TO DIODE LOCATION TABLE

DIODE	BIT	BINARY WEIGHT
CR7	0	1
CR8	1	2
CR9	2	4
CR10	3	8
CR11	4	16
CR12	5	32
CR13	6	64
CR14	7	128

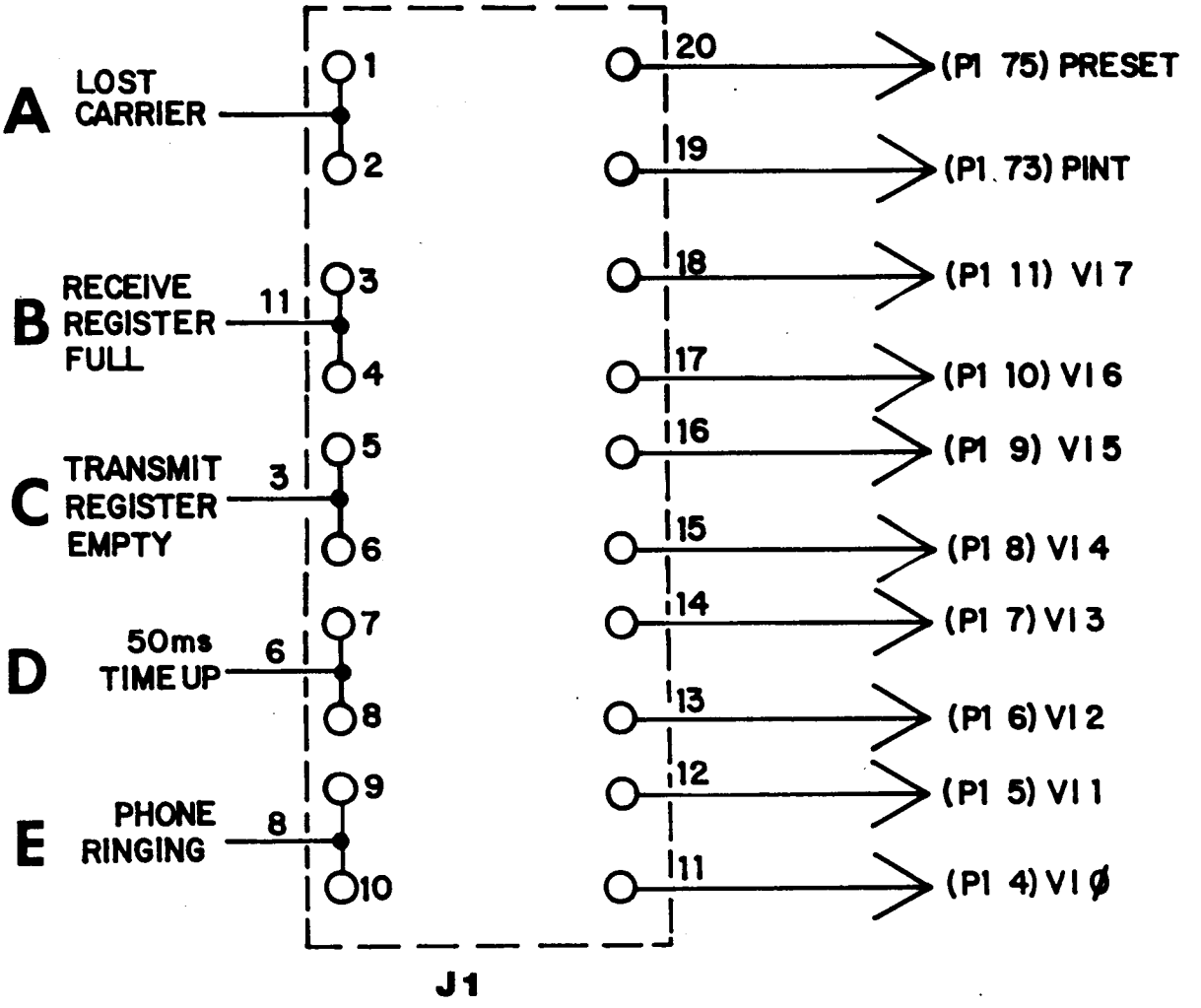
4.6 CARRIER DETECT SENSITIVITY

The carrier detection circuits in the Micromodem 100 will respond to an incoming data carrier as weak as -50 DBM. In most cases this is desirable but sometimes a very noisy line can cause a false indication of carrier. If switch number 7 on dip-switch S1 is turned OFF the carrier detect sensitivity will be reduced to -40 DBM. Since the majority of all data signals are louder than -40 DBM this will have little effect on performance.

If you have problems detecting carrier on weak signals check this switch. Make sure it is in the -50 DBM position (ON).

4.7 INTERRUPTS

The Micromodem 100 provides 5 signals which can be used to generate interrupts. All 5 of these signals and all the S-100 bus interrupt lines are brought out to a 20 pin DIP connector (J1) as illustrated below. All signals are low true and open collector.



Processor reset is also on J1 at pin 20. You may connect the LOST CARRIER or PHONE RINGING signals to this pin if you want the computer to be reset and/or re-boot on one of these conditions. This feature is useful in unattended systems which must recover automatically after a crash.

The following describes the 5 interrupt signals in detail. All signals are active low open collector and may be jumpered to VI0-VI7 or PINT. All except LOST CARRIER are individually maskable. See section 5 on control registers for details.

4.71 PHONE RINGING (E)

This signal goes low for the duration of the 20 HZ telephone ringing signal. Since this signal returns high between rings the program can count them and answer after any number have occurred. It may also be connected to the processor reset line to cause the system to re-boot when the phone rings.

NOTE: This interrupt is enabled only when the Micromodem 100 is in ANSWER mode.

4.72 50 MS TIME-UP (D)

This signal goes low 50 milliseconds after the timer was started. It will remain low until the timer is re-started or the timer interrupt enable bit (D5) in control register 1 is cleared.

4.73 RECEIVE REGISTER FULL (B)

This signal goes low when a character has been received in the data register. It is cleared by reading the data register or by clearing the receive interrupt enable bit in control register 2. (RIE, bit 6)

4.74 TRANSMIT REGISTER EMPTY (C)

This signal goes low when the transmit holding register is empty and ready for another character. It is cleared by writing a byte to the data register or by clearing the transmit interrupt enable bit in control register 2. (TIE, bit 5)

4.75 LOST CARRIER and AUTO-DISCONNECT (A)

This signal goes low 10 seconds after the receive carrier is lost only if the modem is OFFHOOK. If the receive carrier is lost but returns within 10 seconds this circuit will NOT be activated and the 10 second timer will be reset to zero. Note that the time spent dialing and establishing a connection with no

carrier present does not activate this signal because it looks for the presence of a carrier before it arms itself to detect the loss of a carrier.

This signal is provided for emergency re-boot in case the remote user crashes the system then hangs up. It is most useful in single user systems when strapped to processor reset. In normal operation the program should poll the carrier detect bit in the status register and go on hook (hang up) as soon as carrier is lost. If the program can't do this because of a system crash the lost carrier signal will reset the CPU 10 seconds later.

If you only want to reset the Micromodem 100 and hang up without resetting the CPU, the jumper option P4 may be installed.

After connecting LOST CARRIER to an interrupt or reset line you must turn switch 8 on DIP switch S1 to the OFF position to enable this option. This switch is marked ADE (AUTO DISCONNECT ENABLE).

5.1 PROGRAMMABLE REGISTERS

The Micromodem 100 has two read-only registers and four write-only registers. The I/O port address of these registers is determined by switches 1 thru 6 on DIP switch S1. See section 4.3 for details on selecting a base address. The standard base I/O port address is 80 Hex.

PORT		READ-ONLY REGISTERS	WRITE-ONLY REGISTERS
base+0	(80)	receive data	transmit data
base+1	(81)	status	control reg. 1
base+2	(82)	not used	control reg. 2
base+3	(83)	not used	control reg. 3

5.2 TRANSMIT DATA REGISTER

ADDRESS = base+0 (write only)

Data written into this register is converted to serial FSK data and transmitted over the phone line. When 5, 6, or 7 bit characters are used the remaining most significant bits in the byte are not transmitted and their value has no effect on the transmitted data. Data should be written into this register only when the TRANSMITTER REGISTER EMPTY (TRE) bit in the status register is a one (1). This bit should be tested before each output to the transmit data register.

The transmitter is double buffered. It actually has a holding register, which the program writes characters to, and a transmit register which does the parallel to serial conversion. When a character is written into the holding register the transmit register is automatically tested to see if it is empty. If it is found to be empty, the character in the holding register is loaded into transmit register and converted to serial data. When this happens the TRE bit in the status register is set to a 1 to indicate that the holding register is ready for another character. If the transmit register had been busy sending the previous character the TRE status bit would have remained low until the holding register could transfer its contents to the transmit register. All of this may seem complex but it is all handled automatically by the UART chip so the programmer doesn't have to worry about it.

5.3 RECEIVE DATA REGISTER

ADDRESS = base+0 (read-only)

This register will contain the received data from the phone line. It should be read only when the RECEIVE REGISTER FULL (RRF) bit in the status register is a one (1). Reading this register resets the RRF bit.

5.4 STATUS REGISTER

ADDRESS = base+1 (read-only)

bit	7	6	5	4	3	2	1	0
function	-RI	CD	TMR	OE	FE	PE	TRE	RRF

BIT 0 = RRF (RECEIVE REGISTER FULL)

RRF = 0 no data
RRF = 1 data byte received in data register

This bit indicates that a character has been received in the DATA REGISTER. Reading the data register will reset this bit.

BIT 1 = TRE (TRANSMIT REGISTER EMPTY)

TRE = 0 transmitter busy
TRE = 1 transmitter ready for next character

This bit indicates the status of the transmit register. When it is a one (1) the transmit register is ready to accept a character.

BIT 2 = PE (PARITY ERROR)

PE = 0 no parity error
PE = 1 parity error on last character received

This bit, when set, indicates that a parity error was detected in the current character in the receive data register. This bit will remain set until an error free character is received. If the "no parity" mode is selected this bit will always be zero.

BIT 3 = FE (FRAMING ERROR)

FE = 0 no framing error
FE = 1 framing error on last character received

This bit, when set, indicates that the UART receiver failed to detect a stop bit at the proper time and the contents of the receive register are probably invalid. This bit is reset when a character is properly received.

BIT 4 = OE (OVERRUN ERROR)

OE = 0 no overrun error
OE = 1 lost 1 or more characters

This bit, when set, indicates that the data register was not read before the next character was received. This can only be caused by software bugs.

BIT 5 = TMR (TIMER STATUS)

TMR = 0 50 MS period in progress
TMR = 1 50 MS or more have elapsed

This bit indicates the status of the 50 MS timer. It goes to zero when the timer is started and stays zero for 50 milliseconds.

BIT 6 = CD (CARRIER DETECT)

CD = 0 no data carrier present
CD = 1 data carrier present

This bit indicates the status of the received carrier. This bit should be checked frequently during communications to make sure a connection still exist. This bit affects the LOST CARRIER circuits. See section 4.75 for details of automatic disconnect on loss of carrier.

BIT 7 = RI (RING INDICATOR [low true])

RI = 0 the phone is ringing
RI = 1 the phone is NOT ringing

When this bit is zero the phone line is ringing. It goes to a one (1) between rings so they may be counted.

5.5 CONTROL REGISTER 1

ADDRESS = base+1 (write-only)

bit	7	6	5	4	3	2	1	0
function	X	X	TMIE	PI	SBS	LS2	LS1	EPE

BIT 0 = EPE (EVEN PARITY ENABLE)

EPE = 0 odd parity
EPE = 1 even parity

This bit selects the type of parity to be generated or checked.

BIT 1 and BIT 2 = LS1 and LS2 (WORD LENGTH SELECT)

LS2	LS1	
0	0	5 data bits
0	1	6 data bits
1	0	7 data bits
1	1	8 data bits

These two bits set the length of the data word to be transmitted and received.

BIT 3 = SBS (STOP BITS SELECT)

SBS = 0 1 stop bit
SBS = 1 2 stop bits for 6, 7, or 8 data bits
1.5 stop bits for 5 data bits

This bit sets the number of stop bits to be transmitted and received.

BIT 4 = PI (PARITY INHIBIT)

PI = 0 parity is generated and checked
PI = 1 no parity

This bit determines if parity is used during transmitting and receiving. Bit 0 sets the type of parity (even or odd).

BIT 5 = TMIE (TIMER INTERRUPTS ENABLE)

TMIE = 0 interrupts from 50 MS timer disabled
TMIE = 1 enable timer interrupts

When this bit is set to a 1 the timer will generate an interrupt when it times out if the jumper has been installed on J1. See section 4.7 for more information about interrupts.

BITS 6 AND 7 ARE NOT USED

5.6 CONTROL REGISTER 2

ADDRESS = base+2 (write only)

bit	7	6	5	4	3	2	1	0
function	OH	RIE	TIE	ST	BRK	MS/RID	TXE	BRS

BIT 0 = BRS (BAUD RATE SELECT)

BRS = 0 low baud rate (user selectable, 110 std.)
BRS = 1 300 baud

This bit selects 1 of 2 baud rates for transmitting and receiving. See section 4.5 for information on user defined baud rates.

BIT 1 = TXE (TRANSMITER ENABLE)

TXE = 0 modem transmit carrier OFF
TXE = 1 modem transmit carrier ON

This bit turns the modem transmitter on and off. It should always be turned off prior to changing the originate/answer MODE select bit.

BIT 2 = MS/RID (MODE SELECT/RING INTERUPT DISABLE)

MS/RID = 0 answer mode, ring interrupts enabled
MS/RID = 1 originate mode, ring interrupts disabled

This is a dual purpose bit. Its primary function to switch the modem between the answer and originate modes. It also controls interrupts from the ring detector. It must be in ANSWER mode if interrupts from the ring detector are wanted. Of course the ring interrupt jumper on J1 must be installed. See section 4.5 for details. Also note that the ring indicator bit in status register is NOT affected by the state of this bit.

The TXE bit must be ZERO before changing the state of the MS/RID bit. The MC14412 modem chip will sometimes lock up in a strange mode if this is not done.

BIT 3 = BRK (BREAK)

BRK = 0 normal operation
BRK = 1 exchange mark and space frequencys
this effectivly sends a break (constant space)

This bit is normally used to send a break signal (required by some time-sharing services). A BREAK is a steady SPACE tone for at least 3 character times. To send a break, the program should wait 1 character time after TRE goes true to allow the last character to be sent by the UART, then set the BREAK bit for 3 character times. This bit should then be reset to return to normal operation.

You can also detect reception of a BREAK. There is no bit in the status register that will give a direct indication but it can be done indirectly. A BREAK will always cause a FRAMING ERROR and the data register will be zero. The program can check for these conditions and assume a BREAK has been received.

BIT 4 = ST (SELF-TEST)

ST = 0 normal mode
ST = 1 self-test mode

When this bit is set to a 1 the modem receiver switches modes so that it "listens" to the transmitter. Data sent by the transmitter can then be received by the receiver for local testing. The Microcoupler should be disconnected when using the self-test mode.

BIT 5 = TIE (TRANSMIT INTERRUPTS ENABLE)

TIE = 0 transmit interrupts disabled
TIE = 1 transmit interrupts enabled

This bit controls interrupts from the transmitter. When it is a one an interrupt will be sent to the CPU each time the transmit data register is empty. The interrupt jumper at J1 must be installed. See section 4.7 for details.

BIT 6 = RIE (RECEIVE INTERRUPTS ENABLE)

RIE = 0 receiver interrupts disabled
RIE = 1 receiver interrupts enabled

This bit controls interrupts from the receiver. When it is a 1 an interrupt will be sent to the CPU each time a character is received in the data register. The jumper on J1 must be installed. See section 4.7 for details.

BIT 7 = OH (OFF-HOOK)

OH = 0 on-hook (idle)
OH = 1 off-hook (pick up phone)

This bit controls the telephone switch-hook in the Microcoupler. It is set to a one to answer incoming calls and with appropriate software can perform the dialing function.

5.7 CONTROL REGISTER 3

ADDRESS = base+3 (write-only)

This register is used to start the 50 MS timer. It has no data bits. Simply writing anything to this register will start the timer. When started, bit 5 in the status register will go to a zero for 50 MS. Timer can't be triggered again until the end of the 50 MS period.

American Standard Code for Information Interchange

CODE	HEX	DEC	CODE	HEX	DEC	CODE	HEX	DEC	CODE	HEX	DEC
NUL	00	0	SP	20	32	@	40	64	`	60	96
SOH	01	1	!	21	33	A	41	65	a	61	97
STX	02	2	"	22	34	B	42	66	b	62	98
ETX	03	3	#	23	35	C	43	67	c	63	99
EOT	04	4	$	24	36	D	44	68	d	64	100
ENQ	05	5	%	25	37	E	45	69	e	65	101
ACK	06	6	&	26	38	F	46	70	f	66	102
BEL	07	7	'	27	39	G	47	71	g	67	103
BS	08	8	(	28	40	H	48	72	h	68	104
HT	09	9	)	29	41	I	49	73	i	69	105
LF	0A	10	*	2A	42	J	4A	74	j	6A	106
VT	0B	11	+	2B	43	K	4B	75	k	6B	107
FF	0C	12	,	2C	44	L	4C	76	l	6C	108
CR	0D	13	-	2D	45	M	4D	77	m	6D	109
SO	0E	14	.	2E	46	N	4E	78	n	6E	110
SI	0F	15	/	2F	47	O	4F	79	o	6F	111
DLE	10	16	0	30	48	P	50	80	p	70	112
DC1	11	17	1	31	49	Q	51	81	q	71	113
DC2	12	18	2	32	50	R	52	82	r	72	114
DC3	13	19	3	33	51	S	53	83	s	73	115
DC4	14	20	4	34	52	T	54	84	t	74	116
NAK	15	21	5	35	53	U	55	85	u	75	117
SYN	16	22	6	36	54	V	56	86	v	76	118
ETB	17	23	7	37	55	W	57	87	w	77	119
CAN	18	24	8	38	56	X	58	88	x	78	120
EM	19	25	9	39	57	Y	59	89	y	79	121
SUB	1A	26	:	3A	58	Z	5A	90	z	7A	122
ESC	1B	27	;	3B	59	[	5B	91	{	7B	123
FS	1C	28	<	3C	60	\	5C	92	|	7C	124
GS	1D	29	=	3D	61	]	5D	93	}	7D	125
RS	1E	30	>	3E	62	^	5E	94	~	7E	126
US	1F	31	?	3F	63	_	5F	95	DEL	7F	127

NUL Null, or all zeros
SOH Start of Heading
STX Start of Text
ETX End of Text
EOT End of Transmission
ENQ Enquiry
ACK Acknowledge
BEL Bell, or Alarm
BS Backspace
HT Horizontal Tab
LF Line Feed
VT Vertical Tab
FF Form Feed
CR Carriage Return
SO Shift Out
SI Shift In
DLE Data Link Escape

DC1 Device Control 1
DC2 Device Control 2
DC3 Device Control 3
DC4 Device Control 4
NAK Negative Acknowledge
SYN Sync
ETB End Transmission Block
CAN Cancel
EM End of Medium
SUB Substitute
ESC Escape
FS File Separator
GS Group Separator
RS Record Separator
US Unit Separator
SP Space
DEL Delete

INPUT REGISTERS

RECEIVER REGISTER
ADDRESS = BASE+0

7	6	5	4	3	2	1	0
DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA

STATUS REGISTER
ADDRESS = BASE+1

7	6	5	4	3	2	1	0
-RI	CD	TMR	OE	FE	PE	TRE	RRF

RRF	RECEIVER REGISTER FULL	1= CHARACTER IN REG.
TRE	TRANSMITTER REG. EMPTY	1= REGISTER EMPTY
PE	PARITY ERROR	1= PARITY ERROR
FE	FRAMMING ERROR	1= FRAMMING ERROR
OE	OVERFLOW ERROR	1= OVERFLOW ERROR
TMR	TIMER	1= 50 MS TIME UP
CD	CARRIER DETECT	1= CARRIER DETECTED
-RI	NOT RING INDICATOR	0= PHONE RINGING

OUTPUT REGISTERS

TRANSMIT REGISTER
ADDRESS = BASE+0

7	6	5	4	3	2	1	0
DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA

CONTROL REGISTER #1
ADDRESS = BASE+1

7	6	5	4	3	2	1	0
X	X	TMIE	PI	SBS	LS2	LS1	EPE

EPE	EVEN PARITY ENABLE	1= EVEN PARITY

LS2	LS1	WORD LENGTH SELECT
0	0	= 5 BITS
0	1	= 6 BITS
1	0	= 7 BITS
1	1	= 8 BITS

SBS	STOP BIT SELECT	0= 1 STOP BIT
PI	PARITY INHIBIT	1= NO PARITY
TMIE	TIMER INTERRUPT ENABLE	1= ENABLE INTERRUPT

CONTROL REGISTER #2
ADDRESS = BASE+2

7	6	5	4	3	2	1	0
OH	RIE	TIE	ST	BRK	MS/RID	TXE	BRS

BRS	BAUD RATE SELECT	1= 300 BAUD 0= 110
TXE	TRANSMITTER ENABLE	1= CARRIER ON
MS/RID	* MODE SELECT/RING INTERRUPT DIS.	1= ORIGINATE 0= ANSWER
BRK	BREAK	1= EXCHANGE MARK & SPACE
ST	SELF TEST	1= SELF TEST MODE
TIE	TRANSMIT INTERRUPTS ENABLE	1= ENABLE
RIE	RECEIVE INTERRUPTS ENABLE	1= ENABLE
OH	OFF HOOK	1= PICK UP PHONE

CONTROL REGISTER #3
ADDRESS = BASE+3

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

ANY OUTPUT TO CONTROL REGISTER #3 WILL START THE 50 MS TIMER.

* See section 5.6 for details on MS/RID.

TERMINAL PROGRAM

The terminal program is written in 8080 assembly language. It will run on any 8080, 8085, or Z80 machine that uses the CP/M disk operating system. It can be modified for other operating systems by changing those portions of the program which are CP/M dependent. This requires a good understanding of 8080 assembly language and the system it will be run with.

The terminal program is intended to be an example of programming techniques for the Micromodem 100 and not the "last word" in data communication programs. It is split into four sections; Terminal control, Disk I/O, Console I/O, and Modem I/O. You are invited to use the modem I/O section to build your own programs around. The console and disk I/O sections are CP/M dependent and should be changed if you intend to use it with a NON-CP/M system. The terminal control section decodes and executes all user commands. It can be expanded by adding more commands to the table "TABLE".

As is, the terminal program will allow its user to call time sharing computers and transfer data or programs to or from disk. It can communicate with a copy of itself in another S-100 machine to send data back and forth. Also, it has been used to transfer messages in and out of computerized bulletin board systems (CBBS).

LIST OF COMMANDS

Command	description
A	Wait for phone to ring, then answer call
AA	Answer call immediately
AO	Answer call immediately but in ORIGINATE mode
B1	Set baud rate to 110
B3	Set baud rate to 300
C0	Clear capture buffer, disable capture
C1	Start capturing received data in buffer
C2	Stop capturing (C1 will start again at this point)
D5551212	Dial phone number 555-1212, then wait for carrier
D	Redial last number dialed
F	Set for FULL DUPLEX (no local echo)

G	Goodbye, hang up phone
H	Set for HALF DUPLEX (local echo)
J4F13	Jump to address, in this case 4F13. This is useful for getting out of the terminal program and into a monitor or debugger. The terminal program can be re-entered at HEX 100.
L5	Set data word length to 5 bits
L6	6 bits
L7	7 bits
L8	8 bits
PE	Set for EVEN parity
PO	ODD parity
PN	NO parity
R TEST.MSG	Read CP/M file "TEST.MSG" from disk and transmit it.
S1	Set STOP BITS = 1
S2	STOP BITS = 2
T	Type capture buffer to console
T5A	Type capture buffer to console starting at the HEX 5A'th character. NOTE: S starts and stops the listing Carriage return aborts this function.
W TEST.MSG	Write the contents of the capture buffer to disk in a file named "TEST.MSG".
X	Exit to CP/M (does warm boot)

?1 Debug mode ON. All incomming data will be displayed in HEX with 16 bytes per line. Please note that your console device must be at least 4 times as fast as the modem baud rate

?0 Debug mode OFF.

Multiple commands may be typed on a line if they are seperated by commas or spaces.

Example: COMMAND:L7,PE,S1,F,D3944220 <CR>

In this example the word length is set to 7 bits, even parity is selected, 1 stop bit, full duplex, then the phone number 394-4220 is dialed. The line is ended with a carriage return. Once the options are set they remain in force until changed.

Some telephone exchanges require you to dial a digit then wait for a second dial tone before dialing the remaining digits. This can be done by putting a "*" in the phone number where pauses are required. The dialing program will wait 2 seconds when it encounters the "*".

example: COMMAND: D 9*3944220 <CR>

When a call is in progress you can return to the command mode without interrupting the call by typing a control A. The program will prompt with "COMMAND:". You can then enter a line of commands and hit carriage return. The commands will be executed and the program will return to communication mode. The control A and the command line will NOT be transmitted to the line. Also, all data from the line will not be received during this time.

Some time sharing systems require a BREAK to stop print-out. Type control W to send a break.

If control A or W conflict with control characters you need to transmit you can change the program to accept any control codes you want. Just change the two CPI statments in the "CSEND" subroutine and re-assemble the program.

You can exit to CP/M during a call without hanging up. Type control A then the command "X". It will then ask "do you want to hang up first". Type N and CP/M will re-boot and the connection will still exist. If you don't load any other programs which overwrite the Terminal Program RAM area at 1000 HEX you can re-enter by loading and executing the Terminal Program .COM file as if it were a cold start. When the program starts running at 100 hex it will check for a receive carrier. If none is present it will go through the cold start initialization code. If it detects a carrier it will compute the checksum of the scratch pad ram area. If the ram has not been changed it will go directly to the communication loop and type "CONNECTION ESTABLISHED". If the ram has been altered it will print "REENTRY IMPOSSIBLE", reset the modem, and cold start.

This program is avalable on a CP/M format 8 inch single density floppy disk from D.C. Hayes Associates, Inc. for $25.00. Georgia residents please add 4% sales tax.

```
                        TITLE  'MICROMODEM 100 TERMINAL PROGRAM'


                ;       VER. 1.1        5-18-79

                ;WRITTEN BY DALE A. HEATHERINGTON

0100                    ORG     100H

0001 =          CPM     EQU     1       ;SET FOR CPM ASSEMBLY

0005 =          BDOS    EQU     5       ;CPM ENTRY
0000 =          REBOOT  EQU     0       ;CPM WARM BOOT


000D =          CR      EQU     0DH     ;CARRIAGE RETURN
000A =          LF      EQU     0AH     ;LINE FEED
0007 =          BELL    EQU     7       ;ASCII BELL



0100 317B10     START:  LXI     SP,STACK
0103 CDC109             CALL    INIT            ;SET UP CP/M I/O VECTORS
0106 11B301             LXI     D,PROMPT        ;PUSH A RETURN ADDRESS ON STACK
0109 D5                 PUSH    D               ;IN CASE CARRIER IS PRESENT
010A CDFC09             CALL    CDSTAT          ;SEE IF WE GOT CARRIER
010D B7                 ORA     A
010E CA4501             JZ      STR1            ;NO, DO INITIALIZATION
0111 CDCD01             CALL    CKRAM           ;GET CHECKSUM OF RAM AREA
0114 4F                 MOV     C,A
0115 3A3A10             LDA     CKSUM           ;GET OLD CHECKSUM
0118 B9                 CMP     C               ;SAME?
0119 CA9604             JZ      TALK            ;RAM OK, JUMP TO COMM LOOP
011C CDE402             CALL    PRINT
011F 5245454E54         DB      'REENTRY NOT POSSIBLE, RAM OVERWRITTEN',CR
0145 CDD609     STR1:   CALL    RESET           ;ELSE CLEAR MODEM
0148 3A0700             LDA     7               ;GET PAGE ADDRESS OF CP/M
014B D601               SUI     1               ;1 PAGE BELOW IT
014D 323510             STA     MTOP            ;TOP OF USER MEMORY
0150 CDDF02             CALL    CRLF
0153 CDE402             CALL    PRINT
0156 2020202020         DB      '       D.C. Hayes Associates, Inc.',CR
0178 CDE402             CALL    PRINT
017B 4D6963726F         DB      'Micromodem 100 Terminal program  ver. 1.1',CR
01A5 AF                 XRA     A               ;A=0
01A6 321010             STA     NMBR            ;SET PHONE NUMBER BUFFER EMPTY
01A9 21B901             LXI     H,DFAULT        ;POINT TO COMMAND STRING
01AC 223010             SHLD    BUFPNT          ;OF DEFAULT OPTIONS
01AF CDF501             CALL    DEC1            ;EXECUTE THE COMMANDS
01B2 D1                 POP     D               ;CLEAN UP STACK

01B3 CDDB01     PROMPT: CALL    CMND            ;GET AND EXECUTE COMMANDS
01B6 C3B301             JMP     PROMPT          ;LOOP FOREVER
```

```
                ;THIS IS A COMMAND LINE THAT SETS THE DEFAULT OPTIONS.
                ;8 DATA BITS, 1 STOP BIT, NO PARITY, 300 BAUD, FULL DUPLEX,
                ;CLEAR CAPTURE BUFFER. SET DEBUG MODE OFF.

01B9 4C382C5331DFAULT: DB       'L8,S1,PN,B3,F,C0,?0',CR

                ;DOES A CHECKSUM OF THE SCRATCH PAD RAM LOCATIONS

01CD 063A       CKRAM:  MVI     B,EOR-SOR       ;END OF RAM MINUS START OF RAM = COUNT
01CF 210010             LXI     H,SOR           ;START OF RAM POINTER
01D2 3E00               MVI     A,0             ;A IS CHECKSUM
01D4 86         CKR1:   ADD     M               ;ADD MEMORY BYTE
01D5 23                 INX     H               ;INC. POINTER
01D6 05                 DCR     B               ;COUNT 'EM
01D7 C2D401             JNZ     CKR1            ;LOOP IF NOT DONE
01DA C9                 RET                     ;CHECKSUM IN REG A



                ;THIS ROUTINE FILLS THE COMMAND BUFFER WITH
                ;A COMMAND STRING FROM THE CONSOLE THEN
                ;DECODES AND EXECUTES THEM.

01DB CDDF02     CMND:   CALL    CRLF
01DE CDE402             CALL    PRINT
01E1 434F4D4D41         DB      'COMMAND:@'
01EA CD2F02             CALL    FILBUF          ;GET COMMANDS TO BUFFER
01ED B7                 ORA     A               ;TEST FOR NO COMMANDS
01EE C8                 RZ                      ;RETURN IF NONE IN BUFFER

01EF 217C10     DECODE: LXI     H,BUF           ;POINT TO COMMAND BUFFER
01F2 223010             SHLD    BUFPNT          ;SAVE POINTER
01F5 CD8302     DEC1:   CALL    NEXT            ;GET A COMMAND
01F8 FE0D               CPI     CR              ;END OF LINE?
01FA C8                 RZ
01FB B7                 ORA     A               ;DELIMITER?
01FC CAF501             JZ      DEC1
01FF 21BB07             LXI     H,TABLE         ;POINT TO DECODE TABLE
0202 4F                 MOV     C,A             ;STORE COMMAND IN C
0203 7E         SRCH:   MOV     A,M             ;GET BYTE FROM TABLE
0204 B9                 CMP     C               ;DOES IT MATCH?
0205 CA2502             JZ      FOUND           ;YES, JUMP
0208 23                 INX     H               ;NO, ADVANCE POINTERS
0209 23                 INX     H
020A 23                 INX     H
020B B7                 ORA     A               ;TEST FOR END OF TABLE
020C C20302             JNZ     SRCH
020F CDB709             CALL    CONOUT          ;ECHO BAD COMMAND
0212 CDE402             CALL    PRINT
0215 203D204241         DB      ' = BAD COMMAND',CR
0224 C9                 RET

0225 11F501     FOUND:  LXI     D,DEC1          ;RETURN ADDRESS
0228 D5                 PUSH    D               ;PUT IT ON STACK
0229 23                 INX     H
```

```
022A 5E             MOV     E,M             ;GET LOW BYTE OF COMMAND ADR.
022B 23             INX     H
022C 56             MOV     D,M             ;GET HIGH BYTE
022D EB             XCHG                    ;ADDRESS TO HL
022E E9             PCHL                    ;CALL ROUTINE


                ;THIS ROUTINE GETS CHARACTERS FROM THE CONSOLE
                ; AND PUTS THEM IN A BUFFER.  RUBOUT WILL DELETE
                ; THE LAST CHARACTER TYPED. CONTROL U WILL KILL
                ; THE WHOLE LINE.  THE BUFFER IS 64 CHARACTERS
                ; LONG.


022F 217C10     FILBUF: LXI     H,BUF
0232 0600               MVI     B,0             ;CHARACTER COUNTER
0234 CDA109     FIL1:   CALL    CONIN           ;GET ONE
0237 FE7F               CPI     7FH             ;IS IT A RUBOUT?
0239 C24B02     FIL2:   JNZ     FIL3            ;JUMP IF NOT
023C 78                 MOV     A,B
023D B7                 ORA     A               ;SEE IF BUFFER EMPTY
023E CA2F02             JZ      FILBUF
0241 2B                 DCX     H
0242 05                 DCR     B               ;BACK UP EVERYTHING
0243 0E5F               MVI     C,5FH           ;BACK ARROW CHARACTER
0245 CDB709             CALL    CONOUT
0248 C33402             JMP     FIL1

024B FE15       FIL3:   CPI     15H             ;CONTROL U?
024D C25A02             JNZ     FIL4
0250 0E23               MVI     C,'#'
0252 CDB709             CALL    CONOUT          ;ECHO #
0255 CDDF02             CALL    CRLF
0258 AF                 XRA     A
0259 C9                 RET                     ;RETURN A=0

025A FE0D       FIL4:   CPI     CR              ;CARRIAGE RETURN
025C C26502             JNZ     FIL5
025F 77                 MOV     M,A             ;PUT IT IN BUFFER
0260 78                 MOV     A,B             ;GET COUNT
0261 CDDF02             CALL    CRLF
0264 C9                 RET

0265 CD7D02     FIL5:   CALL    UCASE           ;CONVERT TO UPPER CASE
0268 77                 MOV     M,A             ;PUT IT IN MEMORY
0269 4F                 MOV     C,A
026A 3E40               MVI     A,64            ;MAX. LENGTH OF BUFFER
026C B8                 CMP     B
026D C27502             JNZ     FIL6            ;JUMP IF NOT FULL
0270 0E23               MVI     C,'#'
0272 C37702             JMP     FIL7
0275 23         FIL6:   INX     H
0276 04                 INR     B               ;ADVANCE POINTER AND COUNTER
0277 CDB709     FIL7:   CALL    CONOUT          ;ECHO CHARACTER
027A C33402             JMP     FIL1
```

```
                ;LOWER TO UPPER CASE CONVERSION ROUTINE.

027D FE60       UCASE:  CPI     60H
027F D8                 RC                      ;RETURN IF ALREADY UPPER CASE
0280 D620               SUI     20H             ;ELSE CONVERT TO UPPER
0282 C9                 RET


                ;THIS ROUTINE GETS THE NEXT CHARACTER FROM THE BUFFER.
                ;IT RETURNS A=0 IF A COMMA OR SPACE IS ENCOUNTERED.

0283 2A3010     NEXT:   LHLD    BUFPNT          ;GET ADDRESS OF BUFFER
0286 7E                 MOV     A,M             ;GET A BYTE
0287 23                 INX     H
0288 223010             SHLD    BUFPNT          ;INCREMENT AND SAVE POINTER
028B FE20               CPI     20H             ;SPACE?
028D CA9D02             JZ      DELM
0290 FE2C               CPI     ','             ;COMMA?
0292 CA9D02             JZ      DELM
0295 FE0D               CPI     CR              ;CARRIAGE RETURN?
0297 C0                 RNZ                     ;RETURN IF NOT
0298 2B                 DCX     H
0299 223010             SHLD    BUFPNT
029C C9                 RET
029D AF         DELM:   XRA     A
029E C9                 RET

                ;THIS ROUTINE ADVANCES THE BUFFER POINTER TO THE
                ;NEXT COMMAND OR DIGIT STRING.
                ;IF NO MORE CHARACTERS WERE FOUND IT WILL RETURN
                ;WITH A=0.  OTHERWISE A=1

029F CD8302     NXTCHR: CALL    NEXT
02A2 0E00               MVI     C,0
02A4 FE0D               CPI     CR              ;TEST FOR END OF BUFFER
02A6 CAB502             JZ      NXT1
02A9 B7                 ORA     A               ;TEST FOR DELIMITER
02AA CA9F02             JZ      NXTCHR          ;JUMP IF SPACE OR COMMA
02AD 0C                 INR     C               ;C = 1
02AE 2A3010             LHLD    BUFPNT          ;GET BUFFER POINTER
02B1 2B                 DCX     H               ;BACK UP ONE
02B2 223010             SHLD    BUFPNT          ;STORE IT
02B5 79         NXT1:   MOV     A,C             ;A=0 NONE FOUND
02B6 C9                 RET                     ;A=1 CHARACTER FOUND


                ;GET A HEX DIGIT STRING FROM BUFFER AND
                ;CONVERT IT TO A 16 BIT BINARY NUMBER IN DE.

02B7 210000     GETHEX: LXI     H,0
02BA EB         GH1:    XCHG                    ;NUMBER TO DE
```

```
02BB CD8302            CALL    NEXT            ;GET A DIGIT
02BE FE30              CPI     '0'             ;SEE IF IT'S HEX
02C0 D8                RC                      ;RETURN ON FIRST NON HEX CHAR.
02C1 FE47              CPI     'G'
02C3 D0                RNC
02C4 FE41              CPI     'A'
02C6 D2CC02            JNC     GH2
02C9 FE3A              CPI     ':'
02CB D0                RNC
02CC D630      GH2:    SUI     30H             ;REMOVE ASCII BIAS
02CE FE0A              CPI     10
02D0 DAD502            JC      GH3             ;JUMP IF DIGIT 0-9
02D3 D607              SUI     7               ;ELSE REMOVE A-F BISAS
02D5 EB        GH3:    XCHG                    ;NUMBER TO HL
02D6 29                DAD     H               ;SHIFT LEFT 4 BITS
02D7 29                DAD     H
02D8 29                DAD     H
02D9 29                DAD     H
02DA 85                ADD     L               ;ADD IN THIS DIGIT
02DB 6F                MOV     L,A
02DC C3BA02            JMP     GH1


               ;SEND CARRIAGE RETURN-LINE FEED TO CONSOLE

02DF CDE402    CRLF:   CALL    PRINT
02E2 0D                DB      CR
02E3 C9                RET


               ;PRINTS A STRING POINTED TO BY CONTENTS OF TOP
               ; OF STACK. CARRIAGE RET. OR "@" TERMINATES THE STRING.
               ; "@" WILL PRINT STRING WITH NO CR-LF SEQUENCE AT END.


02E4 E3        PRINT:  XTHL                    ;TOP OF STACK TO HL
02E5 F5                PUSH    PSW             ;SAVE REGISTERS
02E6 C5                PUSH    B
02E7 7E        PRN1:   MOV     A,M             ;GET A CHARACTER
02E8 23                INX     H
02E9 FE40              CPI     '@'             ;END?
02EB CAFF02            JZ      PRN2            ;JUMP IF END, NO CRLF
02EE 4F                MOV     C,A
02EF CDB709            CALL    CONOUT          ;PRINT THE CHAR.
02F2 FE0D              CPI     CR              ;WAS IT A CARRIAGE RET.?
02F4 C2E702            JNZ     PRN1            ;NO, GO BACK FOR MORE
02F7 CDE402            CALL    PRINT           ;CALL OURSELF
02FA 0A00000040        DB      LF,0,0,0,'@'    ;LINE FEED AND 3 NULLS
02FF C1        PRN2:   POP     B               ;RESTORE REGISTERS
0300 F1                POP     PSW
0301 E3                XTHL                    ;RETURN ADDRESS TO STACK
0302 C9                RET
```

```
                ;RETURNS TO USERS OPERATING SYSTEM


0303 CDDF02     QUIT:   CALL    CRLF
0306 CD330A             CALL    SHSTAT  ;GET SWITCH-HOOK STATUS
0309 B7                 ORA     A
030A CA0000             JZ      REBOOT
030D CDE402             CALL    PRINT
0310 444F20594F         DB      'DO YOU WANT TO HANG UP FIRST? (Y OR N)@'
0337 CDA109             CALL    CONIN
033A CD7D02             CALL    UCASE
033D 4F                 MOV     C,A
033E CDB709             CALL    CONOUT
0341 79                 MOV     A,C
0342 FE4E               CPI     'N'
0344 C25003             JNZ     QUIT1
0347 CDCD01             CALL    CKRAM           ;GET CURRENT RAM CHECKSUM
034A 323A10             STA     CKSUM           ;SAVE IT
034D C30000             JMP     REBOOT          ;BACK TO CPM
0350 CD7505     QUIT1:  CALL    HANGUP
0353 C30000             JMP     REBOOT


                ;THIS ROUTINE DIALS A PHONE NUMBER THEN JUMPS TO
                ;THE COMMUNICATION ROUTINE.
                ;IF NO PHONE NUMBER FOLLOWS THE "D" COMMAND THE
                ;LAST NUMBER DIALED WILL BE RE-DIALED.

0356 CDE402     DIAL:   CALL    PRINT
0359 4449414C49         DB      'DIALING-@'
0362 CDC40A             CALL    DLTONE          ;GET A DIAL TONE
0365 CD9F02             CALL    NXTCHR          ;FIND PHONE NO. IN BUFFER
0368 B7                 ORA     A
0369 CA7F03             JZ      DL1             ;PHONE NUMBER NOT PRESENT
036C 110F10             LXI     D,NMBR-1        ;POINT TO PHONE NO. BUFFER
036F CD8302     DL0:    CALL    NEXT            ;GET A DIGIT
0372 13                 INX     D               ;ADVANCE POINTER
0373 12                 STAX    D               ;PUT IT IN BUFFER
0374 B7                 ORA     A
0375 CA7F03             JZ      DL1
0378 FE0D               CPI     CR
037A C26F03             JNZ     DL0             ;IF NOT END, LOOP BACK
037D AF                 XRA     A               ;A=0
037E 12                 STAX    D               ;ZERO MARKS END OF NUMBER
037F 211010     DL1:    LXI     H,NMBR          ;POINT TO NUMBER BUFFER
0382 7E         DL2:    MOV     A,M             ;GET A DIGIT
0383 23                 INX     H               ;ADVANCE POINTER
0384 B7                 ORA     A               ;TEST FOR END
0385 CA1F04             JZ      COMM            ;GO TRY TO COMMUNICATE
0388 4F                 MOV     C,A
0389 CDB709             CALL    CONOUT          ;ECHO DIGIT WE'RE DIALING
038C 79                 MOV     A,C
038D 0E28               MVI     C,40
038F FE2A               CPI     '*'             ;2 SECOND DELAY?
0391 F5                 PUSH    PSW
0392 CCA50A             CZ      DELAY           ;YES, WAIT 2 SEC.
```

```
0395 F1              POP     PSW
0396 CA8203          JZ      DL2
0399 FE3A            CPI     3AH             ;IGNORE ALL XCEPT DIGITS
039B D28203          JNC     DL2
039E FE30            CPI     '0'
03A0 DA8203          JC      DL2
03A3 4F              MOV     C,A
03A4 CDD10A          CALL    PULSE           ;DIAL THE DIGIT
03A7 C38203          JMP     DL2             ;LOOP BACK FOR MORE


                     ;THIS ROUTINE WAITS FOR THE PHONE TO RING.
                     ;IT THEN ANSWERS ON THE FIRST RING, TURNS ON
                     ;THE CARRIER, THEN JUMPS TO "COMM1" AND WAITS
                     ;FOR A RECEIVE CARRIER.

03AA CD8302    ANSWR: CALL   NEXT            ;GET NEXT CHARACTER
03AD FE41            CPI     'A'             ;A=ANSWER WITHOUT A RING
03AF CAF603          JZ      ANS2            ;IN ANSWER MODE
03B2 FE4F            CPI     'O'             ;O=ANSWER WITHOUT A RING
03B4 CA1F04          JZ      COMM            ;IN ORIGINATE MODE
03B7 CDE402          CALL    PRINT           ;ELSE WAIT FOR RING
03BA 5741495449      DB      'WAITING FOR RING. PRESS ANY KEY TO ABORT.',CR
03E4 CD050A    ANS1: CALL    RISTAT          ;CHECK FOR PHONE RINGING
03E7 B7              ORA     A
03E8 C2F603          JNZ     ANS2            ;IF YES THEN ANSWER IT
03EB CDAD09          CALL    CONST           ;ABORT?
03EE B7              ORA     A
03EF CAE403          JZ      ANS1            ;NO, LOOP BACK
03F2 CDA109          CALL    CONIN           ;GET CHARACTER
03F5 C9              RET                     ;ABORT
03F6 CDE402    ANS2: CALL    PRINT
03F9 07414E5357      DB      BELL,'ANSWERING CALL',CR
0409 0E01            MVI     C,1
040B CD290A          CALL    SWHOOK          ;ANSWER PHONE
040E 3E00            MVI     A,0             ;A=0 FOR ANSWER MODE
0410 320310          STA     MFLAG
0413 4F              MOV     C,A
0414 CD3D0A          CALL    MODE            ;SET TO ANSWER MODE
0417 0E01            MVI     C,1
0419 CD150A          CALL    TXON            ;TURN ON TRANSMIT CARRIER
041C C32B04          JMP     COMM1


                     ;THIS ROUTINE WAITS FOR A CARRIER THEN GOES INTO
                     ;THE TELETYPE MODE FOR COMMUNICATIONS.

041F CDDF02    COMM: CALL    CRLF
0422 3E01            MVI     A,1             ;ORIGINATE MODE
0424 320310          STA     MFLAG           ;SET TO  MODE
0427 4F              MOV     C,A
0428 CD3D0A          CALL    MODE
042B 1E1E      COMM1: MVI    E,30            ;SET FOR 30 SECONDS
042D CDE402          CALL    PRINT
```

```
0430 5741495449          DB      'WAITING FOR CARRIER, PRESS ANY KEY TO ABORT.',CR
045D 0E14       CM1:     MVI     C,20            ;FOR 1 SECOND DELAY
045F CDA50A              CALL    DELAY           ;WAIT A SECOND
0462 CDFC09              CALL    CDSTAT          ;LOOK FOR A CARRIER
0465 B7                  ORA     A               ;SET FLAGS
0466 C29604              JNZ     TALK
0469 CDAD09              CALL    CONST           ;CHECK CONSOLE
046C B7                  ORA     A
046D C27404              JNZ     CM2             ;ABORT IF ANY KEY PRESSED
0470 1D                  DCR     E               ;COUNT SECONDS
0471 C25D04              JNZ     CM1             ;LOOP 30 TIMES
0474 CDE402     CM2:     CALL    PRINT
0477 4E4F204341          DB      'NO CARRIER RECEIVED',BELL,CR
048C CDAD09              CALL    CONST           ;CHECK CONSOLE
048F B7                  ORA     A
0490 C4A109              CNZ     CONIN           ;THROW AWAY CHARACTER
0493 C37505              JMP     HANGUP

                ;THIS IS THE COMMUNICATIONS LOOP. CHARACTERS FROM THE
                ;LINE ARE SENT TO THE CONSOLE.  CHRACTERS FROM THE
                ;CONSOLE ARE SENT TO THE PHONE LINE.


0496 CDFC09     TALK:    CALL    CDSTAT
0499 B7                  ORA     A
049A CAD104              JZ      TLK2
049D 0E01                MVI     C,1
049F CD150A              CALL    TXON            ;TURN ON CARRIER
04A2 CDE402              CALL    PRINT
04A5 434F4E4E45          DB      'CONNECTION ESTABLISHED',CR
04BC CDAD09     TLK1:    CALL    CONST           ;CHECK FOR CONSOLE CHAR.
04BF B7                  ORA     A
04C0 C4EE04              CNZ     CSEND           ;SEND IT IF PRESENT
04C3 CDF709              CALL    RXSTAT          ;SEE IF MODEM HAS A CHARN
04C6 B7                  ORA     A
04C7 C41005              CNZ     CRECV           ;GET IT IF PRESENT
04CA CDFC09              CALL    CDSTAT          ;CHECK FOR CARRIER
04CD B7                  ORA     A
04CE C2BC04              JNZ     TLK1            ;IF CARRIER, CONTINUE
04D1 CDDF02     TLK2:    CALL    CRLF
04D4 CDE402              CALL    PRINT
04D7 5B2A204C4F          DB      '[* LOST CARRIER *]',BELL,CR
04EB C37505              JMP     HANGUP


                ;SEND A CHARACTER TO MODEM.
                ;CONSOLE CHARACTERS ARE CHECKED FOR 2 CONTROL CODES.
                ;CONTROL A WILL ESCAPE TO THE COMMAND MODE.
                ;CONTROL B WILL CAUSE A BREAK TO BE TRANSMITTED.
                ;NOTE THAT CONTROL A OR B CAN'T BE SENT TO THE MODEM.


04EE CDA109     CSEND:   CALL    CONIN           ;GET CHAR. FROM CONSOLE
04F1 FE01                CPI     1               ;CONTROL A?
04F3 CADB01              JZ      CMND            ;BACK TO COMMAND MODE
04F6 0E03                MVI     C,3             ;PREPARE FOR 150 MS BREAK
```

```
04F8 FE17               CPI     17H             ;CONTROL W?
04FA CA510A             JZ      BREAK           ;YES, SEND BREAK
04FD 4F                 MOV     C,A
04FE CDE209     XMIT:   CALL    TXCHR           ;SEND TO MODEM
0501 3A0310             LDA     MFLAG           ;GET MODE 0=ANS 1=ORG
0504 B7                 ORA     A
0505 CAB709             JZ      CONOUT          ;ECHO IF ANSWER MODE
0508 3A0210             LDA     DPLX            ;GET DUPLEX FLAG 1=FULL
050B B7                 ORA     A
050C CAB709             JZ      CONOUT          ;ECHO IF HALF DUPLEX
050F C9                 RET

                ;RECEIVES CHARACTERS FROM THE MODEM AND SENDS THEM
                ; TO THE CONSOLE.  IF IN FULL DUPLEX ANSWER MODE THE
                ; CHARACTERS WILL BE ECHOED BACK TO THE MODEM.  IN
                ; ORIGINATE MODE OR HALF DUPLEX CHARACTERS WILL NOT
                ;BE ECHOED BACK.
                ;IT ALSO CALLS THE CAPTURE ROUTINE WHICH WILL PUT
                ;THE RECEIVED CHARACTERS IN THE BUFFER IF IT IS ENABLED.

0510 CDED09     CRECV:  CALL    RXCHR           ;GET CHARACTER FROM MODEM
0513 4F                 MOV     C,A
0514 3A3310             LDA     BUGFLAG         ;SEE IF WE PRINT IT IN HEX
0517 B7                 ORA     A
0518 CA3205             JZ      REC1            ;JUMP IF NOT HEX DISP. MODE
051B CD5305             CALL    PRNHEX
051E 0E20               MVI     C,20H           ;ASCII SPACE
0520 CDB709             CALL    CONOUT          ;SPACE BETWEEN HEX NUMBERS
0523 3A3410             LDA     BUGCOUNT
0526 3D                 DCR     A
0527 323410             STA     BUGCOUNT
052A E60F               ANI     0FH
052C CCDF02             CZ      CRLF
052F C34305             JMP     REC2
0532 CD0E0A     REC1:   CALL    ERFLG           ;GER __T ERROR FLAGS
0535 E601               ANI     1
0537 CA3C05             JZ      REC3
053A 0E23               MVI     C,'#'           ;# SUBS FOR CHAR. WITH PARITY ERROR
                REC3:
053C 79                 MOV     A,C
053D E67F               ANI     7FH             ;KILL MSB
053F 4F                 MOV     C,A
0540 CDB709             CALL    CONOUT          ;SEND TO CONSOLE
0543 CDFF06     REC2:   CALL    CPBFIL          ;ALSO TO CAPTURE BUFFER
0546 3A0310             LDA     MFLAG           ;GET MODE  1=ORG 0=ANS
0549 B7                 ORA     A
054A C0                 RNZ                     ;DON'T ECHO TO LINE IF ORIG.
054B 3A0210             LDA     DPLX            ;GET DUPLEX FLAG  1=FULL
054E B7                 ORA     A
054F C2E209             JNZ     TXCHR           ;ECHO BACK TO SENDER
0552 C9                 RET

                ;PRINTS INCOMMING CHARACTERS IN HEX WITH THE MODEM STATUS WORD.
```

```
0553 C5          PRNHEX: PUSH    B       ;CHARACTER IS IN C, SAVE IT
0554 79                  MOV     A,C
0555 0F                  RRC
0556 0F                  RRC
0557 0F                  RRC
0558 0F                  RRC
0559 CD6405              CALL    PRNH1
055C C1                  POP     B       ;RESTORE CHARACTER
055D C5                  PUSH    B       ;SAVE AGAIN
055E 79                  MOV     A,C
055F CD6405              CALL    PRNH1   ;PRINT LOW NIBBLE
0562 C1                  POP     B
0563 C9                  RET

0564 E60F        PRNH1:  ANI     0FH     ;KEEP LOW 4 BITS
0566 C630                ADI     30H     ;ASCII BIAS ADDED
0568 4F                  MOV     C,A
0569 FE3A                CPI     3AH
056B DAB709              JC      CONOUT
056E 3E07                MVI     A,7
0570 81                  ADD     C       ;MAKE LETTER A-F
0571 4F                  MOV     C,A
0572 C3B709              JMP     CONOUT



                 ;THIS ROUTINE HANGS UP AND TURNS OFF THE TX CARRIER.

0575 0E00        HANGUP: MVI     C,0
0577 CD150A              CALL    TXON            ;TRANSMIT CARRIER OFF
057A 0E00                MVI     C,0
057C CD290A              CALL    SWHOOK          ;GO ON-HOOK
057F CDE402              CALL    PRINT
0582 5B2048554E          DB      '[ HUNG UP ]',BELL,CR
058F 0E14                MVI     C,20            ;SET FOR 1 SECOND DELAY
0591 CDA50A              CALL    DELAY           ;WAIT FOR THINGS TO DISCONNECT
0594 C9                  RET

                 ;***** THE FOLLOWING ROUTINES SET OPTIONS ******

                 ;NOTE THAT MOST DON'T CHECK FOR VALID RANGE OF VARIABLES.
                 ;IT IS POSSIBLE FOR EXAMPLE TO SET THE STOP BITS TO 7 OR 9
                 ;OR OTHER IMPOSSIBLE NUMBER.  IN THIS CASE 1 OR 2 STOP BITS
                 ;WOULD ACTUALLY BE SET EVEN THOUGH NO ERROR MESSAGE WAS
                 ;GENERATED.


                 ;THIS ROUTINE LETS THE USER CHANGE THE BAUD RATE.

0595 CDB702      BRATE:  CALL    GETHEX          ;GET A DIGIT
0598 7B                  MOV     A,E
```

```
0599 FE01              CPI     1               ;110 BAUD?
059B CAB405            JZ      BR110
059E FE03              CPI     3
05A0 C2C505            JNZ     BADBR           ;BAD IF NOT 110 OR 300
05A3 CDE402            CALL    PRINT
05A6 3330302042        DB      '300 BAUD',CR
05AF 0E01              MVI     C,1
05B1 C31F0A            JMP     BAUD            ;SET RATE

05B4 CDE402    BR110:  CALL    PRINT
05B7 3131302042        DB      '110 BAUD',CR
05C0 0E00              MVI     C,0
05C2 C31F0A            JMP     BAUD

05C5 CDE402    BADBR:  CALL    PRINT
05C8 4241442042        DB      'BAD BAUD RATE',CR
05D6 C9                RET

               ;LETS USER CHANGE LENGTH OF DATA WORD (5,6,7,8 BITS).

05D7 CDB702    LENGTH: CALL    GETHEX          ;GET LENGTH
05DA CDEF05            CALL    PDIGIT          ;PRINT IT
05DD CDE402            CALL    PRINT
05E0 2044415441        DB      ' DATA BITS',CR
05EB 4B                MOV     C,E
05EC C3870A            JMP     WLS             ;SET MODEM BOARD


05EF 3E30      PDIGIT: MVI     A,30H
05F1 83                ADD     E               ;MAKE NUMBER IN E ASCII
05F2 4F                MOV     C,A
05F3 CDB709            CALL    CONOUT          ;PRINT IT
05F6 C9                RET


               ;SET NUMBER OF STOP BITS ( 1 OR 2).

05F7 CDB702    SBITS:  CALL    GETHEX
05FA CDEF05            CALL    PDIGIT
05FD CDE402            CALL    PRINT
0600 2053544F50        DB      ' STOP BITS',CR
060B 4B                MOV     C,E
060C C39A0A            JMP     NSTOP           ;SET THE MODEM


               ;SETS FULL DUPLEX MODE  ( F COMMAND)

060F 3E01      FULL:   MVI     A,1
0611 320210            STA     DPLX            ;SET FLAG
0614 CDE402            CALL    PRINT
0617 46554C4C20        DB      'FULL DUPLEX',CR
0623 C9                RET

               ;SET HALF DUPLEX MODE  (H COMMAND)

0624 AF        HALF:   XRA     A
```

```
0625 320210            STA     DPLX            ;CLEAR FLAG
0628 CDE402            CALL    PRINT
062B 48414C4620        DB      'HALF DUPLEX',CR
0637 C9                RET

               ;SET PARITY,  N=NO PARITY   E=EVEN PARITY  O=ODD PARITY

0638 CD9F02    PARTYP: CALL    NXTCHR          ;FIND NEXT CHARACTER IN BUFF.
063B CD8302            CALL    NEXT            ;GET IT
063E FE4E              CPI     'N'
0640 CA4E06            JZ      NPAR
0643 FE45              CPI     'E'
0645 CA6006            JZ      EPAR
0648 FE4F              CPI     'O'
064A CA7406            JZ      OPAR
064D C9                RET

064E CDE402    NPAR:   CALL    PRINT
0651 4E4F205041        DB      'NO PARITY',CR
065B 0E00              MVI     C,0
065D C36C0A            JMP     PARITY

0660 CDE402    EPAR:   CALL    PRINT
0663 4556454E20        DB      'EVEN PARITY',CR
066F 0E02              MVI     C,2
0671 C36C0A            JMP     PARITY

0674 CDE402    OPAR:   CALL    PRINT
0677 4F44442050        DB      'ODD PARITY',CR
0682 0E01              MVI     C,1
0684 C36C0A            JMP     PARITY

               ;THIS ROUTINE ALLOWS THE USER TO JUMP TO
               ;AN EXTERNAL PROGRAM AND EXECUTE IT.
               ;FOR EXAMPLE, IF YOU HAD A ROM MONITOR
               ;AT F000 HEX, YOU WOULD ENTER THE COMMAND: J F000
               ;

0687 CD9F02    JUMP:   CALL    NXTCHR          ;FIND ADDRESS
068A CDB702            CALL    GETHEX          ;CONVERT TO BINARY IN DE
068D EB                XCHG                    ;ADDRESS TO HL
068E E9                PCHL                    ;JUMP TO ROUTINE

               ;THIS ROUTINE SETS UP THE THE CAPTURE FUNCTION
               ;COMMAND FORMAT:   C0  RESETS THE BUFFER POINTER
               ;AND DISABLES CAPTURE.
               ;C1   ENABLES CAPTURE AT CURRENT BUFFER ADDRESS
               ;C2   STOPS CAPTURE AT CURRENT ADDRESS.


               CAPTURE:
068F CD9F02            CALL    NXTCHR          ;FIND DIGIT
0692 B7                ORA     A
0693 C8                RZ                      ;RETURN IF NO ARG.
```

```
0694 CD8302             CALL    NEXT            ;GET DIGIT
0697 FE30               CPI     '0'
0699 CAD406             JZ      RSBUF           ;IF ZERO, RESET
069C FE31               CPI     '1'
069E CABB06             JZ      STRT            ;IF 1 START CAPTURE MODE
06A1 FE32               CPI     '2'
06A3 C0                 RNZ                     ;IF 2 STOP CAPTURE
06A4 AF                 XRA     A
06A5 323210             STA     CPFLAG          ;CLEAR CAPTURE FLAG
06A8 CDE402             CALL    PRINT
06AB 4341505455         DB      'CAPTURE STOPED',CR
06BA C9                 RET

06BB 3E01       STRT:   MVI     A,1
06BD 323210             STA     CPFLAG          ;SET CAPTURE FLAG
06C0 CDE402             CALL    PRINT
06C3 4341505455         DB      'CAPTURE ENABLED',CR
06D3 C9                 RET

06D4 21DE10     RSBUF:  LXI     H,CPBUF         ;START OF CAPTURE BUFFER
06D7 223610             SHLD    CBUFP           ;STORE POINTER
06DA AF                 XRA     A
06DB 323210             STA     CPFLAG          ;CLEAR FLAG
06DE 323810             STA     CPCNT           ;CLEAR BYTE COUNTER
06E1 323910             STA     CPCNT+1
06E4 CDE402             CALL    PRINT
06E7 4341505455         DB      'CAPTURE BUFFER CLEARED',CR
06FE C9                 RET

                ;THIS ROUTINE FILLS THE CAPTURE BUFFER WITH CHARACTERS
                ;FROM THE MODEM IF THE "CPFLAG" IS SET TO A 1.
                ;THE BUFFER POINTER IS STORED AT "CBUFP".

06FF 3A3210     CPBFIL: LDA     CPFLAG          ;GET FLAG
0702 B7                 ORA     A
0703 C8                 RZ                      ;RETURN IF NOT ENABLED
0704 E5                 PUSH    H               ;SAVE HL
0705 2A3610             LHLD    CBUFP           ;GET BUFFER POINTER
0708 71                 MOV     M,C             ;PUT CHARACTER IN BUFFER
0709 23                 INX     H               ;ADVANCE POINTER
070A 3A3510             LDA     MTOP            ;TOP OF MEMORY
070D BC                 CMP     H               ;CHECK FOR "OUT OF MEMORY"
070E CA1D07             JZ      OUTMEM
0711 223610             SHLD    CBUFP           ;SAVE POINTER
0714 2A3810             LHLD    CPCNT           ;GET CHARACTER COUNTER
0717 23                 INX     H               ;INCREMENT IT
0718 223810             SHLD    CPCNT           ;STORE IT
071B E1                 POP     H
071C C9                 RET

071D CDDF02     OUTMEM: CALL    CRLF
0720 CDE402             CALL    PRINT
0723 4341505455         DB      'CAPTURE BUFFER FULL',BELL,CR
0738 AF                 XRA     A
0739 323210             STA     CPFLAG          ;DISABLE CAPTURE MODE
073C E1                 POP     H
```

```
073D C9                 RET


                ;TYPES THE CAPTURE BUFFER TO THE CONSOLE.


                ;T   WILL TYPE STARTING AT THE BEGINNING
                ;T FOLLOWED BY A HEX NUMBER WILL CAUSE PRINTING TO
                ;START AT A POINT SPECFIED BY THE NUMBER. FOR EXAMPLE
                ;T 100   WILL START TYPING AT A POINT 100 HEX
                ;CHARACTERS INTO THE BUFFER.  S STARTS
                ;AND STOPS THE PRINTING. CARRIAGE RETURN ABORTS
                ;THIS FUNCTION.

073E CDB702     TYPE:   CALL    GETHEX
0741 21DE10             LXI     H,CPBUF         ;STARTING ADDRESS OF BUFFER
0744 19                 DAD     D               ;ADD USERS COUNT TO STARTING ADR.
0745 3A3810     TY1:    LDA     CPCNT           ;SEE IF WE'RE AT END
0748 BB                 CMP     E
0749 C25107             JNZ     TY0
074C 3A3910             LDA     CPCNT+1
074F BA                 CMP     D
0750 C8                 RZ                      ;QUIT AT END OF BUFFER
0751 4E         TY0:    MOV     C,M             ;GET A CHARACTER
0752 CDB709             CALL    CONOUT          ;PRINT IT
0755 CDAD09             CALL    CONST           ;CHECK FOR CONSOLE INPUT
0758 B7                 ORA     A
0759 CA7907             JZ      TY2
075C CDA109             CALL    CONIN           ;GET CONSOLE
075F FE0D               CPI     CR              ;WANT TO ABORT?
0761 C8                 RZ                      ;IF YES THEN RETURN
0762 FE53               CPI     'S'             ;WANT TO STOP?
0764 C27907             JNZ     TY2
0767 CDAD09     TY3:    CALL    CONST
076A B7                 ORA     A
076B CA6707             JZ      TY3             ;WAIT FOR ANOTHER S
076E CDA109             CALL    CONIN
0771 FE0D               CPI     CR
0773 C8                 RZ
0774 FE53               CPI     'S'
0776 C26707             JNZ     TY3
0779 23         TY2:    INX     H               ;ADVANCE POINTERS
077A 13                 INX     D
077B C34507             JMP     TY1             ;LOOP BACK FOR MORE


077E CD8302     DEBUG:  CALL    NEXT
0781 FE30               CPI     '0'
0783 CAA407             JZ      DBOFF
0786 FE31               CPI     '1'             ;DEBUG ON?
0788 C0                 RNZ
0789 CDE402             CALL    PRINT
078C 4445425547         DB      'DEBUG MODE ON',CR
079A 3E01               MVI     A,1
079C 323310             STA     BUGFLAG
079F 3D                 DCR     A
```

```
07A0 323410             STA     BUGCOUNT
07A3 C9                 RET
07A4 CDE402     DBOFF:  CALL    PRINT
07A7 4445425547         DB      'DEBUG MODE OFF',CR
07B6 AF                 XRA     A
07B7 323310             STA     BUGFLAG
07BA C9                 RET


                ;THIS IS THE COMMAND DECODING TABLE.

                ;EACH ENTRY IS AN ASCII LETTER FOLLOWED BY AN ADDRESS
                ;OF THE ROUTINE WHICH IS CALLED TO EXECUTE THAT COMMAND.
                ;THE END OF THE TABLE IS MARKED WITH A ZERO.


07BB 58         TABLE:  DB      'X'             ;EXIT TO CP/M
07BC 0303               DW      QUIT
07BE 44                 DB      'D'             ;DIAL A NUMBER
07BF 5603               DW      DIAL
07C1 41                 DB      'A'             ;ANSWER CALLS
07C2 AA03               DW      ANSWR
07C4 47                 DB      'G'             ;"GOODBYE" HANG UP PHONE
07C5 7505               DW      HANGUP
07C7 42                 DB      'B'             ;CHANGE BAUD RATE
07C8 9505               DW      BRATE
07CA 53                 DB      'S'             ;SET STOP BITS
07CB F705               DW      SBITS
07CD 4C                 DB      'L'             ;SET DATA WORD LENGTH
07CE D705               DW      LENGTH
07D0 50                 DB      'P'             ;SET PARITY MODE
07D1 3806               DW      PARTYP
07D3 46                 DB      'F'             ;SET FULL DUPLEX
07D4 0F06               DW      FULL
07D6 48                 DB      'H'             ;HALF DUPLEX
07D7 2406               DW      HALF
07D9 4A                 DB      'J'             ;JUMP TO ADDRESS
07DA 8706               DW      JUMP
07DC 43                 DB      'C'             ;CAPTURE
07DD 8F06               DW      CAPTURE
07DF 54                 DB      'T'             ;TYPE BUFFER TO CONSOLE
07E0 3E07               DW      TYPE
07E2 57                 DB      'W'             ;WRITE CAPTURE BUFFER TO DISK
07E3 0E09               DW      SAVE
07E5 52                 DB      'R'             ;READ AND TRANSMIT DISK FILE
07E6 8D08               DW      READ
07E8 3F                 DB      '?'             ;SET TO DEBUG MODE
07E9 7E07               DW      DEBUG
07EB 00                 DB      0               ;END OF TABLE
                ;********** OPTIONAL DISK I/O AREA **********************


                ;THESE ROUTINES ALLOW YOU TO SAVE THE CONTENTS OF
                ;THE CAPTURE BUFFER ON THE DISK.  IF YOU DON'T HAVE
                ;THE CP/M DISK OPERATING SYSTEM THEY WON'T WORK.
```

```
                ;DISK I/O ROUTINES

07EC 11BD10     DELETE: LXI     D,FCB
07EF 0E13               MVI     C,19
07F1 C30500             JMP     BDOS

07F4 11BD10     MAKE:   LXI     D,FCB
07F7 0E16               MVI     C,22
07F9 C30500             JMP     BDOS

07FC E5         SETDMA: PUSH    H
07FD 0E1A               MVI     C,26
07FF CD0500             CALL    BDOS
0802 E1                 POP     H
0803 C9                 RET

                WRITENR:
0804 E5                 PUSH    H
0805 11BD10             LXI     D,FCB
0808 0E15               MVI     C,21
080A CD0500             CALL    BDOS
080D E1                 POP     H
080E C9                 RET

                READNR:
080F E5                 PUSH    H
0810 11BD10             LXI     D,FCB
0813 0E14               MVI     C,20
0815 CD0500             CALL    BDOS
0818 E1                 POP     H
0819 C9                 RET

081A 11BD10     CLOSE:  LXI     D,FCB
081D 0E10               MVI     C,16
081F C30500             JMP     BDOS

0822 11BD10     OPEN:   LXI     D,FCB
0825 0E0F               MVI     C,15
0827 C30500             JMP     BDOS


                ;GET NAME OF FILE FROM CONSOLE BUFFER TO FILE CONTTROL BLOCK

                GETNAME:
082A CD9F02             CALL    NXTCHR          ;SET BUFFER POINTER TO FILE NAME
082D B7                 ORA     A               ;TEST FOR NO FILE NAME
082E C8                 RZ                      ;RETURN IF NONE ENTERED
082F 21BD10             LXI     H,ET    ;POINT TO ENTRY TYPE
0832 AF                 XRA     A               ;ZERO A
0833 060C               MVI     B,12            ;LOAD COPUTER
0835 77         G1:     MOV     M,A             ;FILL FCB AREA WITH 0 + 11 SPACES
0836 3E20               MVI     A,20H           ;A SPACE
0838 23                 INX     H               ;BUMP POINTER
0839 05                 DCR     B               ;COUNT'EM
```

```
083A C23508             JNZ     G1
083D 0608       MVI     B,8                     ;MAX OF 8 CHARS. IN FILE NAME
083F 11BE10             LXI     D,FNAME         ;POINT TO FCB FILE NAME AREA
0842 CD6508             CALL    MVNAM           ;MOVE NAME FROM BUFFER TO FCB
0845 FE2E               CPI     '.'
0847 C25208             JNZ     G2              ;IF NO PERIOD THEN NO FILE TYPE
084A 11C610             LXI     D,FTYPE         ;POINT TO FILE TYPE AREA
084D 0603               MVI     B,3             ;MAX OF 3 CHARS. IN FILE TYPE
084F CD6508             CALL    MVNAM           ;MOVE TYPE INTO FCB AREA
0852 21C910     G2:     LXI     H,EX            ;FILL REST OF FCB WITH ZEROS
0855 0615               MVI     B,(NR+1)-EX     ;COUNTER
0857 3600       GG3:    MVI     M,0
0859 23                 INX     H
085A 05                 DCR     B
085B C25708             JNZ     GG3
085E 118000             LXI     D,80H           ;SET DMA ADDRESS TO 80H
0861 CDFC07             CALL    SETDMA
0864 C9                 RET

                ;MOVES STRINGS FROM CONSOLE BUFFER TO FCB AREA
                ;DE POINT TO DESTINATION
                ;B HAS NUMBER OF CHARACTERS TO MOVE

0865 CD8302     MVNAM:  CALL    NEXT            ;GET NEXT CHARACTER
0868 FE2E               CPI     '.'             ;END OF FILE NAME?
086A C8                 RZ
086B FE0D               CPI     CR              ;END OF LINE?
086D C8                 RZ
086E FE00               CPI     0               ;DELIMITER?
0870 C8                 RZ
0871 12                 STAX    D               ;PUT CAHRACTER IN FCB
0872 13                 INX     D               ;ADVANCE POINTER
0873 05                 DCR     B
0874 C26508             JNZ     MVNAM           ;LOOP IF MORE CHARACTERS
0877 C38302             JMP     NEXT            ;GET NEXT CHARACTER AND RETURN

087A 21C910             LXI     H,EX    ;POINT TO EXTENT AREA OF FCB
087D 0615               MVI     B,(NR+1)-EX
087F 3600       G3:     MVI     M,0     ;FILL REST OF FCB WITH ZERO
0881 23                 INX     H
0882 05                 DCR     B
0883 C27F08             JNZ     G3
0886 118000             LXI     D,80H
0889 CDFC07             CALL    SETDMA
088C C9                 RET

                ;DISK FILE TRANSMIT ROUTINE

088D CD2A08     READ:   CALL    GETNAME         ;GET FILE NAME
0890 CD2208             CALL    OPEN
0893 FEFF               CPI     255     ;OK?
0895 C2A608             JNZ     R1              ;JUMP IF OPEN IS OK
0898 CDE402             CALL    PRINT
089B 4E4F542046         DB      'NOT FOUND',CR
```

```
08A5 C9                 RET
08A6 CD0F08     R1:     CALL    READNR          ;READ 128 BYTES INTO 80H BUFFER
08A9 CDB508             CALL    TXBUF           ;TRANSMIT THE BUFFER CONTENTS
08AC FE1A               CPI     1AH             ;CONTROL Z (END OF FILE)
08AE C2A608             JNZ     R1
08B1 CD1A08             CALL    CLOSE
08B4 C9                 RET

                ;TRANSMITS THE CONTENTS OF THE DISK BUFFER

08B5 218000     TXBUF:  LXI     H,80H
08B8 0680               MVI     B,128
08BA 4E         TXB1:   MOV     C,M             ;GET A CHARACTER
08BB CDFE04             CALL    XMIT            ;TRANSMIT IT
08BE 79                 MOV     A,C
08BF FE1A               CPI     1AH             ;END OF FILE?
08C1 C8                 RZ                      ;YES RETURN
08C2 FE0D               CPI     CR              ;END OF LINE?
08C4 CCE808             CZ      WTRPLY          ;WAIT FOR POSSIBLE PROMPT
08C7 CDF709             CALL    RXSTAT          ;SEE IF CHAR. HAS BEEN RECEIVED
08CA B7                 ORA     A
08CB C41005             CNZ     CRECV           ;GET IT AND SEND TO CONSOLE
08CE CDFC09             CALL    CDSTAT          ;CHECK FOR CARRIER
08D1 B7                 ORA     A
08D2 CADF08             JZ      ABRT            ;ABORT IF NO CARRIER
08D5 CDAD09             CALL    CONST           ;WANT TO ABORT?
08D8 B7                 ORA     A
08D9 CAE208             JZ      TXB2            ;JUMP IF NOT
08DC CDA109             CALL    CONIN           ;ABORT ON ANY KEY PRESSED
08DF 3E1A       ABRT:   MVI     A,1AH           ;FAKE OUT THE CALLING ROUTINE
08E1 C9                 RET

08E2 23         TXB2:   INX     H               ;BUMP BUFFER POINTER
08E3 05                 DCR     B               ;COUNT BYTES
08E4 C2BA08             JNZ     TXB1
08E7 C9                 RET

                ;THIS ROUTINE WAITS FOR 150 MS AFTER THE LAST
                ;CHARACTER IS RECEIVED BY THE MODEM.

08E8 C5         WTRPLY: PUSH    B
08E9 CDF709             CALL    RXSTAT
08EC B7                 ORA     A
08ED C4ED09             CNZ     RXCHR           ;FLUSH UART
08F0 0603       WTRP1:  MVI     B,3             ;150 MS
08F2 CDB80A     WTRP2:  CALL    STIME           ;START TIMER
08F5 CDF709     WTRP3:  CALL    RXSTAT          ;MODEM CHAR. RECEIVED?
08F8 B7                 ORA     A
08F9 F5                 PUSH    PSW             ;SAVE STATUS
08FA C41005             CNZ     CRECV           ;GET AND SEND TO CONSOLE
08FD F1                 POP     PSW             ;POP FLAGS AGAIN
08FE C2F008             JNZ     WTRP1           ;IF CHARACTER REC. LOOP BACK
0901 CDBB0A             CALL    CKTIME          ;TIME UP?
0904 B7                 ORA     A
0905 CAF508             JZ      WTRP3           ;NO, LOOP BACK
0908 05                 DCR     B               ;COUNT LOOPS
```

```
0909 C2F208             JNZ     WTRP2           ;LOOP IF NOT 150 MS
090C C1                 POP     B               ;RESTORE BC
090D C9                 RET


                ;THIS ROUTINE STORES THE CONTENTS OF THE CAPTURE
                ;BUFFER ON THE DISK.


090E CD2A08     SAVE:   CALL    GETNAME
0911 CDEC07             CALL    DELETE  ;ERASE OLD FILE OF SAME NAME
0914 CDF407             CALL    MAKE    ;MAKE NEW FILE
0917 FEFF               CPI     255
0919 C23309             JNZ     S1
091C CDE402             CALL    PRINT
091F 4E4F204449         DB      'NO DIRECTORY SPACE',CR
0932 C9                 RET

0933 CD2208     S1:     CALL    OPEN
0936 2A3610             LHLD    CBUFP           ;CAPTURE BUFFER POINTER
0939 EB                 XCHG                    ;POINTER TO DE
093A 2A3810             LHLD    CPCNT           ;BUFFER CAHRACTER COUNT
093D 3E1A       S2:     MVI     A,1AH           ;END OF FILE MARKER
093F 12                 STAX    D
0940 13                 INX     D
0941 23                 INX     H               ;FILL OUT WITH 1A'S TILL
0942 7D                 MOV     A,L             ;EVEN WITH 128 BYTE BOUNDRY
0943 E67F               ANI     7FH
0945 C23D09             JNZ     S2
0948 29                 DAD     H               ;SHIFT HL LEFT 1 BIT
0949 CE00               ACI     0               ;SHIFT CARRY INTO REG. A
094B 47                 MOV     B,A             ;MSB OF RECORD COUNT TO B
094C 4C                 MOV     C,H             ;LSB OF RECORD COUNT IN C
094D 03                 INX     B               ;ADD 1 MORE
094E 11DE10             LXI     D,CPBUF         ;CAPTURE BUFFER START ADDRESS
0951 C5         S3:     PUSH    B               ;SAVE RECORD COUNT
0952 D5                 PUSH    D               ;SAVE BUFFER POINTER
0953 CDFC07             CALL    SETDMA          ;DMA=BUFFER
0956 CD0408             CALL    WRITENR         ;WRITE A RECORD TO DISK
0959 B7                 ORA     A               ;CHECK FOR ERROR
095A C28E09             JNZ     ERRR
095D D1                 POP     D               ;GET CAPTURE BUFFER POINTER
095E 218000             LXI     H,128
0961 19                 DAD     D               ;ADVANCE IT 128 BYTES
0962 EB                 XCHG
0963 C1                 POP     B               ;GET RECORD COUNT
0964 0B                 DCX     B               ;COUNT 1
0965 78                 MOV     A,B
0966 B1                 ORA     C               ;TEST FOR ZERO
0967 C25109             JNZ     S3              ;IF NOT ZERO, LOOP BACK
096A 118000             LXI     D,80H           ;RESET DMA TO 80H
096D CDFC07             CALL    SETDMA
0970 CD1A08             CALL    CLOSE
0973 FEFF               CPI     255
```

```
0975 C2D406            JNZ     RSBUF           ;CLEAR CAPTURE BUFFER
0978 CDE402            CALL    PRINT
097B 43414E4E4F        DB      'CANNOT CLOSE FILE',CR
098D C9                RET

098E E1        ERRR:   POP     H
098F E1                POP     H       ;CLEAN UP STACK
0990 CDE402            CALL    PRINT
0993 4F5554204F        DB      'OUT OF SPACE',CR
09A0 C9                RET

                ;******* CONSOLE I/O AREA FOR CP/M ******************


                ;CONSOLE INPUT
                ;IT RETURNS THE CHARACTER IN REGISTER A


09A1 E5        CONIN:  PUSH    H
09A2 D5                PUSH    D
09A3 C5                PUSH    B
09A4 CD0A10            CALL    XCONIN
09A7 E67F              ANI     7FH             ;KILL MSB
09A9 C1                POP     B
09AA D1                POP     D
09AB E1                POP     H
09AC C9                RET

                ;CONSOLE STATUS CHECK
                ;RETURNS WITH A=0 IF NO CHARACTER TYPED
                ;OR A=1 IF CHARACTER HAS BEEN TYPED (OR A= NOT ZERO)

09AD E5        CONST:  PUSH    H
09AE D5                PUSH    D
09AF C5                PUSH    B
09B0 CD0710            CALL    XCONST
09B3 C1                POP     B
09B4 D1                POP     D
09B5 E1                POP     H
09B6 C9                RET

                ;CONSOLE OUTPUT.  CHARACTER TO BE PRINTED MUST BE IN
                ; REG. C
```

```
09B7 E5        CONOUT: PUSH    H
09B8 D5                PUSH    D
09B9 C5                PUSH    B
09BA CD0D10            CALL    XCONOT
09BD C1                POP     B
09BE D1                POP     D
09BF E1                POP     H
09C0 C9                RET

               ;INITIALIZATION ROUTINE FOR USER HARDWARE AND SOFTWARE.
               ;PUT YOUR OWN SPECIAL INITIALIZATION ROUTINES HERE.

               ;THIS ROUTINE MOVES THE CP/M CBIOS JUMP TABLE TO
               ;"JTAB" IN RAM.


09C1 2A0100    INIT:   LHLD    1               ;GET ADDRESS OF CBIOS
09C4 110410            LXI     D,JTAB          ;DESTINATION ADDRESS
09C7 060C              MVI     B,4*3           ;NO. OF BYTES TO MOVE
09C9 7E        MOVE:   MOV     A,M             ;GET A BYTE
09CA 12                STAX    D               ;MOVE IT
09CB 23                INX     H               ;ADVANCE POINTERS
09CC 13                INX     D
09CD 05                DCR     B
09CE C2C909            JNZ     MOVE            ;COUNT BYTES
09D1 3E01              MVI     A,1             ;SPECIAL HARDWARE STUFF
09D3 D384              OUT     84H
09D5 C9                RET

                       PAGE                    ;PRINTER TOP OF FORM
```

```
                ;********* MODEM I/O AREA ***********

                ;  ALL MODEM I/O AND CONTROL ARE HERE.


                ;       MODEM I/O SYSTEM        5-10-79


                ;       WRITTEN BY DALE A. HEATHERINGTON
                ;       COPYRIGHT 1979 D.C. HAYES ASSOCIATES, INC.


                ;       PORT EQUATES

0080 =          DATA    EQU     80H
0081 =          STATUS  EQU     DATA+1
0081 =          CR1     EQU     DATA+1
0082 =          CR2     EQU     DATA+2
0083 =          CR3     EQU     DATA+3


                ;       BIT FUNCTIONS

                ;STATUS REGISTER

0001 =          RRF     EQU     1       ;RECEIVE REGISTER FULL
0002 =          TRE     EQU     2       ;TRANSMITER HOLDING REGISTER EMPTY
0004 =          PE      EQU     4       ;PARITY ERROR
0008 =          FE      EQU     8       ;FRAMMING ERROR
0010 =          OE      EQU     10H     ;OVERRUN ERROR
0020 =          TMR     EQU     20H     ;TIMER STATUS
0040 =          CD      EQU     40H     ;CARRIER PRESENT
0080 =          RI      EQU     80H     ;NOT RING INDICATOR (LOW TRUE)

                ;CONTROL REGISTER 1  (CR1)

0001 =          EPE     EQU     1       ;EVEN PARITY ENABLE
0002 =          LS1     EQU     2       ;WORD LENGTH SELECT BIT 1
0004 =          LS2     EQU     4       ;WORD LENGTH SELECT BIT 2
0008 =          SBS     EQU     8       ;STOP BITS
0010 =          PI      EQU     10H     ;PARITY INHIBIT
0020 =          TMIE    EQU     20H     ;TIMER INTERUPTS ENABLE


                ;CONTROL REGISTER 2  (CR2)

0001 =          BRS     EQU     1       ;BAUD RATE CONTROL
0002 =          TXE     EQU     2       ;TRANSMIT CARRIER ENABLE
0004 =          MS      EQU     4       ;MODE (0=ANSWER 1=ORIGINATE)
0008 =          BRK     EQU     8       ;SEND BREAK
0010 =          ST      EQU     10H     ;SELF TEST
```

```
0020 =          TIE     EQU     20H     ;TRANSMITTER INTERUPT ENABLE
0040 =          RIE     EQU     40H     ;RECEIVER INTERUPT ENABLE
0080 =          OH      EQU     80H     ;OFF-HOOK

                ;CONTROL REGISTER 3  (CR3)

                ;WRITING ANYTHING TO CR3 STARTS THE 50 MS TIMER.



                ;THIS ROUTINE CLEARS ALL THE CONTROL REGISTERS

09D6 AF         RESET:  XRA     A       ;ZERO A
09D7 D381               OUT     CR1
09D9 D382               OUT     CR2
09DB 320110             STA     MICR2   ;CLEAR MEMORY IMAGES TOO
09DE 320010             STA     MICR1
09E1 C9                 RET


                ;THIS ROUTINE SENDS A BYTE IN REGISTER C TO THE MODEM
                ;TRANSMITTER. ONLY REGISTER A IS DISTURBED

09E2 DB81       TXCHR:  IN      STATUS  ;GET MODEM STATUS
09E4 E602               ANI     TXE     ;CHECK FOR TRANSMITER EMPTY
09E6 CAE209             JZ      TXCHR   ;LOOP IF NOT
09E9 79                 MOV     A,C
09EA D380               OUT     DATA    ;SEND THE BYTE
09EC C9                 RET


                ;GETS A BYTE FROM THE MODEM RECEIVER IN REGISTER A

09ED DB81       RXCHR:  IN      STATUS
09EF E601               ANI     RRF
09F1 CAED09             JZ      RXCHR   ;LOOP IF NO CHARACTER REC.
09F4 DB80               IN      DATA
09F6 C9                 RET             ;CHARACTER IN A


                ;THIS ROUTINE RETURNS WITH A=1 IF A RECEIVED
                ;CHARACTER IS WAITING OR A=0 IF NOT.

09F7 DB81       RXSTAT: IN      STATUS  ;GET MODEM STATUS
09F9 E601               ANI     RRF     ;CHECK RECEIVER FLAG
09FB C9                 RET             ;A=1 IF CHARACTER PRESENT


                ;CARRIER STATUS CHECK. A=1 IF CARRIER IS PRESENT

09FC DB81       CDSTAT: IN      STATUS
09FE E640               ANI     CD      ;TEST CARRIER DETECT FLAG
0A00 3E00               MVI     A,0     ;ASSUME 0
0A02 C8                 RZ
0A03 3C                 INR     A       ;MAKE A=1 IF CARRIER PRESENT
0A04 C9                 RET
```

```
                ;RING INDICATOR STATUS CHECK.  A=1 IF PHONE RINGING

0A05 DB81       RISTAT: IN      STATUS
0A07 E680               ANI     RI
0A09 3E01               MVI     A,1     ;ASSUME RINGING
0A0B C8                 RZ              ;RETURN IF IT IS (LOW TRUE BIT)
0A0C 3D                 DCR     A       ;ELSE MAKE A=0
0A0D C9                 RET

                ;GET THE ERROR FLAGS. ERROR BITS ARE RETURNED IN REG. A.
                ;A=0 IF NO ERRORS WERE DETECTED ON LAST CHARACTER RECEIVED.
                ;BIT 0=PARITY ERROR   BIT 1=FRAMMING ERROR
                ;BIT 2=OVERRUN ERROR

0A0E DB81       ERFLG:  IN      STATUS
0A10 1F                 RAR             ;SHIFT BITS RIGHT 2 PLACES
0A11 1F                 RAR
0A12 E607               ANI     7       ;KILL UNWANTED BITS
0A14 C9                 RET             ;LOW 3 BITS ARE ERROR FLAGS

                ;TRANSMIT CARRIER CONTROL.
                ;ENTER WITH REG. C=0 FOR CARRIER OFF
                ; OR REG. C=1 FOR TRANSMIT CARRIER ON

0A15 79         TXON:   MOV     A,C
0A16 B7                 ORA     A
0A17 3E02               MVI     A,TXE   ;PUT TXE BIT IN A
0A19 CA060B             JZ      CLR2    ;CLEAR IT IF 0
0A1C C3FC0A             JMP     SET2    ;SET IT IF A 1

                ;SET THE BAUD RATE.  C=0 FOR LOW BAUD RATE (110)
                ; C=1 FOR HIGH BAUD RATE (300)

0A1F 79         BAUD:   MOV     A,C
0A20 B7                 ORA     A
0A21 3E01               MVI     A,BRS   ;BAUD RATE BIT IN A
0A23 CA060B             JZ      CLR2    ;RESET BIT IF 0
0A26 C3FC0A             JMP     SET2    ;SET IF 1



                ;GO ON/OFF HOOK.   C=0 FOR ON-HOOK (IDLE)
                ;   C=1 FOR OFF-HOOK.


0A29 79         SWHOOK: MOV     A,C
0A2A B7                 ORA     A
0A2B 3E80               MVI     A,OH    ;SWITCH-HOOK BIT
0A2D CA060B             JZ      CLR2
0A30 C3FC0A             JMP     SET2


                ;GET SWITCH HOOK STATUS.
                ; A=0 ON HOOK    A=1 OFF HOOK
```

```
0A33 3A0110    SHSTAT: LDA     MICR2           ;GET CR2
0A36 E680              ANI     OH              ;MASK FOR OFF HOOK
0A38 3E01              MVI     A,1
0A3A C0                RNZ                     ;RET. IF OFF HOOK A=1
0A3B 3D                DCR     A
0A3C C9                RET                     ;ELSE A=0

               ;SET THE MODE.  C=0 FOR ANSWER MODE
               ;  C=1 FOR ORGINATE MODE

0A3D 79        MODE:   MOV     A,C
0A3E B7                ORA     A
0A3F 3E04              MVI     A,MS    ;MODE SELECT BIT
0A41 CA060B            JZ      CLR2
0A44 C3FC0A            JMP     SET2


               ;SET OR CLEAR SELF-TEST MODE.
               ;C=1 FOR SELF TEST     C=0 FOR NORMAL

0A47 79        SELFT:  MOV     A,C
0A48 B7                ORA     A
0A49 3E10              MVI     A,ST    ;SELF-TEST BIT
0A4B CA060B            JZ      CLR2    ;CLEAR IT
0A4E C3FC0A            JMP     SET2    ;OR SET IT


               ;SEND A BREAK CONSISTING OF SENDING A SPACE (LOW TONE) FOR
               ; A NUMBER OF 50 MS INTERVALS SPECIFIED IN THE REG. C.

0A51 C5        BREAK:  PUSH    B       ;SAVE BREAK DURATION VALUE
0A52 0E02              MVI     C,2
0A54 CDA50A            CALL    DELAY   ;WAIT 100 MS
0A57 C1                POP     B       ;RESTORE BREAK DURATION VALUE
0A58 E5                PUSH    H
0A59 210110            LXI     H,MICR2
0A5C 3E08              MVI     A,BRK   ;SET THE BREAK BIT
0A5E B6                ORA     M
0A5F D382              OUT     CR2
0A61 CDA50A            CALL    DELAY   ;C=50 MS INTERVALS TO WAIT
0A64 3E08              MVI     A,BRK
0A66 2F                CMA
0A67 A6                ANA     M       ;CLEAR BREAK BIT
0A68 D382              OUT     CR2     ;WRITE IT TO MODEM
0A6A E1                POP     H
0A6B C9                RET


               ;SETS THE PARITY.   C=0 FOR NO PARITY
               ;  C=1 FOR ODD PARITY    C=2 FOR EVEN PARITY

0A6C 79        PARITY: MOV     A,C
0A6D B7                ORA     A       ;TEST FOR NO PARITY
```

```
0A6E CA820A             JZ      PAROFF
0A71 3E10               MVI     A,PI
0A73 CD1B0B             CALL    CLR1    ;CLEAR PARITY INHIBIT BIT
0A76 79                 MOV     A,C
0A77 1F                 RAR             ;LSB IS NOW EVEN/ODD PARITY
0A78 E601               ANI     1
0A7A 3E01               MVI     A,EPE
0A7C CA1B0B             JZ      CLR1    ;CLEAR EPE IF ODD
0A7F C3110B             JMP     SET1    ;SET EPE IF EVEN

0A82 3E10       PAROFF: MVI     A,PI
0A84 C3110B             JMP     SET1    ;INHIBIT PARITY

                ;SELECT WORD LENGTH.
S               ;C=NUMBER OF DATA BITS (5,6,7,8)

0A87 79         WLS:    MOV     A,C
0A88 D605               SUI     5       ;REMOVE BIAS
0A8A 87                 ADD     A       ;SHIFT LEFT 1 PLACE
0A8B E606               ANI     6       ;KILL ANY STRAY BITS
0A8D 4F                 MOV     C,A     ;SAVE RESULT IN C
0A8E 3A0010             LDA     MICR1   ;GET MEMORY IMAGE
0A91 E6F9               ANI     NOT LS1+LS2     ;CLEAR WORD LENGTH BITS
0A93 B1                 ORA     C       ;PUT IN NEW ONES
0A94 320010             STA     MICR1   ;UPDATE MEMORY
0A97 D381               OUT     CR1     ;WRITE TO MODEM
0A99 C9                 RET

                ;SET NUMBER OF STOP BITS
                ;C=STOP BITS  (1 OR 2)

0A9A 79         NSTOP:  MOV     A,C
0A9B D601               SUI     1
0A9D 3E08               MVI     A,SBS   ;STOP BIT SELECT BIT
0A9F CA1B0B             JZ      CLR1    ;CLEAR IT
0AA2 C3110B             JMP     SET1    ;ELSE SET IT


                ;DELAY ROUTINE.
                ;WAITS 50 MS TIMES THE VALUE IN REGISTER C.

0AA5 79         DELAY:  MOV     A,C
0AA6 B7                 ORA     A
0AA7 C8                 RZ              ;WAIT ZERO TIME IF C=0
0AA8 C5                 PUSH    B       ;SAVE CONTENTS OF C
0AA9 D383       DLY1:   OUT     CR3     ;START TIMER
0AAB DB81       DLY2:   IN      STATUS  ;WAIT TILL IT TIMES OUT
0AAD E620               ANI     TMR     ;LOOK AT TIMER BIT
0AAF CAAB0A             JZ      DLY2    ;LOOP BACK IF ZERO
0AB2 0D                 DCR     C       ;COUNT DOWN
0AB3 C2A90A             JNZ     DLY1    ;DO AGAIN IF NOT ZERO
0AB6 C1                 POP     B
0AB7 C9                 RET


                ;STARTS THE 50 MS TIMER
```

```
0AB8 D383       STIME:  OUT     CR3     ;START TIMER
0ABA C9                 RET

                ;CHECKS THE STATUS OF THE TIMER.
                ; RETURNS WITH  A=1 IF TIMED OUT OR  A=0 IF NOT.

0ABB DB81       CKTIME: IN      STATUS
0ABD E620               ANI     TMR     ;MASK IN TIMER BIT
0ABF 3E01               MVI     A,1
0AC1 C0                 RNZ             ;RET.  A=1 IF TIMED OUT
0AC2 3D                 DCR     A
0AC3 C9                 RET             ;RET.  A=0 IF TIME NOT UP

                ;GOES OFF-HOOK AND WAITS FOR A DIAL TONE.  SINCE THE
                ;MICROMODEM HAS NO DIAL TONE DETECTOR WE JUST GO
                ;OFF-HOOK, WAIT 2 SECONDS, ASSUME A DIAL TONE IS
                ;PRESENT, THEN RETURN.

0AC4 C5         DLTONE: PUSH    B
0AC5 0E01               MVI     C,1
0AC7 CD290A             CALL    SWHOOK  ;GO OFF-HOOK
0ACA 0E28               MVI     C,40    ;40*50MS=2 SECONDS
0ACC CDA50A             CALL    DELAY   ;WAIT 2 SECONDS
0ACF C1                 POP     B       ;RESTORE BC
0AD0 C9                 RET

                ;DIAL PULSE ROUTINE. DIALS THE DIGIT IN REGISTER C.
                ;IF C=0 THEN 10 PULSES WILL BE PRODUCED. THE DIGIT IN
                ;THE C REGISTER CAN BE BINARY OR ASCII.  AFTER THE
                ;DIGIT IS OUT-PULSED THIS ROUTINE WILL PAUSE FOR 600 MS
                ;FOR INTER-DIGIT SPACING.

0AD1 C5         PULSE:  PUSH    B       ;SAVE BC
0AD2 79                 MOV     A,C
0AD3 060A               MVI     B,10    ;PRE-LOAD PULSE COUNTER
0AD5 E60F               ANI     0FH     ;KILL POSSIBLE ASCII BIAS
0AD7 FE00               CPI     0       ;SEE IF WE'RE DOING 10 PULSES
0AD9 CADD0A             JZ      PULS1
0ADC 47                 MOV     B,A     ;PULSE COUNT TO B
0ADD 0E00       PULS1:  MVI     C,0     ;GO ON HOOK
0ADF CD290A             CALL    SWHOOK
0AE2 0E01               MVI     C,1
0AE4 CDA50A             CALL    DELAY   ;WAIT 50 MS
0AE7 0E01               MVI     C,1
0AE9 CD290A             CALL    SWHOOK  ;OFF HOOK AGAIN
0AEC 0E01               MVI     C,1
0AEE CDA50A             CALL    DELAY   ;WAIT 50 MS
0AF1 05                 DCR     B       ;WE MADE 1 PULSE, COUNT IT
0AF2 C2DD0A             JNZ     PULS1   ;IF NOT ZERO, MAKE SOME MORE
```

```
0AF5 0E0B               MVI     C,11    ;11*50 MS=550 MS
0AF7 CDA50A             CALL    DELAY   ;INTER-DIGIT DELAY
0AFA C1                 POP     B       ;RESTORE BC REGISTERS
0AFB C9                 RET


                ;SETS BITS IN CONTROL REGISTER 2.
                ;ENTER WITH REG. A CONTAINING 1'S WHERE BITS ARE TO BE
                ;SET.

0AFC E5         SET2:   PUSH    H       ;SAVE HL
0AFD 210110             LXI     H,MICR2
0B00 B6                 ORA     M       ;OR IN BITS TO BE SET
0B01 77                 MOV     M,A
0B02 D382               OUT     CR2     ;SET THE MODEM
0B04 E1                 POP     H
0B05 C9                 RET

                ;CLEAR BITS IN CONTROL REGISTER 2.
                ;REG. A HAS 1'S FOR BITS TO BE CLEARED.

0B06 E5         CLR2:   PUSH    H
0B07 210110             LXI     H,MICR2 ;MEMORY IMAGE OF CR2
0B0A 2F                 CMA
0B0B A6                 ANA     M       ;TURN OFF THE BIT
0B0C 77                 MOV     M,A     ;UPDATE MEMORY
0B0D D382               OUT     CR2     ;WRITE TO CRL. REG. 2
0B0F E1                 POP     H       ;RESTORE HL
0B10 C9                 RET

                ;SET BITS IN CONTROL REGISTER 1.
                ;REG. A HAS 1'S IN BITS TO BE SET.

0B11 E5         SET1:   PUSH    H
0B12 210010             LXI     H,MICR1 ;POINT TO MEMORY IMAGE
0B15 B6                 ORA     M       ;OR IN THE BITS
0B16 77                 MOV     M,A     ;UPDATE MEMORY
0B17 D381               OUT     CR1     ;WRITE TO MODEM
0B19 E1                 POP     H
0B1A C9                 RET

                ;CLEAR BITS IN CONTROL REGISTER 1
                ;REG. A HAS 1'S IN BITS TO BE CLEARED.

0B1B E5         CLR1:   PUSH    H
0B1C 210010             LXI     H,MICR1 ;POINT TO MEMORY IMAGE
0B1F 2F                 CMA             ;INVERT A
0B20 A6                 ANA     M       ;CLEAR BITS
0B21 77                 MOV     M,A     ;UPDATE MEMORY
0B22 D381               OUT     CR1
0B24 E1                 POP     H
0B25 C9                 RET
```

```
                ;THIS AREA MUST BE R/W MEMORY.

1000                    ORG     1000H           ;RAM MEMORY AREA
                SOR:                            ;START OF RAM

1000            MICR1:  DS      1       ;MEMORY IMAGE OF CONTROL REG. 1
1001            MICR2:  DS      1       ;MEMORY IMAGE OF CONTROL REG. 2
1002            DPLX:   DS      1       ;FULL DUPLEX FLAG 1=FULL  0=HALF
1003            MFLAG:  DS      1       ;0=ANSWER    1=ORIGINATE

1004            JTAB:   DS      3       ;CBIOS JUMP TABLE IS MOVED HERE
1007            XCONST: DS      3       ;CONSOLE STATUS
100A            XCONIN: DS      3       ;CONSOLE INPUT
100D            XCONOT: DS      3       ;CONSOLE OUTPUT

1010            NMBR:   DS      32      ;32 BYTES PHONE NUMBER STORAGE
1030            BUFPNT: DS      2       ;BUFFER POINTER STORAGE

1032            CPFLAG: DS      1               ;CAPTURE FLAG 1=ON  0=OFF
1033            BUGFLAG:DS      1               ;1=DEBUG MODE  0= NORMAL
1034            BUGCOUNT: DS    1               ;COUNTS HEX BYTES PER LINE
1035            MTOP:   DS      1               ;TOP OF MEMORY (PAGE)
1036            CBUFP:  DS      2               ;CAPTURE BUFFER POINTER
1038            CPCNT:  DS      2               ;CAPTURED CHARACTER COUNTER
                EOR:                            ;END OF RAM CHECKED BY CHECKSUM
103A            CKSUM:  DS      1               ;CHECKSUM STORED HERE

103B                    DS      64
107B            STACK:  DS      1       ;64 BYTES FOR STACK

107C            BUF:    DS      65      ;CONSOLE BUFFER

                ;CPM FILE CONTROL BLOCK

                FCB:
10BD            ET:     DS      1               ;ENTRY TYPE
10BE            FNAME:  DS      8               ;FILE NAME
10C6            FTYPE:  DS      3               ;FILE TYPE
10C9            EX:     DS      3               ;EXTENT
10CC            RCC:    DS      1               ;RECORD COUNT
10CD            DM:     DS      16              ;DISK MAP
10DD            NR:     DS      1               ;NEXT RECORD

10DE            CPBUF:  DS      1               ;CAPTURE BUFFER

10DF                    END     START   ;THATS ALL
```

```
TITLE   'REMOTE CONSOLE FOR CP/M AND THE MICROMODEM 100'

;WRITTEN BY DALE HEATHERINGTON

;COPYRIGHT 1979, D.C. HAYES ASSOCIATES, INC.


;THIS PROGRAM ALLOWS THE USER TO IMPLEMENT THE
;MICROMODEM 100 AS AN ADDITIONAL CONSOLE FOR CP/M.
;IT WILL ONLY WORK WITH VERSION 1.4 BECAUSE IT
;DEPENDS ON THE SPECIAL FEATURE WHICH CHECKS THE
;CONSOLE BUFFER FOR A COMMAND JUST AFTER WARM BOOT.
;THE NAME OF THIS PROGRAM MUST BE IN THIS BUFFER SO
;IT WILL EXECUTE EACH TIME CP/M BOOTS UP.  THE "INSTALL"
;PROGRAM TAKES CARE OF PUTTING THE NAME IN THE BUFFER.
;
;      ------ OPERATION -------
;
;WHEN CP/M BOOTS IT LOOKS IN ITS CONSOLE BUFFER.  IT
;FINDS "REMOTE" AND LOADS IT.  THIS PROGRAM THEN USES
;THE ADDRESS AT LOCATION 6 TO FIND OUT WHERE THE BOTTOM
;OF THE CCP (CONSOLE COMMAND PROCESSOR)  IS.  IT THEN
;RELOCATES ITSELF 512 BYTES BELOW THE CCP AND CHANGES
;THE ADDRESS AT LOCATION 6 TO INDICATE A NEW TOP OF USER
;MEMORY.  IT THEN GETS THE FIRST 5 JUMPS FROM THE CBIOS
;JUMP TABLE AND MOVES THEM INTO ITS OWN MEMORY SPACE.
;THESE JUMPS ARE ALL THE CONSOLE FUNCTIONS.  IT THEN
;PUTS JUMPS TO ITS OWN MICROMODEM 100 CONSOLE ROUTINES
;IN THE CBIOS JUMP TABLE.  IT THEN RETURNS TO CP/M
;WHICH PUTS UP THE PROMPT A>  .

;THE NEW CONSOLE STATUS ROUTINE CHECKS FOR PHONE RINGING
;AND CARRIER DETECT AS WELL AS MODEM CHARACTER RECEIVED.
;IT THEN USES THE OLD CBIOS CONSOLE STATUS VECTOR TO CONTINUE
;ON TO THE CBIOS ROUTINE.

;THE USER OF CP/M AT THE LOCAL CONSOLE WILL NOTICE NO
;DIFFERENCE IN THE OPERATION OF CP/M EXCEPT THAT LESS
;MEMORY WILL BE AVALABLE FOR TRANSIENT PROGRAMS.  ALSO
;"STAT" WILL ALWAYS RETURN  0 K BYTES LEFT FOR SOME
;UNKNOWN REASON.  WHEN THE PHONE RINGS IT WILL ANSWER
;AND PUT A CARRIER ON THE LINE.  THEN IT WAITS 15 SECONDS
;FOR THE CALLER TO SEND A CARRIER.  IT HANGS UP IF NO
;CARRIER IS DETECTED.  WHEN IT DETECTS A CARRIER IT
;REBOOTS CP/M SO THE CALLER WILL GET THE PROMPT  A> .
;
;IF YOU WANT TO MAKE MODIFICATIONS TO THIS PROGRAM
;MAKE SURE YOU MARK ALL ADDRESS WHICH WILL BE RE-LOCATED
;WITH SOME LABEL AND PUT THAT LABEL IN THE RELOCATION TABLE.
;I USED "RL" FOLLOWED BY A DIGIT FOR LABELS WHOSE SOLE
;PURPOSE WAS TO MARK RELOCATED ADDRESSES.

        PAGE
```

```
0005 =          BDOS    EQU     5       ;CP/M DOS ENTRY

0080 =          DATA    EQU     80H     ;MODEM DATA PORT
0081 =          CCR1    EQU     DATA+1  ;CONTROL AND STAT. REGISTER
0082 =          CCR2    EQU     DATA+2  ;CONTROL REGISTER #2
0083 =          TIMER   EQU     DATA+3  ;STARTS TIMER

                ;BIT FUNCTIONS

0001 =          EPE     EQU     1       ;EVEN PARITY
0002 =          LS1     EQU     2       ;LENGTH SELECT
0004 =          LS2     EQU     4       ;LENGTH SELECT 2
0008 =          SBS     EQU     8       ;STOP BITS
0010 =          PI      EQU     10H     ;PARITY INHIBIT

0001 =          RRF     EQU     1       ;CHARACTER RECEIVED
0002 =          TRE     EQU     2       ;TRANSMITTER EMPTY
0004 =          PE      EQU     4       ;PARITY ERROR
0008 =          FE      EQU     8       ;FRAMING ERROR
0010 =          OE      EQU     10H     ;OVERRUN ERROR
0020 =          TMR     EQU     20H     ;TIMER STATUS 1=TIMED OUT
0040 =          CD      EQU     40H     ;CARRIER DETECT
0080 =          RI      EQU     80H     ;NOT RING DETECT

0001 =          BRS     EQU     1       ;BAUD RATE (300)
0002 =          TXE     EQU     2       ;TRANSMITTER ON
0004 =          MS      EQU     4       ;MODE SELECT (ORIG)
0080 =          OH      EQU     80H     ;OFF HOOK



0100            ORG     100H

0100 C30901             JMP     START
0103 000000             DB      0,0,0
0106 C30000     XBDOS:  DB      0C3H,0,0        ;BDOS VECTOR PLACED HERE
0109 CD7102     START:  CALL    SIGNON
010C DB81               IN      CCR1
010E E640               ANI     CD              ;LOOK FOR CARRIER DETECT
0110 C4D801             CNZ     ANSBAK          ;SIGNON MESSAGE TO MODEM IF CD
0113 2A0600             LHLD    BDOS+1          ;GET BASE OF BDOS
0116 1100F6             LXI     D,-0A00H        ;LENGTH OF CCP+512
0119 19                 DAD     D
011A EB                 XCHG                    ;BASE OF CCP -512 TO DE
011B 2A0600             LHLD    BDOS+1
011E E5                 PUSH    H
011F EB                 XCHG
0120 220600             SHLD    BDOS+1          ;NEW JUMP VECTOR FOR BDOS
```

```
0123 E1              POP     H               ;RECOVER OLD VECTOR
0124 220701          SHLD    XBDOS+1

                ;ADD THE RELOCATION CONSTANT TO ALL ADDRESSES IN TABLE

0127 2A0600          LHLD    6               ;GET NEW BDOS ENTRY
012A 11FAFF          LXI     D,-6
012D 19              DAD     D               ;SUBTRACT 6
012E 1100FF          LXI     D,-256
0131 19              DAD     D
0132 223702          SHLD    OFFSET          ;STORE IT
0135 213D02          LXI     H,TABLE
0138 223902          SHLD    TBLPNT          ;STORE TABLE POINTER
013B 2A3902    ADD1: LHLD    TBLPNT          ;GET POINTER
013E 5E              MOV     E,M             ;GET LOW BYTE
013F 23              INX     H
0140 56              MOV     D,M             ;GET HIGH BYTE
0141 7B              MOV     A,E
0142 B2              ORA     D               ;TEST FOR 0 (END OF TABLE)
0143 CA5C01          JZ      MOVIT
0146 23              INX     H               ;INC. POINTER
0147 223902          SHLD    TBLPNT          ;SAVE TABLE POINTER
014A EB              XCHG                    ;ADDRESS TO MODIFY TO HL
014B 23              INX     H               ;POINT TO ADDRESS FIELD
014C 5E              MOV     E,M             ;GET ADDRESS
014D 23              INX     H
014E 56              MOV     D,M
014F E5              PUSH    H               ;SAVE POINTER
0150 2A3702          LHLD    OFFSET          ;GET OFFSET
0153 19              DAD     D               ;ADD IT TO ADDRESS
0154 EB              XCHG                    ;MODIFIED ADR. BACK TO DE
0155 E1              POP     H               ;RECOVER POINTER
0156 72              MOV     M,D             ;PUT ADDRESS BACK IN MEM.
0157 2B              DCX     H
0158 73              MOV     M,E
0159 C33B01          JMP     ADD1            ;LOOP

                ;MOVE THIS PROGRAM UP UNDER THE CCP

015C 010002    MOVIT: LXI    B,512
015F 2A3702          LHLD    OFFSET
0162 110001          LXI     D,256
0165 19              DAD     D
0166 223B02          SHLD    SADDR   ;RELOCATED STARTING ADDRESS
0169 1A        MV1:  LDAX    D
016A 77              MOV     M,A
016B 23              INX     H
016C 13              INX     D
016D 0B              DCX     B
016E 79              MOV     A,C
016F B0              ORA     B
0170 C26901          JNZ     MV1     ;MOVE 512 BYTES

0173 C37601    RL1:  JMP     $+3     ;START EXECUTING RELOCATED CODE

                ;EXCHANGE THE CBIOS JUMP TABLE WITH OUR SPECIAL TABLE.
```

```
0176 2A0100          LHLD   1        ;GET CBIOS ADDRESS
0179 2B              DCX    H
017A 2B              DCX    H
017B 2B              DCX    H
017C 112802   RL2:     LXI    D,XBOOT
017F 060F            MVI    B,15     ;15 BYTES OF JUMP TABLE TO MOVE
0181 4E       MV3:   MOV    C,M      ;GET A CBIOS BYTE
0182 1A              LDAX   D        ;GET OUR BYTE
0183 77              MOV    M,A      ;PUT OUR BYTE IN CBIOS
0184 79              MOV    A,C
0185 12              STAX   D        ;PUT CBIOS BYTE IN OUR TABLE
0186 13              INX    D
0187 23              INX    H
0188 05              DCR    B
0189 C28101   RL3:     JNZ    MV3      ;LOOP 15 TIMES
018C C9              RET             ;BACK TO CPM

018D CD7301   BOOT:    CALL   RL1      ;EXCHANGE JUMP TABLES
0190 2A0100          LHLD   1
0193 2B              DCX    H
0194 2B              DCX    H
0195 2B              DCX    H
0196 E9              PCHL   ;DO CPM BOOT

0197 CD7301   WBOOT:   CALL   RL1
019A C30000          JMP    0



              ;THIS IS THE CONSOLE STATUS CHECK ROUTINE FOR THE
S             ;MODEM.  IT ALSO CHECKS FOR CARRIER DETECT AND RINGING.
              ;WHEN FINISHED IT GOES TO THE ORIGINAL CBIOS ROUTINE.

019D DB81     CONST:   IN     CCR1            ;CHECK FOR CHAR. IN MODEM
019F E601              ANI    RRF
01A1 3EFF              MVI    A,255
01A3 C0                RNZ                    ;RET. 255 IF TRUE
01A4 DB81              IN     CCR1            ;CHECK FOR CARRIER
01A6 E640              ANI    CD
01A8 C2AF01   RL5:     JNZ    CK1
01AB 3E00              MVI    A,0
01AD D382              OUT    CCR2            ;NO CARRIER, HANG UP
01AF DB81     CK1:     IN     CCR1            ;TEST FOR RING
01B1 E680              ANI    RI
01B3 C22E02   RL0:     JNZ    XCONST          ;IF NO RING GO TO LOCAL CONSOLE
01B6 3E83              MVI    A,OH+TXE+BRS    ;ANSWER THE PHONE
01B8 D382              OUT    CCR2
01BA 3E16              MVI    A,PI+LS1+LS2
01BC D381              OUT    CCR1            ;8 DATA, 1 STOP, NO PARITY
01BE DB80              IN     DATA            ;CLEAR OUT UART
01C0 DB80              IN     DATA
01C2 0696              MVI    B,150           ;WAIT 15 SECONDS FOR CARRIER
01C4 CDE701   RL7:     CALL   WAIT100         ;100 MS WAIT
01C7 DB81              IN     CCR1
01C9 E640              ANI    CD              ;LOOK FOR CARRIER
```

```
01CB C20000             JNZ     0               ;WARM BOOT CP/M
01CE 05                 DCR     B
01CF C2C401     RL9:    JNZ     RL7
01D2 AF                 XRA     A
01D3 D382               OUT     CCR2            ;NO CARRIER, HANG UP
01D5 C32E02     RL10:   JMP     XCONST

                ;THIS ROUTINE SENDS THE SIGN ON MESSAGE TO THE MODEM.
                ;IT IS ONLY CALLED ONCE BEFORE IT IS MOVED AND NEVER
                ;AGAIN.  ITS ADDRESSES ARE NOT RELOCATED.

01D8 217A02     ANSBAK: LXI     H,MSG           ;POINT TO MESSAGE
01DB 7E         ANSB1:  MOV     A,M             ;GET CHAR.
01DC FE24               CPI     '$'             ;END?
01DE C8                 RZ                      ;QUIT WHEN DONE
01DF 4F                 MOV     C,A
01E0 CD1D02             CALL    CO1             ;SEND TO MODEM
01E3 23                 INX     H
01E4 C3DB01             JMP     ANSB1

                ;WAIT  100 MS BEFORE RETURNING

01E7 CDEA01     WAIT100: CALL   WAIT50
01EA D383       WAIT50: OUT     TIMER   ;START 50 MS TIMER
01EC F5                 PUSH    PSW
01ED DB81       WA1:    IN      CCR1
01EF E620               ANI     TMR     ;IS TIME UP?
01F1 CAED01     RL11:   JZ      WA1     ;LOOP IF NOT
01F4 F1                 POP     PSW
01F5 C9                 RET


01F6 CD9D01     CONIN:  CALL    CONST   ;SEE IF ANY DATA READY
01F9 B7                 ORA     A
01FA CAF601     RL14:   JZ      CONIN   ;NO, WAIT UNTIL IT IS
01FD DB81               IN      CCR1
01FF E601               ANI     RRF     ;FROM MODEM?
0201 CA3102     RL15:   JZ      XCONIN  ;NO,MUST BE LOCAL CONSOLE
0204 DB80               IN      DATA
0206 C9                 RET
```

```
                ;THIS IS THE CONSOLE OUTPUT ROUTINE.  FOR THE
                ;USERS WHO MAY CALL IN WITH PRINTERS IT WILL
                ;DELAY 100 MS AFTER ALL LINE FEEDS.
```

```
0207 79        CONOUT: MOV     A,C
0208 2F                CMA
0209 D3FF              OUT     255
020B DB81              IN      CCR1            ;LOOK FOR CARRIER
020D E640              ANI     CD
020F CA3402    RL12:   JZ      XCONOUT         ;BYPASS IF NO CARRIER
0212 CD1D02    RL16:   CALL    CO1             ;OUTPUT THE CHARACTER
0215 FE0A              CPI     0AH             ;WAS IT A LINE FEED
0217 CCE701    RL17:   CZ      WAIT100         ;WAIT 100 MS IF LINE FEED
021A C33402    RL6:    JMP     XCONOUT         ;OUT TO LOCAL CONSOLE
021D DB81      CO1:    IN      CCR1
021F E602              ANI     TRE
0221 CA1D02    RL13:   JZ      CO1             ;LOOP UNTIL TX EMPTY
0224 79                MOV     A,C             ;GET CHARACTER
0225 D380              OUT     DATA            ;TO MODEM
0227 C9                RET


                ;THESE 5 JUMPS ARE EXCHANGED WITH THE FIRST 5 JUMPS
                ;IN THE CBIOS JUMP TABLE.


0228 C38D01    XBOOT:  JMP     BOOT
022B C39701    XWBOOT: JMP     WBOOT
022E C39D01    XCONST: JMP     CONST
0231 C3F601    XCONIN: JMP     CONIN
0234 C30702    XCONOUT:JMP     CONOUT


                ;SOME STORAGE LOCATIONS

0237           OFFSET: DS      2
0239           TBLPNT: DS      2
023B           SADDR:  DS      2

                ;TABLE OF ADDRESSES TO ADD RELOCATION CONSTANT TO.

023D B3017301  TABLE:  DW      RL0,RL1
0241 7C018901A8        DW      RL2,RL3,RL5,RL6,RL7,RL9,RL10
024F F1010F0221        DW      RL11,RL12,RL13
0255 FA010102          DW      RL14,RL15
0259 12021702          DW      RL16,RL17
025D 8D019701E7        DW      BOOT,WBOOT,WAIT100
0263 F601              DW      CONIN
0265 28022B022E        DW      XBOOT,XWBOOT,XCONST
026B 31023402          DW      XCONIN,XCONOUT
026F 0000              DW      0

                ;PRINTS SIGN ON MESSAGE.  THIS CODE IS NOT RELOCATED
```

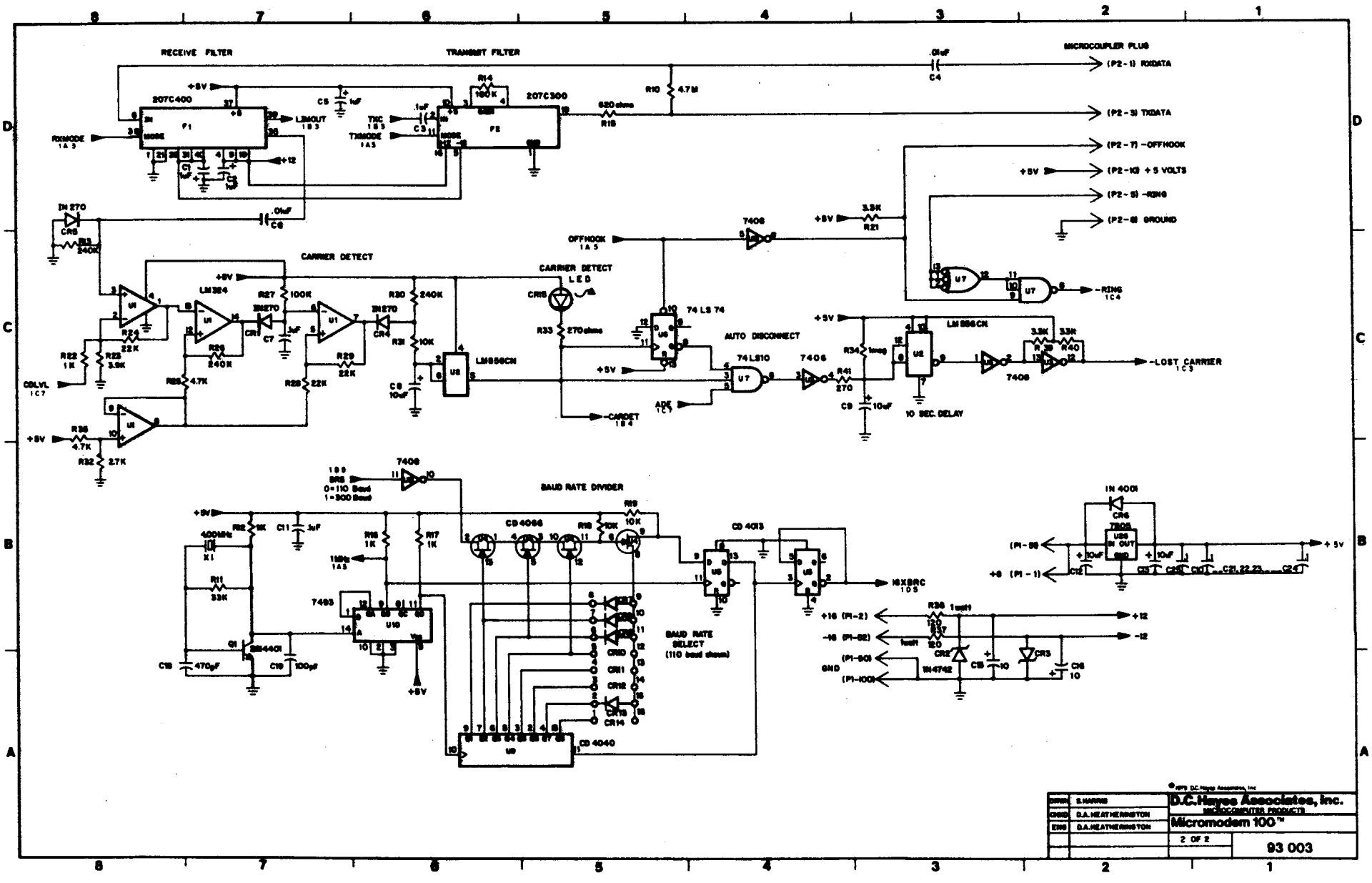
```
0271 117A02    SIGNON: LXI     D,MSG   ;POINT TO MESSAGE
0274 0E09              MVI     C,9     ;PRINT BUFFER CODE
0276 CD0500            CALL    5       ;CALL BDOS
0279 C9                RET

027A 442E432E20MSG:    DB      'D.C. HAYES ASSOCIATES, INC.',0DH,0AH
0297 4D4943524F        DB      'MICROMODEM 100 REMOTE CONSOLE VER. 1.0',0DH,0AH,'$'

02C0                   END 100H        ;GOOD LUCK
```

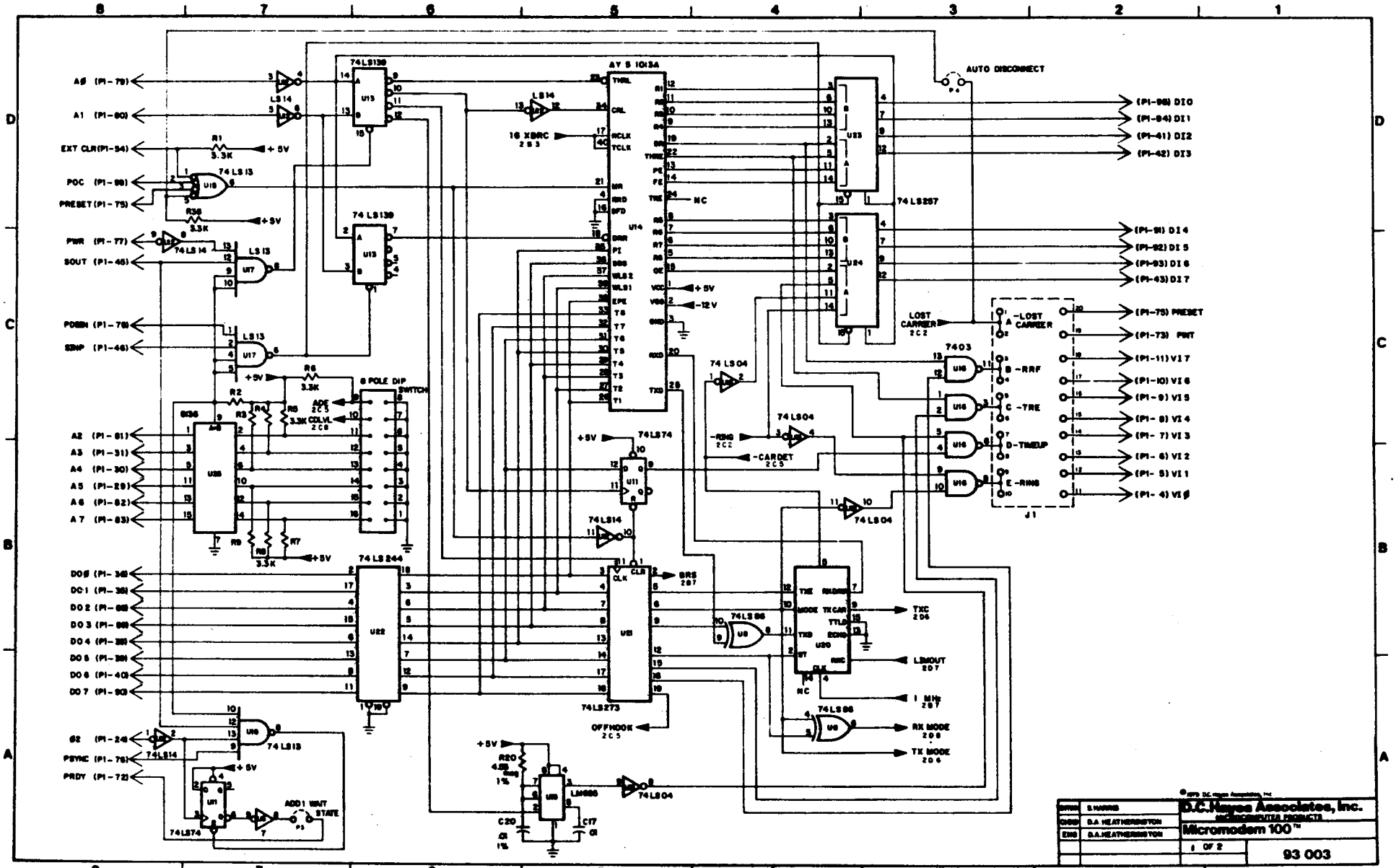
```
                TITLE   'INSTALL OR REMOVE REMOTE CONSOLE'

                ;THIS PROGRAM WILL INSTALL OR REMOVE
                ;REMOTE CONSOLE IN 1.4 CP/M.
                ;NAME:  INSTALL.ASM


                ;WRITTEN BY DALE HEATHERINGTON
                ;
                ;COPYRIGHT 1979, D.C. HAYES ASSOCIATES, INC.

                ;THIS PROGRAM ONLY WORKS WITH VERSION 1.4 CP/M.
                ;IT LOADS THE FIRST 128 BYTES OF THE CP/M CCP MODULE
                ;INTO RAM AT 80 HEX.  THIS IS ASSUMED TO BE ON TRACK ZERO,
                ;SECTOR 2.  IF THE USER TYPES "I" TO INSTALL REMOTE CONSOLE
                ;THE PROGRAM WILL INSERT "REMOTE" INTO THE CCP CONSOLE BUFFER
                ;ALONG WITH IT LENGTH, 6.  IT THEN WRITES THIS MODIFIED SECTOR
                ;OF THE CCP BACK TO THE DISK.  "REMOVE" WORKS THE SAME
                ;EXCEPT THAT IT SETS THE LENGTH TO 0 AND WRITES 6 SPACES TO
                ;THE CONSOLE BUFFER.
                ;       WHEN CP/M WARM BOOTS IT CHECKS THE CONSOLE BUFFER TO SEE
                ;IF IT HAS ANY THING IN IT.  IF IT DOES CP/M READS IT AND
                ;PERFORMS THE FUNCTION.  IN THIS CASE IT WILL LOAD AND EXECUTE
                ;THE FILE "REMOTE.COM",  THE REMOTE CONSOLE PROGRAM.


0100                    ORG 100H
0005 =          BDOS    EQU     5
0082 =          CR2     EQU     82H     ;MICROMODEM 100 CONTROL REGISTER 2

000D =          CR      EQU     0DH     ;ASCII CARRAGE RETURN


0100 C33001             JMP     START


                ;THE CBIOS JUMP TABLE IS MOVED HERE


0103            BOOT:   DS      3
0106            WBOOT:  DS      3
0109            CONST:  DS      3
010C            CONIN:  DS      3
010F            CONOUT: DS      3
0112            LIST:   DS      3
0115            PUNCH:  DS      3
0118            READER: DS      3
011B            HOME:   DS      3
011E            SELDSK: DS      3
0121            SETTRK: DS      3
0124            SETSEC: DS      3
0127            SETDMA: DS      3
012A            READ:   DS      3
012D            WRITE:  DS      3

0130 314F03     START:  LXI     SP,STACK
```

```
0133 2A0100            LHLD    1       ;GET WARM BOOT ADDRESS
0136 2B                DCX     H
0137 2B                DCX     H
0138 2B                DCX     H
0139 110301            LXI     D,BOOT  ;DESTINATION ADDRESS
013C 062D              MVI     B,15*3  ;NUMBER OF BYTES TO MOVE
013E CDCD01            CALL    MOVE    ;MOVE THE CBIOS JUMP TABLE
0141 018000            LXI     B,80H
0144 CD2701            CALL    SETDMA  ;SET DMA ADDRESS TO 80H
0147 0E00              MVI     C,0
0149 CD2101            CALL    SETTRK  ;SET TRACK TO 0
014C 0E02              MVI     C,2
014E CD2401            CALL    SETSEC  ;SET SECTOR TO 2
0151 CD2A01            CALL    READ    ;READ IN FIRST SECTOR OF CPM
0154 CD9D01    QQ:     CALL    PRINT
0157 494E535441        DB      'INSTALL OR REMOVE ? (I OR R)',CR
0174 CD8401            CALL    INPUT   ;GET REPLY
0177 FE49              CPI     'I'
0179 CAD601            JZ      INSTALL
017C FE52              CPI     'R'
017E CA0B02            JZ      REMOVE
0181 C35401            JMP     QQ

               ;CONSOLE INPUT ROUTINE

0184 E5        INPUT:  PUSH    H
0185 D5                PUSH    D
0186 C5                PUSH    B
0187 0E01              MVI     C,1
0189 CD0500            CALL    5
018C 7B                MOV     A,E
018D FE60              CPI     60H
018F DA9401            JC      IN1
0192 D620              SUI     20H     ;CONVERT TO UPPER CASE
0194 C1        IN1:    POP     B
0195 D1                POP     D
0196 E1                POP     H
0197 C9                RET

0198 CD9D01    CRLF:   CALL    PRINT
019B 0D                DB      CR
019C C9                RET

               ;PRINTS ASCII STRINGS POINTED TO BY TOP OF STACK
               ; TO CONSOLE.

019D E3        PRINT:  XTHL            ;GET STRING POINTER
019E F5                PUSH    PSW
019F C5                PUSH    B
01A0 7E        PO1:    MOV     A,M
01A1 23                INX     H
01A2 FE40              CPI     '@'     ;NO CR IF @ IS END OF STRING
01A4 CABA01            JZ      NOCR
01A7 CDBE01            CALL    COUT
```

```
01AA FE0D              CPI     CR      ;CARRAGE RET?
01AC CAB201            JZ      THEEND
01AF C3A001            JMP     P01
01B2 CD9D01    THEEND: CALL    PRINT
01B5 0A00000040        DB      0AH,0,0,0,'@'
01BA C1        NOCR:   POP     B
01BB F1                POP     PSW
01BC E3                XTHL
01BD C9                RET

               ;CONSOLE OUTPUT ROUTINE

01BE F5        COUT:   PUSH    PSW
01BF C5                PUSH    B
01C0 D5                PUSH    D
01C1 E5                PUSH    H
01C2 5F                MOV     E,A
01C3 0E02              MVI     C,2     ;WRITE CONSOLE
01C5 CD0500            CALL    5
01C8 E1                POP     H
01C9 D1                POP     D
01CA C1                POP     B
01CB F1                POP     PSW
01CC C9                RET

               ;GENERAL PURPOSE MEMORY TO MEMORY BLOCK MOVE ROUTINE
               ; HL POINT TO THE SOURCE,  DE POINT TO THE DESTINATION
               ;  THE B REGISTER HAS THE COUNT.

01CD 7E        MOVE:   MOV     A,M
01CE 12                STAX    D
01CF 23                INX     H
01D0 13                INX     D
01D1 05                DCR     B
01D2 C2CD01            JNZ     MOVE
01D5 C9                RET

               ;THIS ROUTINE PUTS "REMOTE" INTO THE CCP CONSOLE BUFFER

               INSTALL:
01D6 214102            LXI     H,INSTL
01D9 118700            LXI     D,87H
01DC 0607              MVI     B,7
01DE CDCD01            CALL    MOVE
01E1 CD2D01            CALL    WRITE   ;WRITE SECTOR TO DISK
01E4 CD9801            CALL    CRLF
01E7 CD9D01            CALL    PRINT
01EA 43502F4D20        DB      'CP/M REMOTE CONSOLE INSTALLED',CR
0208 C30000            JMP     0

               ;THIS ROUTINE CLEARS THE CCP CONSOLE BUFFER
```

```
020B 214802     REMOVE: LXI     H,REMV
020E 118700             LXI     D,87H
0211 0607               MVI     B,7
0213 CDCD01             CALL    MOVE
0216 CD2D01             CALL    WRITE
0219 CD9801             CALL    CRLF
021C CD9D01             CALL    PRINT
021F 43502F4D20         DB      'CP/M REMOTE CONSOLE REMOVED',CR
023B AF                 XRA     A               ;ZERO A
023C D382               OUT     CR2             ;MAKE SURE MODEM PHONE LINE IS HUNG UP
023E C30000             JMP     0

                ;THESE ARE THE STRINGS WHICH ARE PUT IN THE
                ;CCP CONSOLE BUFFER

0241 0652454D4FINSTL:   DB      6,'REMOTE'

0248 0020202020REMV:    DB      0,'      '



024F                    DS      256
                STACK:


034F                    END     100H
```


RECEIVE FILTER
TRANSMIT FILTER
MICRODOUPLER PLUG
207C400
207C300
(P2-1) RXDATA
(P2-3) TXDATA
(P2-7) -OFFHOOK
+5V
(P2-10) +5 VOLTS
(P2-9) -RING
(P2-8) GROUND
CARRIER DETECT
CARRIER DETECT LED
LM324
LM556CN
74LS74
74LS10
7406
AUTO DISCONNECT
10 SEC. DELAY
-LOST CARRIER
-CARDET
OFFHOOK
ADE
BAUD RATE DIVIDER
CD4060
CD4013
CD4040
7493
BAUD RATE SELECT
(110 baud shown)
IN4001
IN4742
TBOS
RXSRC
+16 (P1-2)
-16 (P1-20)
GND
(P1-100)
(P1-99)
(P1-1)
D.C.Hayes Associates, Inc.
MICROCOMPUTER PRODUCTS
Micromodem 100™
2 OF 2
93 003



DRWN	S.HANNS	D.C.Hayes Associates, Inc. MICROCOMPUTER PRODUCTS
CHKD	D.A.HEATHERINGTON	Micromodem 100™
ENG	D.A.HEATHERINGTON	1 OF 2 — 93 003