PACKET RADIO - THE 3RD GENERATION SOFTWARE APPROACH

AX. 25 PROTOCOL

Robert M. Richardson, W4UCH
22 North Lake Drive
Chautauqua Lake, N.Y. 14722

ABSTRACT:

The 3rd generation 'software approach@ to 1200 baud packet radio using the AX.25 protocol is described. This approach consists of software written in assembly language to replace the Tucson Amateur Packet Radio (TAPR) terminal node controller (TNC) which includes:
- the TNC's 68093 microprocessor.
- the TNC's costly SDLC/HDLC controller.
- the TNC's large 25K to 35K EPROM.
- the TNC's dynamic RAM.
- the TNC's RS232 UART
- the TNC's ancillary support chips.

The software approach also eliminates the need for an RS232 interface (approx. $100 cost) on the host microcomputer which may be either a Model I or Model III TRS-80. The RS232 interface is replaced by a low cost port zero encoder/decoder (parts cost approx. $15) which is used to interface the microcomputer to a home brew modem (parts cost approx. $15) which may use the low cost EXAR 2206/2211 AFSK modulator and demodulator chips that are used in both the Vancouver and TAPR modems.

A more sophisticated modem of the users choice for noise-prone and fade-prone circuits such as OSCAR 10 may be required for satisfactory weak-signal operation, though the author regularly and reliably is able to work the Toronto, Canada area packet repeater, VE3PKT, some 110+ miles distant.

A number of major improvements for the 3rd generation packet radio software approach which are included in Volume 2, 'Synchronous Packet Radio Using The Software Approach - AX.25 Protocol,' are described in detail.

INTRODUCTION:

Just as the TAPR terminal node controller has undergone a number of iterations and improvements since its inception, the 'software approach' has followed a similar course. Looking at a typical exponential growth/learning curve, 1984's software approach is about 85% up the vertical scale and approaching the knee of the curve whereas the software packet program written in 1982 was at the 33% level. This decided improvement was largely motivated by the disclosure in 1983 to the public at large, of the brilliant AX.25 protocol by Terry Fox, WB4JFI et al at the Second ARRL Amateur Radio Computer Networking Conference. The AX.25 protocol is to packet, what SSB was to amateur radio communication techniques in the mid-1950's; i.e., not revolutionary, but a giant evolutionary step forward. We doff our collective hats to the many authors of the excellllent AX. 25 protocol.

The 3rd generation software approach has a significant number of improvements over the 1st generation that was presented in Volume 1, 'Synchronous Packet Using The Software Approach - Vancouver Protocol.' These improvements are:

1. Receive mode synchronous to parallel byte conversion is done in real-time.
2. Automatic; AX.25 repeater if your call+ SSID in repeater segment address field
3. CRC generation and checking is done in virtual real-time = 27 times FASTER.
4, AUTO connect mode option for unattended operation with T2 timer auto reset.
5. FORMAT on/off option for receive video recognizes or ignores C/R's and L/F's.
6. Multi-frame packets are input from the keyboard same as single frame packets.
7. Information field length may be set from the menu from 1 to 256 bytes.
8. Frames per packet may be set from the menu from 1 to 7 frames.
9. NOW connected mode displays and stores only the information field each frame.
10. NOT connected mode displays and stores everything except flags and CRC bytes.
11. Disk I/O from within the program.

Here is a rundown of major improvements.

A. REAL-TIME SYNCHRONOUS BIT CONVERSION:

In Volume 1, received packets were stored in memory using the byte per received bit approach. This was a great teaching tool as it allowed the user to visualize the SDLC received bit pattern a full page of memory (1024 bytes per page) at a time using the program's edit/modify mode. Each and every received bit, flags, data bits, and zero insertion bits were there to be seen. Some times a picture is worth a thousand words and it was quite useful for the newcomer to synchronous packet radio to be able to see the brilliant IBM synchronous data link control concept in action.

So much for the pluses of this approach. Its disadvantages were that it took precious time to decode tne data after the packet had been received and more importantly ate up memory faster than a hungry bear. The time factor was not noticeable with single frame packets, but was measurable when multi-frame packets of maximum length were received. The voluminous memory requirement for the byte per received bit storage was this approach's major detriment.

Along comes Sir Galahad, ne Gil Boelke-W2EUP, on his white charger to rescue Volume 2 from the memory monster. Not only does W2EUP's superb real-time serial synchronous bit to parallel decimal byte conversion subroutine solve the memory problem, but it also eliminates the measurable time delay for decoding long multi-frame packets.

The author's software digital phase-locked loop (DPLL) used in Volume 1, was again used in Volume 2 with only cosmetic changes. It was an old trusty/reliable friend and allowed the user to copy 1200 baud packets whose timing was off as much as 10 percent from the norm. It is somewhat analagous to the hardware approach used by the Intel 8273 dedicated SDLC controller MSI chip. Figure 1 illustrates two, bit time periods where there was a change from space to mark (mark and space are used only as colloquial terms since SDLC/HDLC are only interested in the relative change and not the absolute value),

Each 1200 baud 833.333 microsecond bit time is divided into quadrants with each quadrant testing for a change of state (mark or space) of the incoming serial synchronous data bit. Ideally, all transitions from mark to space or vice versa, will occur exactly between quadrants 2 and 3, so that the bit sample taken after time 4 is exactly at the dead-center of each incoming bit time. Since our software DPLL is somewhat less than ideal/perfect, it adjusts the variable time 4 countdown value so that the average bit transition is usually between time 2 and time 3. If it occurs during time 1 or time 4 a much larger correction is made to time 4 to bring the sample time back to near dead-center again.

All bit processing is done by the program between time 4 and time 1. The bit processing time is less than 10% of the total 833.333 microsecond bit time period so has little or no effect on the DPLL as long as each processing time period is exactly the same. Equalizing time delays in the processing routine are used to insure that the processing time period is exactly the same. Equalizing time delays in the processing routine are used to insure that the processing time period remains constant.

The DPLL's fixed time constants of TYME 1, 2, and 3 with values of 23 decimal are for the Model I TRS-80. The Model III with its slightly faster clock uses decimal 28 for TYME 1, 2, and 3. The DPLL subroutine's calculated TYME 4 countdown correction values are the same for both Models.

Figure 3 is the commented source code for the 1200 baud real-time synchronous to parallel decimal conversion, most of which from lines 900 to 1880 were generously contributed by W2EUP. The author's DPLL begins at the label TYME in line 1880 and runs through the end of this subroutine. Fig. 3 starts off with MODE which is the beginning of the receive mode subroutine. All the folderol before line 900 are simply the cues to tell you what the program has done automaticaly (if in the NOW CONNECTED mode of operation), such as displaying <CONNECTION ACK> on video if the program was in the AUTO ON mode, and so forth.

In the receive mode, the program continually cycles between NEWONE in line 690 and line 840/860 while looking for a valid (mark or space) change in the input from the EXAR 2211 demodulator via port zero. When a change is found, the program progresses to FL1 where it searches for the first opening flag. If the DCD (data carrier detect) from the EXAR 2211 drops before a flag is found, it starts all over again at BEFOR1.

Once an opening flag has been found, it proceeds to FL2 where further opening flags are ignored as this subroutine is searching for the first non-flag data byte in the frame. Again, if DCD drops it starts all over again at BEFOR1. When the first non-flag byte of the first frame is assembled, line 1270 jumps of to line 1600. The IN1 subroutine is the work horse of this real-time receive mode decoding section.

Only the first flag that is decoded by FL1 is stored at 40959 in memory. Decoded packet data bytes are stored from 40960 on up in memory by the IN1 subroutine. All converted bytes except frame ending flags are stored here for each packet. Each frame's ending flag location is stored sequentially in memory beginning at STORE.

After the entire packet has been decoded in real-time, IN1 exits to the MOVEM subroutine that is not shown in Figure 3 as it is too lengthy for this conference paper. MOVEM's function is determined solely by the mode the operator has selected; i.e., NOW connected or NOT connected.

B. AUTOMATIC REPEATER + NOW/NOT CONNECT:

In the NOW connected mode of operation each frame is CRC checked and if ok, the repeater segment of the address field then tested. If it contains your call letters and SSID, then the repeated bit is set for each frame, the CRC recalculated for each frame, and the total packet retransmitted. As such, your packet station serves as an automatic repeater. Video will display <FORWARDING> when this function is used. If the automatic repeater function is not called, the program then tests the other station's and your call letters + SSIDs (and repeater call letters + repeated bit where applicable) and if ok, sequentially tests each frame's control byte to determine the function.

Let's assume it was an information frame. Since you know who you are connected to, (the other station's call letters are displayed by the program in the 1st three right hand columns of Figure 2's main menu in both the auto and non-auto modes), ONLY the information field of each frame is displayed on video and stored in high memory. The ACK is then transmitted automatically while the video display remains in the receive mode. See Figure 2 for the main and shift menus.

In the NOT connected mode, everything except the flags and each frame's CRC bytes are displayed on video. The call letters and repeater if used, of the address field are right shifted one bit so as to display their real ASCII values. If you selected the NOW FORMAT option from the main menu, all carriage returns and line feeds are recognized and acted upon on the video display, If NOT FORMAT, they are ignored and the TRS-80 symbols for ASCII 13 and 10 displayed. NOW or NOT format may be used in either the NOW or NOT connected modes.

Intentionally, there is no CRC check of each frame in the NOT connected mode as we wish to see everything the EXAR 2211 is capable of demodulating, good and bad, up to 4K bytes in length per packet. Simultaneously with the video display function, all received bytes are stored sequentially in high memory beginning at 53248 decimal. Each received packet with CRC bytes may be inspected a full 1024 bvte page at a time by going to the edit/modify mode via press M from the main menu to go to 40960 in mid-memory. Press ENTER to move up a page or the MINUS key to move down a page. 40960+ is re-used bv each received packet. To inspect everything sequentially received so far (up to 12 pages = 12,288 bytes) except flags and CRC bytes, press shift M to take you to 53248+ in memory and then page up or down in memory as you wish. The BREAK key will return you to the main menu from the edit/modify mode.

C. HI-SPEED CRC USING THE BYTE-WISE LOOK-UP TABLE APPROACH SUGGESTED BY PEREZ:

Volume 1's software CRC generation and checking subroutines emulated the hardware approach used by IBM and the other SDLC controller chip manfacuturers. By that we mean the software aprroach emulated the same multi-shift register format and derived the CRC value on a bit by bit basis. It worked very well thank you, but it ate up precious time, especially with long multi-frame packets.

One approach we tried was to do the transmit mode CRC generation in real-time while the frame was actually being sent, just as the Intel 8273 SDLC controller chip does it and just as this program does the zero-insertion in real-time. It worked, but it solved the wrong problem. The real problem lay in the receive mode CRC checking time delay that was measurable when maximum length multi-frame packets were being received.

Much like Sir Galahad saving the SDLC maiden from the memory monster, along comes Sir Lancelot, ne Aram Perez, and saves the CRC damsel from the time eating dragon. The June '83 issue of IEEE Micro Journal had a fascinating paper by Aram Perez that covered his 'byte-wise' CRC look-up table approach for the CRC16 (Bisync) polynominal. Without too much difficulty we were able to generate a look-up table for the IBM SDLC polynomial used by the AX.25 and Vancouver protocols.

The results? An incredible 27 times SPEED-UP of both CRC checking and generation compared with Volume 1 of the software approach. The majority of this section and its subroutine is courtesy of Mr. Perez' excellent paper.

The CRC we will cover will detect:

- all one or two bit errors.
- all odd number of bit errors.
- all burst errors less than or equal to the the degree of the polynomial used.
- most burst errors greater than the degree of the polynomial used.

What this adds up to in AX.25 protocol is a probability in the vicinity of 10 to umpteenth power, that if the CRC tests ok, the received frame that was CRC checked is correct and identical to that which was transmitted. If it is good enough for banks to transfer funds by electronic mail (it is), it should be good enough for us.

HERE IS HOW IT WORKS:

In a protocol utilizing the cyclic redundancy check, the message to be transmitted between the last opening flag and the closing flag in each frame is considered to be a binary polynomial M(X). It is first multiplied by X to the K power and then divided by an arbitrary generator polynomial G(X) of degree K. This yields a quotient Q(X) and a remainder R(X) divided by G(X). All arithmetic is done in modulo 2; i.e., the results of subtraction are equal to the results of addition. The equation looks like this:

$$\frac{x\,M(X)}{G(X)} = Q(X) \oplus \frac{R(X)}{G(X)}$$

The $\oplus$ sign signifies addition in modulo 2 arithmetic. Simplifying this equation yields:

$$x\,M(X) \oplus R(X) = Q(X)G(X)$$

Where R(X) will always be of degree K or less. The CRC algorithm calculates R(X) and tacks these 2 bytes onto the end of the frame to be transmitted. Since as illustrated above, x M(X) $\oplus$ R(X) does indeed equal Q(X)G(X), the original message with the 2 byte CRC tacked on will be evenly divisible by G(X), IF and only IF no bits were erroneously received. Using the IBM SDLC (CCITT) polynomial as shown below, the remainder will always be 61624 decimal IF the frame was received correctly.

IBM SDLC AND BISYNC GENERATOR POLYNOMIALS

NOTE the [figure = exponentiation

IBM SLDC (CCITT)	X[16+X[12+X[5+1
SDLC REVERSE	X[16+X[11+X[4+1
CRC1 6 (BISYNC)	X[16+X[15+X[2+1
CRC16 REVERSE	X[16+X[14+X[1+1

The reverse polynomials are the same as their big brothers, except taken in reverse order. Since the rather simple CRC arithmetic is done in modulo 2, it is quite easily implemented by the MSI chips used by both Vancouver and TAPR TNC boards. The former using the Intel 8273 MSI chip and the latter using the Western Digital 1933/1935 MSI chip.

One of the drawbacks to using the hardware rather than the software approach is that the user never knows what the CRC value is that he/she transmitted or received. Some packet operators could care less, but then again, some radio amateurs prefer to fully understand what they are doing.

This program allows you to read out exactly what the generated and received CRC values are for every packet that is transmitted or received by using the edit/modify mode.

Unfortunately there is no such thinq as 'free lunch.' The price we have to pay for this extremely FAST CRC subroutine is a modest bit of memory for the 512 byte lookup table. Nevertheless, it is a small price to pay for the near 'speed of light' swiftness gained. Again, this approach is 27 times faster than the bit by bit CRC routine used in Volume 1.
Both received frame CRC checking and transmit frame CRC generation are each quite simple using Aram Perez' byte-wise approach modified for IBM SDLC (CCITT) polynomial. Let's look at the transmit mode CRC generation first.

All frames to be transmitted are first moved to MEM location 43008 + a frame at a time, then the CRC is generated, and inserted. For multi-frame packets, a frame is moved, the CRC generated for that frame and inserted, and then the next frame moved, CRC generated and inserted, etc. This only requires milliseconds of real-time.

The memory location denoted by the label ENDCRC always contains the generated CRC value of the packet just transmitted IF it was a single frame packet or the generated CRC value of the last frame transmitted if it was a multi-frame packet. If the current packet being transmitted is a single frame info packet and the program in the NOT connected mode, the CRC value in decimal will be displayed on the top line of video, and the packet immediately below it while it is being transmitted.

Why bother with displaying the CRC decimal value in the unconnected mode?

Only for convenience. Sometimes it can be very useful for the station at the other end who is trouble shooting his/her receive mode system. Even the hardware approach using the Western Digital WD-1933 or Intel 8273 SDLC chips can on occasion have problems with its real-time CRC. Some of the early SDLC controller chips exhibited this type of problem.

Figure 4 starts off with the commented source code for generating the two IBM SDLC CRC bytes for each frame to be transmitted. Almost the same routine is used for testing the CRC value of each incoming frame of each packet. See lines 870 through lines 990 of this subroutine for the receive mode CRC check. For either transmitted or received frames, this CRC function is accomplished virtually in real-time.

D. TRANSMITTING MULTI-FRAME PACKETS:

Data for the information fields of all multi-frame packets originates in low memory beginning at 17408 decimal. 12288 LO-MEM bytes are reserved here for the automatic multi-frame transmit function. Data may be input directly from the keyboard by pressing shift L to go to 17408 in LO-MEM in the edit/modify mode and then typing away till done, or data may be input from disk without leaving the program.

Referring to Figure 2's main menu, the operator presses G to input the number of frames per packet 1 - 7, and then presses N to input the information field length of 1 to 256 bytes per frame. Actually, any info field length up to 2000 bytes may be specified for use between agreeing packeteers if a reliable path is available. Now, press E to commence the LO-MEM multi-frame transmit function.

In NOW connected mode, the program

will calculate the number of frames to be transmitted, divide them by the number of frames per packet specified, calculate the total number of packets to be transmitted, calculate the number of frames in the last packet, and calculate the number of bytes in the last frame of the last packet. It will then begin sending them automatically. While they are being transmitted, the top line of video will display the remaining number of frames to be transmitted, and up to the first 15 lines of the packet being sent.

After the packet is transmitted, the program will switch to the receive mode and display <OK> if the acknowledgment was correctly received, or <RESENDING> if it was not received correctly or the T1 resend timer times out. Assuming that the ACK was correctly received, it will then assemble and transmit the next packet. The total assembly time for each multi-frame packet including CRC'ing each frame, is measured in milliseconds. This process will continue automatically till all LO-MEM data has been transmitted and acknowledged.

In the NOT connected mode, the operation is almost identical to that just described, except the operator must press the E key from the main menu to sequence and then transmit each packet till all LO-MEM has been transmitted, as ACK's will obviously not be received. This function is seldom if ever used in the NOT connected mode and was included only to satiate one of our rather unique BETA testers who gets his jollies from sending long multi-frame packets in this mode.

Figure 5 is the commented source code for the multi-frame transmit mode subroutine. It is easy to follow when one understands how the regular registers, alternate registers, and stack are used from SEND7 onward.

REGULAR REGISTERS:
A = parallel byte from memory
BC= time delay routine in SN1
D = parallel byte value in SN1
E = bits per byte counter SN1
HL = JP (HL) countdown jump SN1
IX = unused
IY = xmit byte memory location

ALTERNATE REGISTERS:
A = unused
B = frames in the current packet
C = last frame last pack pointer
DE = last frame last packet length
HL = frame length except for last

STACK IN SEND7:
Bytes remaining to send in frame

The SEND7 subroutine in Figure 5 is not really a sticky wicket if one realizes that the program always sets alternate C register to 1 more than B register, except for the last packet being transmitted from LO-MEM. As such, it never jumps to KYBD4B except for the last frame of the last packet. For the last multi-frame packet only, alternate C and alternate B are set to one less than the number of frames to transmit in this final packet. When the next to last frame of this last packet has been transmitted, alternate C is decremented to zero, so jumps off to KYBD4B that pushes alternate DE on the stack which is the length of the final frame of the last packet.

The SN1 and SN1A subroutines in Figure 5 do the yeoman job of converting the parallel decimal byte to the synchronous 1200 baud serial bit that is output via port zero. SN1A is used for 126 decimal flags that do not utilize zero-insertion, and SN1 is called for data bytes between flags that may require zero-insertion.

E. DISK I/O FROM WITHIN PACKET PROGRAM:

At first glance appears as easy as falling off a log. Always be suspicious of easy logs in this game. On second glance, when one realizes that virtually all of RAM memory from 17408 to 28672 is used by the TRS-80 for disk subroutines, and this is the area where the software approach stores long data from the keyboard or disk to be transmitted in multi-frame packets, it becomes apparent that both the packet data and disk subroutines cannot occupy the same memory at the same time.

One simple solution is to leave the packet program, do the disk I/O functions desired, return to the packet program, clear out low memory, and resume packeteering. Though simple and easy to accomplish, it is a decided inconvenience and time consuming approach for the operator.

What we desired was having our cake and eating it too; i.e., having the write to disk and read from disk functions within the software approach program, while at almost the same time being able to use low memory for storing long data to be transmitted in multi-frame packets.

The solution to this apparent paradox was to save the TRS-80's minimum disk operating system (system 1) in mid-memory and write our own disk I/O subroutine that this section delineates. Our disk I/O subroutine requires only 1859 bytes of memory and serves three purposes:

1. Volume 2 is a teaching textbook as well as a working AX.25 program. As such, it familiarizes the reader with writing direct disk I/O subroutines.
2, Allows disk I/O without leaving the packet program.
3. Provides the basis for Volume 3's automatic-unattended disk access by another packet station. In essence, it is a mini-version of a computer bulletin board system with the SEND, SAVE, LIST,

and HELP commands sent via packet.

Figure 2% SHIFT menu illustrates the 3 commands used for the disk I/O functions from within the software approach program. Shift R loads a disk file of up to 12K bytes in length to high memory (53248 up) and shift D moves it low memory for multi-frame packet transmission. Shift Q saves up to 12K bytes of high memory in a disk file of whatever name the operator wishes to give it. The high memory data may be either input from the keyboard using the edit/modify mode, or conversely may be received packets the operator wishes to save on disk.

Figure 6 is the commented source code for this subroutine which is largely self-explanatory. It works quite well with the Model I TRS-80 and on a maybe basis for the Model III TRS-80 depending upon which version of ROM the user's system has installed.

F. REAL-TIME EDIT/MODIFY/MONITOR MODE FOR COMPUTER NETWORKING PROGRAMS:

Whether the software or hardware approach to packet radio is used, we have found that an in-program (within the terminal interface program TIP) subroutine that allows instant access to the computer's 64K bytes of memory, 1024 bytes per page displayed on video, is a useful adjunct to the packet operator% tool kit.

Memory may be reviewed in the edit mode and modified in the modify mode if desired. If the operator wishes to save the modified TIP it may be dumped to disk thus eliminating the time consuming requirement of exiting the TIP program, loading the TIP source code into an Editor/Assembler, modifying the source code, assembling the program, and then writing it on disk.

In addition to the edit/modify/monitor functions, this subroutine serves as the keyboard input subroutine for typing packet messages into low memory beginning at 17408. Up to 12 pages, 1024 bytes per page, may be used by enthusiastic typists. A carriage return followed by a line feed is input by pressing ENTER, End of message delimiters, 128 decimal, are input by pressing shift zero.

The short 866 byte subroutine that performs the edit/modify mode functions is illustrated in Figure 7 which is the commented source code.

The edit/modify program may be considered a subroutine if you wish, but it is truly a stand alone program that may be appended to any variety of software where the user wishes to access to all 64K bvtes of memory WITHOUT leaving the program. Depending on the ROM/RAM varieties in the particular computer, the user may not only review, but actively modify anywhere from 48K to 64K of memory while the program is up and running.

EDIT/MODIFY PROGRAM ENTRY POINTS:

There are 3 entry points to save the user the trouble of having to page too far through memory. They may be called from the TIP program's main menu in Figure 2 by:

1. Press M to go to the 1024 byte page of beginning in mid-memory at 40960 decimal.
2. Pressing SHIFT M from the menu will display-the 1024 byte memory paqe beginning at 53248 in high memory.
3. SHIFT L from the menu will display the 1024 byte memory page beginning at 17408 in low memory.

We will assume you pressed M from the TIP menu which takes us to memory location 38912 that is in line 5240 of Figure 7's source code program. Had you pressed SHIFT M or SHIFT L, then the HL register would have been loaded with 53248 or 17408, respectively and the jump made to 38915 in MEM that is in line 5250.

The rest of the subroutine in Figure 7's commented source code is largely self-explanatory.

The edit/modify/monitor in-program subroutine is a useful tool for the packeteer. It is elegant in its simplicity, yet a very POWERFUL tool. By all means modify it to suit your own operating habits and fancy. If you are used to using memory modifier and/or monitor programs such as SUPERZAP, DEBUG, ZAPSIT, etc., you may abandon them for this short in-program subroutine once you become accustomed to using it.

A new version of the edit/modify subroutine using a number of the Electric Pencil (tm) word processing program commands for keyboard input of packet messages may be implemented later this year.

CONCLUSION:

First, a personal note. Writing the 'software approach' for both Volumes 1 and 2 was great fun and very gratifying.

why?

Because so many experienced packeteers told us it could not be done using a modestly priced 2 MHz ballpark crystal clock Model I or Model III TRS-80 microcomputer. Actually, most any computer with a 1 MHz or faster clock should be able to handle 1200 baud synchronous packet using the software approach. The Model I or Model III TRS-80 is quite capable of running 2400 baud packet using this program if the timing constants are carefully trimmed and adjusted.

With the new Ziloq Z-800 microprocessor and its extremely fast clock, (and internal cache memory), the software approach may be extended to 9600 baud and well beyond.

Want to dig deeper? If so, try Volume 1 or 2 of 'Synchronous Packet Radio Using The Software Approach.'

Vol. 1 - Vancouver Protocol is $22 ppd and Vol. 2 - AX.25 Protocol also $22 ppd. If you want the programs on disk in addition to the book which is required for instructions to personalize the disk, specify Model I or Model III TRS-80. The disk programs are an additional $29 ppd. U.S. phone orders are welcome during business hours at (716)-753-2654 or you may order from:

Richcraft Engineering Ltd.
#1 Wahmeda Industrial Park
Chautauqua, New York 14722

Do not want to dig deeper? Then we highly recommend to you the Tucson Amateur Packet Radio terminal node controller. It is a highly efficient, very professional, and first-rate kit. It is available for $252 which is about one half the price were it produced by a profit making enterprise that most likely would not do as thorough a job as TAPR.

We salute TAPR and all those who have contributed to the development of its TNC, for an outstanding service to amateur radio.

REFERENCES:

Proceedings - 2nd ARRL Amateur Radio Computer Networking Conference
AX.25 Protocol pp 4 -14
by Terry Fox, WB4JFI

Proceedings : 2nd ARRL Amateur Radio Computer Networking Conference
Link Level Address Mechanisms pp 47-49
by Henry S. Magnuski, KA6M

IEEE Micro Journal - June 1983 issue
Byte-Wise CRC Calculations pp 40 - 50
by Aram Perez

QST - February 1984 issue
'On Line' Column pp 77
by Stan Horzepa, WA1LOU

Gunnplexer Cookbook - A 10 GHz Microwave Primer 335 pp
by Bob Richardson, W4UCH

Advanced Baudot RTTY for the TRS-80
Vol 5 Disassembled Handbook 223 pp
by Bob Richardson, W4UCH

AX.25 Protocol Modifications/Update
personal communication
by Paul L. Rinaldo, W4RI

REFERENCES continued:

Z-800 Micro-P Product Specification
Zilog, Inc.
1315 Dell Avenue
Campbell, CA 95008

FIGURE 1

1200 BAUD SOFTWARE DIGITAL PHASE-LOCKED LOOP QUADRANTS

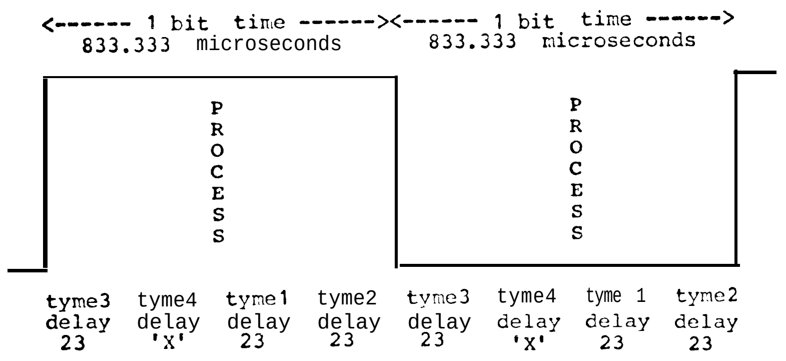

FIGURE 2:

```
                 ENTER OPTION DESIRED ? _

CHANGE ADDRESSEE CALL LTRS = A   W2EUP CONNECT REQUEST CQ  = B
MOT CONNECTED TOGGLE       = C   W2EUP DISCONNECT REQUEST  = D
SEND PACKETS FRO!: LO-MEM  = E   W2EUP CONNECT ACKNOWLEDGE = F
INPUT FRAMES/PACKET LO-MEM = G   THIS IS AX.25 PROTOCOL    = H
BACKOFF DELAY TOGGLE OFF   = I   AUTO CONNECT TOGGLE OFF   = J
NOW IN UPPER CASE MODIFY   = K   W2EUP - GIL BOELKE MESSAGE = L
DISPLAY/EDIT MEMORY PAGE   = M   SET INFO FIELD LOMEM PACKS = N
NOW FORMAT VIDEO TOGGLE    = 0   QUICK BROWN FOX MESSAGE   = P
VIA WA2EGW/R REPEATER ON   = Q   SET OPENING FLAG LENGTH   = R
CHANGE REPEATER CALL LTRS  = S   INPUT/XMIT NORMAL INFO = V & T
CLEAR NON-PGM MEM 17K-62K  = U   INPUT/XMIT UNNUMB INFO = V & W
ABORT LOW-MEM PAK SEQUENCE = X   NOT CONEK TO OWN STATION  = Y
SHIFT MENU                 = 1   SET RE-TRY IN CONNECT MODE = 2
SEND WAIT REQUEST (RNR)    = 3   SEND CLEAR WAIT (RR)      = 4
(not shown) :                    (not shown):
DISCONNECTED MODE          = 5   FRMR FRAME REJECT         = 6

                       SHIFT MENU 3 _

XMIT 40960 Up CONTINUOUSLY = A   BOOT DOS READY              = B
LOAD MID-ME": ASCII UUUUUU = C   MOVE HI-MEM TO LOW-MEMORY   = D
EDIT/MODIFY INSTRUCTIONS  = E    CHANGE RECEIVE DPLL BASE #  = F
TRANSMIT EXTERNALLY ONLY   = G   TRANSMIT TO HI-MEMORY ONLY  = H
SEND MORSE: I.D.           = I   SEND SEQUENTIAL ACKS        = J
CAUTION ** RESTORE DOS **  = K   DISPLAY LOW MEMORY @ 17403  = L
DISPLAY RECV PACKS @ 53248 = M   RESTORE PROGRAM POINTERS    = N
OSCAR 10 CALL/HANDLE LIST  = 0   MOVE PROGRAM TO LOW MEMORY  = P
SAVE HI-MEM OIJ DISK       = Q   LOAD DISK FILE TO HI-MEM    = R
TRANSMIT BAUD RATE SELECT  = S   TEST OTHER STATION STATUS   = T
CLEAR HI-MEMORY 53248 +    = U   SEND MORSE FROM KEYBOARD    = V
RECEIVE AX.25 PROTOCOL     = W   RECV VANCOUVER NOT CONNECT  = X
NORMAL DISPLAY - NOT DPLL  = Y   DISPLAY DPLL LAST QUADRANT  = Z
         NOTE: SPACE BAR IN RECEIVE MODE = RESLND LAST PACK
```

FIGURE 3

```
00100 ;
00110
00120 ; RECEIVE MODE REAL-TIME SDLC/HDLC SERIAL SYNCHRONOUS
00130 ; DATA STREAM TO PARALLEL DECIMAL BYTE CONVERSION.
00140
00150 ; THE REGISTERS USED IN THIS RECEIVE MODE SUBROUTINE FROW
00160 ; LINE 900 ON ARE:     REGULAR REGISTERS
00170 ; A = USED + NEW PORT ZERO VALUE IN EACH DPLL QUADRANT
00180 ; F = USED THROUGHOUT
00190 ; B = DPLL COUNTDOWN VALUE FOR FIRST 3 DPLL QUADRANTS
00200 ; C = 8 BITS PER BYTE COUNTER
00210 ; D = CALCULATED DPLL COUNTDOWN VALUE FOR 4TH QUADRANT
00220 ; E = LAST PORT ZERO VALUE
00230 ; HL= MEM LOCATION TO STORE ENDING FLAG ADDRESS
00240 ; IX= ONLY FOR EQUALIZING TIME DELAYS; INC IX & DEC IX
00250 ; IY= UNUSED
00260 ;                       ALTERNATE REGISTERS
00270 ; A = UNUSED
00280 ; F = UNUSED
00290 ; B = RECEIVED PARALLEL BYTE WITH ZERO-DELETION
00300 ; C = RECEIVED PARALLEL BYTE WITHOUT ZERO-DELETION
00310 ; D = INCOMING BIT VALUE AT CENTER OF BIT TIME FRAME
00320 ; E = LAST BIT VALUE AT CENTER OF BIT TIME FRAME
00330 ; HL= MEM LOCATION TO STORE CONVERTED DECIMAL BYTE
00340
00350 ; THIS SUBROUTINE IS ENTERED IN LINE 440, 490,OR 500
00360 ; DEPENDING ON WHETHER RECEIVE MODE IS ENTERED FROM THE
00370 ; MAIN MENU, NOT CONNECTED MODE, OR NOW CONNECTED MODE.
00380
00390 ; THE SOFTWARE DIGITAL PHASE LUCKED LOOP (DPLL) IS AT THE
00400 ; END OF THIS SUBROUTINE IN LINES 1880 - 2230.
00410
00420 ; SIGNIFICANT RECEIVE MODE SUBROUTINES FROM VOLUME 2
00430
00440 MODE    LD      BC,6500         ;.01 SECOND DEBOUNCE
00450         CALL    060H            ;TIME DELAY SINCE THE
00460         LD      A,(14400)       ;CLEAR KEY IS USED TO
00470         CP      2               ;TOGGLE BETWEEN THE MENU
00480         JP      Z,MODE          ;AND RECEIVE MODE.
00490 MODE1   CALL    RESKCV          ;RESTORE RECEIVE VIDEO
00500 MODE1A  CALL    TESTSP          ;TEST SP FOR PGM ERRORS
00510         LD      A,(SIGN7)       ;DISPLAY ON VIDEO
00520         CP                      ;THAT A CONNECTION
00530         CALL    Z,CNRQ          ;ACKNOWLEDGE WAS SENT.
00540         LD      A,(SIGN6)       ; IF LONG DATA FROM LOMEM,
00550         CP      1               ;UP TO 12288 BYTES, THEN
00560         CALL    Z,SETIT         ;RESET POINTERS.
00570         LD      A, (SIGN5)      ;IF AX.25 STATUS REQUEST,
00580         CP      1               ;THEN DELAY 1 SECOND
00590         JP      Z,SPACK-10      ;BEFORE SENDING RR/RNR.
00600         LD      A, (SIGN4)      ;DISPLAY ON VIDEO
00610         CP                      ;THAT <DISCONNECT ACK>
00620         CALL    Z,DISCAK        ;WAS TRANSMITTED.
00630 BEFOR1  EXX                     ; SWAP ALTERNATE REGISTERS
00640         LD      HL,40959        ;MIDMEM TO STGRE PACKET
00650         LD      DE,0            ;INITIALIZE AT ZERO
00660         LD      BC,0            ;INITIALIZE AT ZERO
00670         EXX                     ;RESTORE REG. REGISTERS
00680         CALL    CLRMUL          ;CLEAR CLOSING FLAG STORE
```

```
00690 NEWONE LD     A,(AUT)       ;AUTOMATIC CONNECT MODE ?
00700        CP     1             ;IF SO, AND CONNECTED, T2
00710        CALL   Z,TIMOUT      ;TIMES OUT 6 1/2 MINUTES.
00720        LD     A,(RTRY)      ;IN RE-TRY CONDITION ?
00730        CP     1             ;THEN ACTUATE T1 RE-TRY
00740        CALL   Z,TESTRY      ;TIMER BEFORE RESENDING.
00750        IN     A,(0)         ;EX-2211 OUTPUT PORT ZERO
00760        LD     D,A           ;SAVE IT IN 'D' REGISTER
00770        LD     A, (14400)    ;KEYBOARD PSUEDO MEMORY
00780        CP     2             ;CLEAR KEY PRESSED ?
00790        JP     Z,MENU0       ;IF SO, GOTO MAIN MENU
00800        CP     128           ;SPACE BAR PRESSED ?
00810        JP     Z,RSEND       ;IF SO, MANUAL RESEND.
00820        IN     A,(0)         ;EX-2211 OUTPUT PORT ZERO
00830        CP     D             ;ANY CHANGE SINCE LAST ?
00840        JP     Z,NEWONE      ; IF NOT, GO LOOK AGAIN
00850        BIT    0,A           ;DCD CARRIER DETECT ?
00860        JP     Z,NEWONE      ;NO 1200/2200 TONES
00870        LD     HL,STORE      ;END FLAG ADDRESS STORE
00880        LD     A,(DVAL)      ;DPLL COUNTDOWN VALUE
00890        LD     D,A           ;START OFF WITH NOMINAL
00900 FLG1   CALL   TYME          ;SOFTWARE DPLL LINE 1880
00910        INC    (IX)          ;EQUALIZING TIME DELAY
00920        DEC    (IX)          ;EQUALIZING TIME DELAY
00930        INC    (IX)          ;EQUALIZING TIME DELAY
00940        DEC    (IX)          ;EQUALIZING TIME DELAY
00950        BIT    0,A           ;DCD CARRIER DROPPED ?
00960        JP     Z,BEFOR1      ;THEN START OVER AGAIN
00970        EXX                  ;SWAP ALTERNATE REGISTERS
00980        LD     D,A           ;SAVE INCOMING BIT IN 'D'
00990        XOR    E             ;COMPARE WITH LAST ONE
01000        CPL                  ;DATA IN BIT 7
01010        LD     E,D           ;UPDATE E FOR NEXT ONE
01020        RLCA                 ;SHIFT INTO CARRY
01030        RR     C             ;INCOMING BIT PATTERN
01040        LD     A,C           ;SWAP FOR COMPARE
01050        CP     126           ;FOUND AN OPENING FLAG ?
01060        JP     Z,FLG2+31     ;IF SO, GOT0 LINE 1280
01070        EXX                  ;ELSE GO BACK TO FLG1
01080        JP     FLG1          ;START LOOKING AGAIN.
01090 FLG2   BIT    0,A           ;DCD CARRIER DROPPED ?
01100        JP     Z,BEFOR1      ;THEN START OVER AGAIN
01110        EXX                  ;SWAP ALTERNATE REGISTERS
01120        LD     D,A           ;INCOMING BIT VALUE TO D
01130        XOR    E             ;COMPARE WITH LAST ONE
01140        LD     E,D           ;UPDATE E FOR NEXT TIME
01150        CPL                  ;DATA IN BIT 7
01160        RLCA                 ;SHIFT INTO CARRY
01170        RR     B             ;INPUT DATA -
01180        RRCA                 ;ACCUMULATES HERE.
01190        RR     C             ;INCOMING BIT PATTERN
01200        LD     A,C           ;TEST IT
01210        CP     126           ;FOR ANOTHER OPENING FLAG
01220        JP     Z,FLG2+41     ;IF SO, JUMP TO LINE 1330
01230        LD     A,B           ;BUILT UP DATA VALUE
01240        EXX                  ;RESTORE REG. REGISTERS
01250        DEC    C             ;DECREMENT BIT COUNTER
01260        JP     NZ,FLG2+35    ;NOT ZERO, GET NEXT BIT
01270        JP     IN1A+1        ;1ST FRAME DATA GOTO 1600
01280        LD     (HL),A        ;STUFF 1ST FLAG HERE
01290        EXX                  ;RESTORE REG. REGISTERS
01300        LD     C,8           ;RESET BIT/BYTE COUNTER
01310        CALL   TYME          ;DIGITAL PHASE LOCK LOOP
01320        JP     FLG2          ;GO LOOK AGAIN
01330        INC    (IX)          ;EQUALIZING TIME DEALY
01340        DEC    (IX)          ;EQUALIZING TIME DELAY
01350        JP     FLG2+32       ;GO LOOK FOR NEXT BYTE
01360 IN1    BIT    0,A           ;PACKET TONES DROPPED ?
01370        JP     Z,MOVEM+1     ;IF SO, PROCESS IT.
01380        EXX                  ;SWAP ALTERNATE REGISTERS
01390        LD     D,A           ;INCOMING BIT VALUE TO D
01400        XOR    E             ;COMPARE WITH LAST ONE
01410        LD     E,D           ;UPDATE E FOR NEXT TIME
01420        CPL                  ;DATA IN BIT 7
01430        RLCA                 #SHIFT INTO CARRY
01440        RR     B             ;INPUT DATA BITS -
01450        RRCA                 ;ACCUMULATES HERE.
01460        RR     C             ;INCOMING BIT PATTERN
01470        LD     A,C           ;TEST IT
01480        CP     126           ;FOR A CLOSING FLAG ?
01490        JP     Z,FL1         ;IF SO, GOT0 LINE 760
01500        CP     254           ;PACKET TONES DROPPED ?
01510        JP     Z,MOVEM       ; IF SO, PROCESS IT
01520        AND    254           ;REMOVE BIT ZERO
01530        CP     124           ;0111110X PATTERN ?
01540        JP     Z,DELETE      ;IF SO, DO ZERO DELETION
01550        LD     A,B           ;BUILT UP DATA VALUE
01560        EXX                  ;RESTORE REG. REGISTERS
01570        DEC    C             ;DECREMENT BIT COUNTER
01580        JP     NZ,IN4        ;NOT ZERO, GET NEXT BIT
01590 IN1A   NOP                  ;SAVED FOR DPLL TESTING
01600        EXX                  ;SWAP ALTERNATE REGISTERS
01610        INC    HL            ;BYTE STASH MEM LOCATION
01620 IN2    LD     (HL),A        ;STASH IT IN MEMORY
01630        LD     A,H           ;TOO LONG A PACKET ?
01640        CP     176           ;OVER 4096 BYTES LONG ?
01650        JP     Z,MOVEM-3     ;IF SO, PROCESS IT
01660 IN3    EXX                  ;RESTORE REG. REGISTERS
01670        LD     C,8           ;RESET BITS/BYTE COUNTER
01680 IN4    CALL   TYME          ;DIGITAL PHASE LOCK LOOP
01690        JP     IN1           ;CONVERT INCOMING BIT
01700 FL1    PUSH   HL            ;GOT A CLOSING FLAG
01710        EXX                  ;RESTORE REG. REGISTERS
01720        POP    Bc            ;FLAG LOCATION MINUS ONE
01730        INC    Bc            #FLAG MEM LOCATION
01740        LD     (HL),C        ;STORE FLAG ADDRESS LSB
01750        INC    HL            ;NEXT STORE LOCATION
01760        LD     (HL),B        ;STORE FLAG ADDRESS MSB
01770        INC    HL            ;NEXT STORE LOCATION
01780        LD     A,144         ;OUT OF BOUNDS DUE TO -
01790        CP     H             ;RUN AWAY TNC ?
01800        JP     Z,MOVEM+1     ;IF SO, PROCESS IT
01810        JP     IN3+1         ;ELSE GO FOR NEXT ONE
01820 DELETE RL     B             ;ZERO DELETION, SO -
01830        EXX                  ;BACKUP ALTERNATE B
01840        INC    (IX)          ;EQUALIZING
01850        DEC    (IX)          ;TIME DELAY.
01860        CALL   TYME          ;DIGITAL PHASE LOCK LOOP
```

```
01870          JP      IN1            ;CONVERT NEXT BIT
01880 TYME     LD      A,(14400)      ;ESCAPE IS CLEAR KEY
01890          CP      2              ;IF PRESSED GOTO -
01900          JP      Z,MENU0-1      ;MAIN MENU FOR INSTRUCTS.
01910          LD      B,23           ;MODEL I COUNTDOWN VALUE
01920 TYME1    DJNZ    TYME1          ;1ST QUADRANT COUNTDOWN
01930          IN      A,(0)          ;PORT ZERO VALUE TO 'A'
01940          CP      E              ;ANY CHANGE FROM LAST ?
01950          JP      NZ,DEC2        ;IF SO, GOT0 LINE 2120
01960          LD      B,23           ;MODEL I COUNDOWN VALUE
01970 TYME2    DJNZ    TYME2          ;2ND QUADRANT COUNTDOWN
01980          IN      A,(0)          ;PORT ZERO VALUE TO 'A'
01990          CP      E              ;ANY CHANGE FROM LAST ?
02000          JP      NZ,DEC1        ;IF SO, GOT0 LINE 2150
02010          LD      B,23           ;MODEL I COUNTDOWN VALUE
02020 TYME3    DJNZ    TYME3          ;3RD QUADRANT COUNTDOWN
02030          IN      A,(0)          ;PORT ZERO VALUE TO 'A'
02040          CP      E              ;ANY CHANGE FROM LAST ?
02050          JP      NZ,INC1        ;IF SO, GOT0 LINE 2180
02060          LD      B,D            ;ADJUSTED COUNTDOWN VALUE
02070 TYME4    DJNZ    TYME4          ; 4TH QUADRANT COUNTDOWN
02080          IN      A,(0)          ;PORT ZERO VALUE TO 'A'
02090          CP      E              ;ANY CHANGE FROM LAST ?
02100          JP      NZ,INC2        ;IF SO, GOT0 LINE 2210
02110          RET                    ;DPLL DONE. GO PROCESS IT
02120 DEC2     LD      E,A            ;SAVE NEW BIT IN 'E'
02130          LD      D,15           ;WAY TOO LATE, SO SHORTEN
02140          JP      TYME2-2        ;LAST QUAD COUNT A BUNCH.
02150 DEC1     LD      E,A            ;SAVE NEW BIT IN 'E'
02160          LD      D,20           ;TINY BIT TOO LATE, SO -
02170          JP      TYME3-2        ;SHORTEN LAST QUAD A BIT.
02180 INC1     LD      E,A            ;SAVE NEW BIT IN 'E'
02190          LD      D,24           ;TINY BIT TOO SOON, SO -
02200          JP      TYME4-2        ;LENGTHEN LAST QUAD A BIT
02210 INC2     LD      E,A            ;SAVE NEW BIT IN 'E'
02220          LD      D,29           ;WAY TOO SOON, LENGTHEN
02230          RET                    ;LAST QUADRANT A BUNCH.
02240 ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
02250 ; END OF SYNCHRONOUS BIT TO PARALLEL BYTE CONVERSION VOL 2
```

```
00100 ;                  FIGURE 4
00110
00120 ; IBM SDLC CRC GENERATION AND CRC CHECKING SUBROUTINES
00130
00140 ; CRC1 AND CRC2 ARE FOR GENERATING THE 2 BYTE CRC VALUE
00150 ; FOR A FRAME OF (LENG1) BYTES IN LENGTH.  ADDREZ IS THE
00160 ; MEMORY LOCATION OF THE BEGINNING OF THE SINGLE FRAME
00170 ; PACKET TO BE TRANSMITTED.  MULTI-FRAME PACKETS USE A
00180 ; VARIABLE ADDREZ DEPENDING UPON WHERE EACH FRAME HAS
00190 ; BEEN SEQUENTIALLY MOVED IN MEMORY STARTING AT 43008.
00200
00210 ; RCRC BEGINNING IN LINE 870 TESTS THE RECEIVED CRC VALUE
00220 ; OF A FRAME STARTING AT (BGINIT) IN MEMORY WITH A TOTAL
00230 ; LENGTH OF 'BC' REGISTER BYTES.  MULTI-FRAME PACKETS OF
00240 ; 1 TO 7 FRAMES/PACKET ARE SEQUENTIALLY ACCOMODATED.
00250
00260 ; TABLE BEGINNING ON PAGE THREE IS THE LOOK-UP TABLE FOR
00270 ; THE BRILLIANT 'BYTE WISE' CRC SUBROUTINE SUGGESTED BY
00280 ; ARAM PEREZ IN THE THE JUNE '83 ISSUE OF I.E.E.E. MICRO.
00290 ; THE TABLE VALUES FOR THE IBM SDLC 'CRC' WERE GENERATED
00300 ; BY W4UCH AS THE PEREZ PAPER ONLY GAVE THE CRC16 VALUES.
00310
00320 CRCVAL  DEFW    0              ;RECEIVE CRC VALUE STASH
00330 ENDCRC  DEFW    0              ;XMIT CRC VALUE STASH
00340 CRC1    LD      HL,ADDREZ      ;BEGIN MESSAGE LOCATION
00350         LD      BC,(LENG1)     ;LENGTH OF FRAME IN BYTES
00360         LD      DE,65535       ;INTIALIZE DIVIDEND 1'S
00370         CALL    CRCT           ;GENERATE CRC LINE 490
00380         CALL    FINCRC         ;SORT/STUFF RIGHT ORDER
00390         LD      A,(SIGN2)      ;DISPLAY CRC VALUE -
00400         CP      1              ;ON VIDEO DISPLAY ?
00410         RET     Z              ;IF NOT, RETURN.
00420         LD      HL,(ENDCRC)    ;IF SO, THEN DISPLAY IT
00430         CALL    DZ             ;ON TOP LINE OF VIDEO.
00440 CRC2    LD      BC,960         ;= 15 LINES OF VIDEO
00450         LD      HL,ADDREZ      ;BEGIN PACKET ADDRESS
00460         LD      DE,15424       ;2ND LINE OF VIDEO
00470         LDIR                   ;DISPLAY MESSAGE SENT
00480         RET                    ;RETURN WHENCE U CAME +1
00490 CRCT    LD      A,(HL)         ;FIRST BYTE TO CRC
00500         INC     HL             ;INCREMENT FOR NEXT ONE
00510         PUSH    BC             ;SAVE BYTES TO CRC
00520         PUSH    HL             ;SAVE NEXT BYTE LOCATION
00530         XOR     E              ;XOR REMAINDER LSB W/'A'
00540         LD      C,A            ;SAVE RESULT IN 'C'
00550         LD      B,0            ;ZERO OUT 'B'
00570         ADD     HL,BC          ;ADD BC TO LOCATION
00580         ADD     HL,BC          ;ADD BC TO LOCATION
00590         LD      A,D            ;REMAINDER MSB TO 'A'
00600         XOR     (HL)           ;XOR WITH TABLE VALUE
00610         LD      E,A            ;SAVE RESULT IN 'E'
00620         INC     HL             ;NEXT TABLE LOCATION
00630         LD      D,(HL)         ;SAVE VALUE IN 'D'
00640         POP     HL             ;NEXT BYTE TO CRC MEM
00650         POP     BC             ;NUMBER BYTES TO CRC
00660         DEC     BC             ;LESS ONE
00670         LD      A,B            ;TEST FOR
00680         OR      C              ;ZERO
```

```
00690          JP     NZ,CRCT          ;IF NOT, CRC NEXT ONE
00700          RET                     ;ELSE ALL DONE. RETURN
00710 FINCRC   LD     A,E              ;DE = CRC 2 BYTE VALUE
00720          CPL                     ;COMPLEMENT IT
00730          LD     HL,(WHER4B)      ;END OF MESSAGE +1
00740          LD     (HL),A           ;LD 1ST CRC ON MESSAGE
00750          LD     (ENDCRC+1),A     ;AND SAVE IT HERE
00760          INC    HL               ;NEXT MESSAGE LOCATION
00770          LD     A,D              ;SECOND CRC BYTE
00780          CPL                     ;COMPLEMENT IT
00790          LD     (HL),A           ;LD 2ND CRC ON MESSAGE
00800          LD     (ENDCRC),A       ;AND SAVE IT HERE
00810          RET                     ;RETURN WHENCE U CAME +1
00820
00830 ; FOLLOWING IS RECEIVE CRC CHECK FOR EACH FRAME. IT IS
00840 ; CALLED WITH 'BC' REGISTER ALREADY HAVING THE TOTAL
00850 ; NUMBER OF BYTES IN THE FRAME (INCLUDING CRC BYTES).
00860
00870  RCRC    LD     DE,65535         ;RECEIVE CRC CHECK
00880          LD     HL, (BGINIT)     ;BEGIN FRAME LOCATION
00890          CALL   CRCT             ;CRC ALL INCLUDING CRC
00900          LD     (CRCVAL),DE      ;SAVE REMAINDER IN MEM
00910          LD     HL,61624         ;COMPARE REMAINDER WITH
00920          RST    18H              ;61624 DECIMAL
00930          JP     NZ,BADCRC        ;NOT ZERO = BAD ONE
00940          RET                     ; OK, SO RETURN
00950 BADCRC   POP    AF               ;ADJUST STACK
00960          POP    AF               ;FOR 2 CALLS
00970          LD     IY,37692         ;<BAD CRC> MESSAGE
00980          CALL   SHOWIT           ;DISPLAY ON VIDEO
00990          JP     MODE1A           ;GO AWAIT NEXT PACKET
```

------- CRC LOOKUP TABLE ------->

FIGURE 4 CONTINUED

- -

This is the 512 byte CRC lookup table printed out as 256 two byte words to save space. The label TABLE is at location 1.

```
  1 DEFW     0
  2 DEFW  4489
  3 DEFW  8978
  4 DEFW 12955
  5 DEFW 17956
  6 DEFW 22445
  7 DEFW 25910
  8 DEFW 29887
  9 DEFW 35912
 10 DEFW 40385
 11 DEFW 44890
 12 DEFW 48851
 13 DEFW 51820
 14 DEFW 56293
 15 DEFW 59774
 16 DEFW 63735
 17 DEFW  4225
 18 DEFW   264
 19 DEFW 13203
 20 DEFW  8730
 21 DEFW 22181
 22 DEFW 18220
 23 DEFW 30135
 24 DEFW 25662
 25 DEFW 40137
 26 DEFW 36160
 27 DEFW 49115
 28 DEFW 44626
 29 DEFW 56045
 30 DEFW 52068
 31 DEFW 63999
 32 DEFW 59510
 33 DEFW  8450
 34 DEFW 12427
 35 DEFW   528
 36 DEFW  5017
 37 DEFW 26406
 38 DEFW 30383
 39 DEFW 17460
 40 DEFW 21949
 41 DEFW 44362
 42 DEFW 48323
 43 DEFW 36440
 44 DEFW 40913
 45 DEFW 60270
 46 DEFW 64231
 47 DEFW 51324
 48 DEFW 55797
 49 DEFW 12675
 50 DEFW  8202
 51 DEFW  4753
 52 DEFW   792
 53 DEFW 30631
 54 DEFW 26158
 55 DEFW 21685
 56 DEFW 17724
 57 DEFW 48587
 58 DEFW 44098
 59 DEFW 40665
 60 DEFW 36688
 61 DEFW 64495
 62 DEFW 60006
 63 DEFW 55549
 64 DEFW 51572
 65 DEFW 16900
 66 DEFW 21389
 67 DEFW 24854
 68 DEFW 28831
 69 DEFW  1056
 70 DEFW  5545
 71 DEFW 10034
 72 DEFW 14011
 73 DEFW 52812
 74 DEFW 57285
 75 DEFW 60766
 76 DEFW 64727
 77 DEFW 34920
 78 DEFW 39393
 79 DEFW 43898
 80 DEFW 47859
 81 DEFW 21125
 82 DEFW 17164
 83 DEFW 29079
 84 DEFW 24606
 85 DEFW  5281
 86 DEFW  1320
 87 DEFW 14259
 88 DEFW  9786
 89 DEFW 57037
 90 DEFW 53060
 91 DEFW 64991
 92 DEFW 60502
 93 DEFW 39145
 94 DEFW 35168
 95 DEFW 48123
 96 DEFW 43634
 97 DEFW 25350
 98 DEFW 29327
 99 DEFW 16404
100 DEFW 20893
101 DEFW  9506
102 DCFW 13483
103 DEFW  1584
104 DEFW  6073
105 DEFW 61262
106 DEFW 65223
107 DEFW 52316
108 DEFW 56789
109 DEFW 43370
110 DEFW 47331
111 DEFW 35448
112 DEFW 39921
113 DEFW 29575
114 DEFW 25102
115 DEFW 20629
116 DEFW 16668
117 DEFW 13731
118 DEFW  9258
119 DEFW  5809
120 DEFW  1848
121 DEFW 65487
122 DEFW 60998
123 DEFW 56541
124 DEFW 52564
125 DEFW 47595
126 DEFW 43106
127 DEFW 39673
128 DEFW 35696
129 DEFW 33800
130 DEFW 38273
131 DEFW 42778
132 DEFW 46739
133 DEFW 49708
134 DEFW 54181
135 DEFW 57662
136 DEFW 61623
137 DEFW  2112
138 DEFW  6601
139 DEFW 11090
140 DEFW 15067
141 DEFW 20068
142 DEFW 24557
143 DEFW 28022
144 DEFW 31999
145 DEFW 38025
146 DEFW 34048
147 DEFW 47003
148 DEFW 42514
149 DEFW 53933
150 DEFW 49956
151 DEFW 61887
152 DEFW 57398
153 DEFW  6337
154 DEFW  2376
155 DEFW 15315
156 DEFW 10842
157 DEFW 24293
158 DEFW 20332
159 DEFW 32247
160 DEFW 27774
167 DEFW 42250
162 DEFW 46211
163 DEFW 34328
164 DEFW 38801
165 DEFW 58158
166 DEFW 62119
167 DEFW 49212
168 DEFW 53685
169 DEFW 10562
170 DEFW 14539
171 DEFW  2640
172 DEFW  7129
173 DEFW 28518
174 DEFW 32495
175 DEFW 19572
176 DEFW 24061
177 DEFW 46475
178 DEFW 41986
179 DEFW 38553
180 DEFW 34576
181 DEFW 62383
182 DEFW 57894
183 DEFW 53437
184 DEFW 49460
185 DEFW 14787
186 DEFW 10314
187 DEFW  6865
188 DEFW  2904
189 DEFW 32743
190 DEFW 28270
191 DEFW 23797
192 DEFW 19836
193 DEFW 50700
194 DEFW 55173
195 DEFW 58654
196 DEFW 62615
197 DEW  32808
198 DEFW 37281
199 DEFW 41786
200 DEFW 45747
201 DEFW 19012
202 DEFW 23501
203 DEFW 26966
204 DEFW 30943
205 DEFW  3168
206 DEFW  7657
207 DEFW 12146
208 DEFW 16123
209 DEFW 54925
210 DEFW 50948
211 DEFW 62879
212 DEFW 58390
213 DEFW 37033
214 DEFW 33056
215 DEFM 46011
216 DEFW 41522
217 DEFW 23237
218 DEFW 19276
219 DEFW 31191
220 DEFW 26718
221 DEFW  7393
222 DEFW  3432
223 DEFW 16371
224 DEFW 11898
225 DEFW 59150
226 DEFW 63111
227 DEFW 50204
228 DEFW 54677
229 DEFW 41258
230 DEFW 45219
231 DEFW 33336
232 DEFW 37809
233 DEFW 27462
234 DEFW 31439
235 DEFW 18516
236 DEFW 23035
237 DEFW 11618
238 DEFW 15595
239 DEFW  3696
240 DEFW  8185
241 DEFW 63375
242 DEFW 58886
243 DEFW 54429
244 DEFW 50352
245 DEFW 45483
246 DEFW 40993
247 DEFW 37561
248 DEFW 33584
249 DEFW 31687
250 DEFW 27214
251 DEFW 22741
252 DEFW 18780
253 DEFW 15843
254 DEFW 11370
255 DEFW  7921
256 DEFW  3960
```

```
00100 ;                  FIGURE 5
00110
00120 ; TRANSMIT SUBROUTINE:  SINGLE OR MULTI-FRAME 1200 BAUD
00130
00140 ; PACKET WITH REAL-TIME ZERO INSERTION WHERE APPLICABLE
00150
00160
00170 SENPAK  EXX                        ;SWAP ALTERNATE REGISTERS
00180         LD      HL,(NORMFM)        ;NORMAL FRAME LENGTH
00190         LD      DE,(LASTFM)        ;LAST FRAME LAST PACK LEN
00200         LD      A,(FRMCNT)         ;FRAMES PER PACKET
00210         LD      B,A                ;SAVE IN ALTERNATE 'B'
00220         LD      A,(TESCNT)         ;LAS FRM LAS PACK POINTER
00230         LD      C,A                ;SAVE IN ALTERNATE 'C'
00240         EXX                        ;RESTORE REG. REGISTERS
00250         LD      IY,(STARPK)        ;ASSEMBLED PACK BEGIN ADR
00260         LD      A,1                ;LAST BIT VALUE POINTER
00270         LD      (LASONE),A         ;SAVE IT IN LASONE
00280         ED      (SIGN6),A          ;SET XMIT LO-MEM POINTER
00290         XOR     A                  ;ZERO OUT
00300         LD      (ZEROMK),A         ;MARKS IN A ROW COUNTER
00310         LD      (ZEROSP),A         ;SPACES IN A ROW COUNTER
00320 FLGDLY  LD      A,(BK)             ;BACKOFF DELAY 'ON' ?
00330         CP      1                  ;IF SO, DO RANDOM
00340         CALL    Z,BAKOFF           ;BACKOFF AFTER CLEAR
00350 FLGNUM  LD      A,60               ;NUMBER FLAGS YOU INPUT
00360 FLG     DEC     A                  ;MINUS 1
00370         JP      Z,SEND7            ;IF DONE SEND DATA IN 470
00380         PUSH    AF                 ;NUMBER FLAGS REMAINING
00390         CALL    FLAG               ;SEND SDLC/HDLC FLAG
00400         POP     AF                 ;RESTORE FLAG COUNTER
00410         JP      FLG                ;DO NEXT ONE
00420 FLAG    LD      HL,98              ;1200 BAUD COUNT NUMBER
00430         LD      (SPEED),HL         ;STASH IT IN SPEED
00440         LD      A,126              ;FLAG BYTE VALUE
00450         CALL    SN1A               ;NO ZERO INSERT TRANSMIT
00460         RET                        ;RETURN TO LINE 400
00470 SEND7   EXX                        ;SWAP ALTERNATE REGISTERS
00480         PUSH    HL                 ;FRAME LENGTH TO STACK
00490         EXX                        ;RESTORE REG. REGISTERS
00500         POP     DE                 ;FRAME LENGTH TO 'DE'
00510         DEC     DE                 ;DECREMENT FRAME LENGTH
00520         LD      A,D                ;TEST
00530         OR      E                  ;FOR ZERO
00540         JP      Z,KYBD4            ;IF ZERO, GOTO LINE 600
00550         PUSH    DE                 ;FRAME LENGTH ON STACK
00560         LD      A,(IY)             ;BYTE TO TRANSMIT
00570         INC     IY                 ;NEXT BYTE LOCATION
00580         CALL    SN1                ;ZERO INSERTION TRANSMIT
00590         JP      SEND7+3            ;GOTO LINE 500
00600 KYBD4   CALL    FLAG               ;XMIT FRAME CLOSING FLAG
00610         EXX                        ; SWAP ALTERNATE REGISTERS
00620         DEC     C                  ;LAST FRAME LAST PACK ?
00630         JP      Z,KYBD4A           ;IF ZERO JUMP TO LINE 670
00640         DEC     B                  ; DEC NORMAL FRAMES/PACK
00650         JP      NZ,SEND7+1         ;NOT ZERO, GOTO LINE 480
00660         JP      DUN1               ;IF ZERO, GOTO LINE 690
00670 KYBD4A  PUSH    DE                 ;LAST FRAME LAST PACK LEN
00680         JP      SEND7+2            ;GOTO LINE 490

00690 DUN1    XOR     A                  ;SWITCH T/R RELAY
00700         OUT     (0),A              ;TO RECEIVE
00710         EXX                        ;RESTORE REG. REGISTERS
00720         JP      MODE1              ;GOTO RECEIVE MODE
00730 SN1     LD      D,A                ;BYTE VALUE TO TRANSMIT
00740         LD      E,8                ;NUMBER OF BITS PER BYTE
00750 SN2     LD      A,(LASONE)         ;1 = SPACE & 5 = MARK
00760         CP      1                  ;WAS IT A SPACE ?
00770         JP      Z,LASTSP           ;IF SO, GOTO LAST SPACE
00780         BIT     0,D                ;SET Z FLAG FOR BIT ZERO
00790         CALL    NZ,MARK            ;IF NOT ZERO SEND MARK
00800         BIT     0,D                ;SET Z FLAG FOR BIT ZERO
00810         CALL    Z,SPACE            ;IF ZERO SEND SPACE
00820         NOP                        ;2 USEC TIMING ADJUST
00830         DEC     E                  ;-1 FROM BIT COUNTER
00840         RET     Z                  ;IF ZERO, RETURN LINE 590
00850         RRC     D                  ;RIGHT SHIFT ALL 1 BIT
00860         JP      SN2                ;GO BACK FOR NEXT BIT
00870 LASTSP  BIT     0,D                ;SET Z FLAG FOR BIT ZERO
00880         CALL    NZ,SPACE           ;IF NOT ZERO SEND SPACE
00890         BIT     0,D                ;SET Z FLAG FOR BIT ZERO
00900         CALL    Z,MARK             ;IF ZERO SEND MARK
00910         NOP                        ;2 USEC TIMING ADJUST
00920         DEC     E                  ;-1 FROM BIT COUNTER
00930         RET     Z                  ;IF ZERO, RETURN LINE 590
00940         RRC     D                  ;RIGHT SHIFT ALL 1 BIT
00950         JP      SN2                ;GO BACK FOR NEXT BIT
00960 SPACE   LD      A,5                ;SEND SPACE TONE
00970         OUT     (0),A              ;VIA PORT ZERO
00980         XOR     A                  ;ZERO OUT 'A' REGISTER
00990         LD      (ZEROMK),A         ;AND ZERO MARK COUNTER
01000         LD      A,(SPEED)          ;COUNTDOWN VALUE
01010         LB      HL,SPACEA          ;RETURN MEM LOCATION
01020         PUSH    HL                 ;PUSH ON TOP OF STACK
01030         LD      HL,DECSP           ;JP (HL) ADDRESS
01040 DECSP   DEC     A                  ;-1 COUNTDOWN VALUE
01050         RET     Z                  ;GOTO SPACEA WHEN ZERO
01060         JP      (HL)               ;JUMP TO DECSP
01070 SPACEA  LD      A,(LASONE)         ;PREVIOUS BIT SENT
01080         CP      5                  ;WAS IT A MARK ?
01090         JP      Z,SPACEB           ;IF SO, DON'T COUNT IT
01100         LD      A,(ZEROSP)         ;SPACE COUNTER STASH
01110         INC     A                  ;+1 TO SPACE COUNTER
01120         CP      5                  ;5 SPACES IN A ROW ?
01130         JP      Z,SPACEC           ;IF SO, DO ZERO INSERTION
01140         LD      (ZEROSP),A         ; IF NOT, SAVE NEW VALUE
01150         NOP                        ;2 USEC TIMING ADJUST
01160         RET                        ;RETURN WHENCE U CAME +1
01170 SPACEB  LD      A,1                ;SINCE NOT SAME CHANGE IT
01180         LD      (LASONE),A         ;UPDATE LASTONE
01190         NOP                        ;EQUALIZING DELAY
01200         NOP                        ;EQUALIZING DELAY
01210         NOP                        ;EQUALIZING DELAY
01220         RET                        ;RETURN WHENCE U CAME +1
01230 SPACEC  LD      A,1                ;1 = SPACE & 5 = MARK
01240         LD      (LASONE),A         ;UPDATE LASTONE
01250         LD      BC,1               ;DELAY - NO SN2 ITERATION
01260         CALL    060H               ;APPROX.  30  MICROSECONDS
01270         CALL    MARK               ;DO ZERO INSERTION
```

```
01280         XOR     A             ;ZERO OUT 'A' REGISTER
01290         LD      (ZEROMK),A    ;AND ZERO MARK COUNTER
01300         RET                   ;RETURN WHENCE U CAME +1
01310 SPACE1  LD      A,5           ;1310-1410 ONLY FOR FLAG
01320         OUT     (0),A         ;SPACE TONE PORT ZERO
01330         LD      A,1           ;1 = SPACE & 5 = MARK
01340         LD      (LASONE),A    ;UPDATE LASTONE
01350         XOR     A             ;ZERO OUT 'A' REGISTER
01360         LD      (ZEROMK),A    ;AND ZERO MARK COUNTER
01370         LD      A,(SPEED)     ;COUNTDOWN VALUE
01380         LD      HL,DECSP1     ;JP (HL) ADDRESS
01390 DECSP1  DEC     A             ;-1 COUNTDOWN VALUE
01400         RET     Z             ;RETURN WHENCE U CAME +1
01410         JP      (HL)          ;JUMP TO DECSP1
01420 MARK    LD      A,1           ;SEND MARK TONE
01430         OUT     (0),A         ;VIA PORT ZERO
01440         XOR     A             ;ZERO OUT 'A' REGISTER
01450         LD      (ZEROSP),A    ;AND ZERO SPACE COUNTER
01460         LD      A,(SPEED)     ;COUNTDOWN VALUE
01470         LD      HL,MARKA      ;RETURN MEM LOCATION
01480         PUSH    HL            ;PUSH ON TOP OF STACK
01490         LD      HL,DECMK      ;JP (HL) ADDRESS
01500 DECMK   DEC     A             ;-1 COUNTDOWN VALUE
01510         RET     Z             ;GOTO MARKA WHEN ZERO
01520         JP      (HL)          ;JUMP TO DECMK
01530 MARKA   LD      A,(LASONE)    ;PREVIOUS BIT SENT
01540         CP      1             ;WAS IT A SPACE ?
01550         JP      Z,MARKB       ;IF SO, DON'T COUNT IT
01560         LD      A,(ZEROMK)    ;MARK COUNTER STASH
01570         INC     A             ;+1 TO MARK COUNTER
01580         CP      5             ;5 MARKS IN A ROW ?
01590         JP      Z,MARKC       ;IF SO, DO ZERO INSERTION
01600         LD      (ZEROMK),A    ;IF NOT, SAVE NEW VALUE
01610         NOP                   ;2 USEC TIMING ADJUST
01620         RET                   ;RETURN WHENCE U CAME +1
01630 MARKB   LD      A,5           ;SINCE NOT SAME CHANGE IT
01640         LD      (LASONE),A    ;UPDATE LASTONE
01650         NOP                   ;EQUALIZING DELAY
01660         NOP                   ;EQUALIZING DELAY
01670         NOP                   ;EQUALIZING DELAY
01680         RET                   ;RETURN WHENCE U CAME +1
01690 MARKC   LD      A,5           ;1 = SPACE & 5 = MARK
01700         LD      (LASONE),A    ;UPDATE LASTONE
01710         LD      BC,1          ;DELAY - NO SN2 ITERATION
01720         CALL    060H          ;APPROX. 30 MICROSECONDS
01730         CALL    SPACE         ;DO ZERO INSERTION
01740         XOR     A             ;ZERO OUT 'A' REGISTER
01750         LD      (ZEROSP),A    ;AND ZERO SPACE COUNTER
01760         RET                   ;RETURN WHENCE U CAME +1
01770 MARK1   LD      A,1           ;1770-1870 ONLY FOR FLAG
01780         OUT     (0),A         ;SEND MARK TONE
01790         LD      A,5           ;1 = SPACE & 5 = MARK
01800         LD      (LASONE),A    ;UPDATE L&STONE
01810         XOR     A             ;ZERO OUT 'A' REGISTER
01820         LD      (ZEROSP),A    ;AND ZERO SPACE COUNTER
01830         LD      A,(SPEED)     ;COUNTDOWN VALUE
01840         LD      HL,DECMK1     ;JP (HL) ADDRESS
0 1850 DECMK1 DEC     A             ;-1 COUNTDOWN VALUE
01860         RET     Z             ;RETURN WHENCE U CAME +1
```

```
01870         JP      (HL)          ;JUMP TO DECMK1
01880 SN1A    LD      D,A           ;1880-2080 ONLY FOR FLAG
01890         LD      E,8           ;NUMBER OF BITS PER BYTE
01900 SN2A    LD      A,(LASONE)    ;1 = SPACE & 5 = MARK
01910         CP      1             ;WAS XT A SPACE ?
01920         JP      Z,LASSP       ;IF SO, GOT0 LAST SPACE
01930         BIT     0,D           ;SET Z FLAG FOR BIT ZERO
01940         CALL    NZ,MARK1      ;IF NOT ZERO SEND MARK
01950         BIT     0,D           ;SET Z FLAG FOR BIT ZERO
01960         CALL    Z,SPACE1      ;IF ZERO SEND SPACE
01970         DEC     E             ;-1 FROM BIT COUNTER
01980         RET     Z             ;IF ZERO, RETURN LINE 460
01990         RRC     D             ;RIGHT SHIFT ALL 1 BIT
02000         JP      SN2A          ;GO BACK FOR NEXT BIT
02010 LASSP   BIT     0,D           ;SET Z FLAG FOR BIT ZERO
02020         CALL    NZ,SPACE1     ;IF NOT ZERO SEND SPACE
02030         BIT     0,D           ;SET Z FLAG FOR BIT ZERO
02040         CALL    Z,MARK1       ;IF ZERO SEND MARK
02050         DEC     E             ;-1 FROM BIT COUNTER
02060         RET     Z             ;IF ZERO, RETURN LINE 460
02070         RRC     D             ;RIGHT SHIFT ALL 1 BIT
02080         JP      SN2A          ;GO BACK FOR NEXT BIT
02090 ZEROSP  DEFB    0             ;SPACE COUNTER STASH
02100 ZEROMK  DEFB    0             ;MARK COUNTER STASH
02110 SPEED   DEFW    98            ;XMIT COUNTDOWN VALUE
02120 LASONE  DEFB    1             ;LAST BIT SENT VALUE
02130 ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
02140 ;END OF SINGLE/MULTI-FRAME 1200 BAUD SYNCHRONOUS TRANSMIT
02150
02160 ; FOR MODEL III CLOCK CHANGE LINE 420 FROM 98 TO 115.
```

```
00100 ;                         FIGURE 6
00110
00120 ; IN-PROGRAM DISK I/O SUBROUTINES FOR AX.25 PROTOCOL
00130
00140 ; FOR TRSDOS 1.3 - TRSDOS 2.3 - NEWDOS + AND 1.0
00150
12790          ORG     49632           ;SUBROUTINE MEM LOCATION
12800 FCB      DEFS    32              ;DISK FILE CONTROL BLOCK
12810 BUFFER   DEFS    256             ;DISK I/O WORKING SPACE
12820 DIZ      LD      A,(HL)          ;DISPLAY MESSAGE ON VIDEO
12830          CP      0               ;END OF MESSAGE DELIMITER
12840          JP      Z,FINISH        ; IF ZERO, ALL DONE
12850          CALL    033H            ;DISPLAY BYTE ON VIDEO
12860          INC     HL              ;NEXT MESG BYTE LOCATION
12870          JP      DIZ             ;GO DISPLAY NEXT BYTE
12880 FINISH   RET                     ;RETURN WHENCE U CAME +1
12890 INPNAM   CALL    CLS             ;C L E A R   V I D E O
12900          LD      HL,NAM1         ;REMEMBER DELIMITERS MSG?
12910          CALL    DIZ             ;DISPLAY XT ON VIDEO
12920          CALL    049H            ;AWAIT KEYBOARD INPUT
12930          CP      1               ;BREAK KEY PRESSED ?
12940          JP      Z,ESCAPE        ;IF SO, ESCAPE LINE 13180
12950          CALL    CLS             ;C L E AR   V I D E O
12960          LD      HL,NAM1A        ;INPUT BEGIN ADDRESS MSG?
12970          CALL    DIZ             ;DISPLAY IT ON VIDEO
12980          CALL    1BB3H           ;KEYBOARD INPUT ROUTINE
12990          RST     10H             ;SCAN STRING SET 'C' FLAG
13000          CALL    1E5AH           ;CONVERT UNSIGNED INTEGER
13010          EX      DE,HL           ;PUT INTEGER IN HL
13020          LD      (DUMP+1),HL     ;STUFF BEGIN ADDRESS DUMP
13030          LD      (HOWFAR+1),HL   ;AND IN HOWFAR MEM
13040 INNAME   CALL    CLS             ;C L E AR   V I D E O
13050          LD      HL,NAM2         ;INPUT FILE NAME MESSAGE?
13060          CALL    DIZ             ;DISPLAY IT ON VIDEO
13070          CALL    1BB3H           ;KEYBOARD INPUT ROUTINE
13080          LD      HL,41E8H        ;WHERE STASHED IN MEM
13090          LD      A,(HL)          ;FIRST BYTE OF FILE NAME
13100          CP      0               ;YOU INPUT NOTHING ?
13110          JP      Z,ESCAPE        ;IF SO, ESCAPE LINE 13180
13120          CALL    LONG            ;HOW MANY BYTES IN NAME ?
13130          LD      HL,41E8H        ;NAME ADDRESS IN MEM
13140          LD      DE,FCB          ;FILE CONTROL BLOCK ADR
13150          LDIR                    ;MOVE TO CONTROL BLOCK
13160          CALL    DRIVE           ;AND MOVE DRIVE NO. TOO
13170          RET                     ;RETURN WHENCE U CAME +1
13180 ESCAPE   POP     AF              ;ADJUST STACK FOR CALL
13190          LD      HL,53248        ;RESET TO NORMAL
13200          LD      (DUMP+1),HL     ;DUMP AND
13210          LD      (HOWFAR+1),HL   ;HOWFAR
13220          JP      MENU            ;TIP MENU FOR INSTRUCTS
13230 LONG     LD      BC,0            ;HOW LONG IS FILE NAME ?
13240 LON1     LD      A,(HL)          ;BYTE FROM NAME STRING
13250          CP      0               ;ZERO DELIMITER
13260          RET     Z               ;RETURN WITH COUNT
13270          INC     C               ;1 MORE BYTE
13280          INC     HL              ;NEXT MEM LOCATION
13290          JP      LON1            ;GO COUNT IT
13300 LBYTES   DEFW    0               ;NUMBER BYTES READ STASH
13310 CLRLO    LD      HL,16872        ;CLEAR
13320          LD      DE,16873        ;LOW MEMORY
13330          LD      BC,12878        ;WITH
13340          LD      (HL),0          ;ZEROS
13350          LDIR                    ;DO IT RIGHT NOW
13360          RET                     ;RETURN WHENCE U CAME +1
13370 OPEN1    LD      DE,FCB          ;FILE CTRL BLOCK MEM ADR
13380          LD      HL,BUFFER       ;DISK I/O BUFFER ADDRESS
13390          LD      B,0             ;256 BYTE RECORD LENGTH
13400          CALL    4424H           ;OPEN AN EXISTING FILE
13410          JR      NZ,ERROR        ;Z FLAG SET IF ERROR
13420          RET                     ;RETURN WHENCE U CAME +1
13430 READ     LD      HL,53248        ;WHERE TO PUT FILE IN MEM
13440          LD      DE,FCB          ;FILE CTRL BLOCK ADDRESS
13450 LG       PUSH    HL              ;SAVE MEM LOCATION STACK
13460          CALL    13H             ;READ BYTE FROM DISK FILE
13470          POP     HL              ;RESTORE MEM LOCATION
13480          LD      (HL),A          ;AND LOAD IT IN MEM
13490          INC     HL              ;NEXT MEM LOCATION
13500          PUSH    HL              ;SAVE IT IN STACK
13510          PUSH    DE              ;SAVE FCB POINTER
13520 LONG1    LD      DE,65535        ;FILE END ADDRESS IN MEM
13530          OR      A               ;CLEAR CARRY FLAG
13540          SBC     HL,DE           ;SUB HL - DE SET Z FLAG
13550          POP     DE              ;RESTORE FCB POINTER
13560          POP     HL              ;RESTORE MEM LOCATION
13570          RET     Z               ;RETURN IF ALL DONE
13580          JP      LG              ;GO READ NEXT BYTE
13590 CLOSE    LD      DE,FCB          ;FILE CTRL BLOCK ADDRESS
13600          CALL    4428H           ;CLOSE FILE SUBROUTINE
13610          PUSH    AF              ;SAVE IN STACK
13620          LD      HL,53248        ;BEGIN HI-MEM ADDRESS
13630          LD      (DUMP+1),HL     ;RESET DUMP
13640          LD      (HOWFAR+1),HL   ;RESET HOWFAR
13650          POP     AF              ;RESTORE AF
13660          RET     Z               ;RETURN UNLESS ERROR
13670          POP     HL              ;ADJUST STACK FOR CALL
13680 ERROR    LD      H,0             ;ZERO OUT 'H'
13690          LD      L,A             ;ERROR NUMBER TO 'L'
13700          CALL    0A9AH           ;MOVE HL INTO ACCUM
13710          CALL    0A7FH           ;MAKE SURE AN INTEGER
13720          CALL    OFBDH           ;CONVERT TO STRING
13730          LD      DE,MS2C+9       ;ERROR MESSAGE LOCATION
13740 ER1      LD      A,(HL)          ;ERROR NUMBER
13750          CP      0               ;ZERO STRING DELIMITER
13760          JP      Z,ER2           ;ALL DONE ? GOT0 ER2
13770          LD      (DE),A          ;ERROR NUMBER TO MEM
13780          INC     HL              ;NEXT ERROR # LOCATION
13790          INC     DE              ;NEST MESSAGE LOCATION
13800          JP      ER1             ;GO MOVE NEXT ONE
13810 ER2      CALL    CLS             ;C L E A R   V I D E O
13820          POP     AF              ;ADJUST STACK
13830          CALL    SETUP           ;RESTORE PGM POINTERS
13840          CALL    CLRLO           ;CLEAR OUT DOS
13850          CALL    CLRHY           ;CLEAR OUT HI-MEM
13860          LD      HL,MS2C         ;ERROR # MESSAGE
13870          CALL    DIZ             ;DISPLAY IT ON VIDEO
13880          CALL    049H            ;PRESS ANY KEY
13890          JP      MENU            ;GOTO MENU FOR INSTRUCTS
13900 DRIVE    LD      A,@:'           ;DRIVE # SEPARATOR
```

```
13910          LD      (DE),A          ;FILE CONTROL BLOCK
13920          LD      (BC),A          ;FUTURE USE VOL. 3
13930          INC     DE              ;NEXT FCB LOCATION
13940          INC     Bc              ;FUTURE USE VOL. 3
13950          LD      A,'1'           ;DRIVE# CHANGE UR CHOICE
13960          LD      (DE),A          ;INTO FILE CTRL BLOCK
13970          LD      (BC),A          ;FUTURE USE VOL. 3
13980          INC     DE              ;FCB NEXT LOCATION
13990          INC     Bc              ;FUTURE USE VOL. 3
14000          LD      A,13            ;FCB DELIMITER
14010          LD      (DE),A          ;INTO FILE CTRL BLOCK
14020          LD      (BC),A          ;FUTURE USE VOL. 3
14030          RET                     ;RETURN WHENCE U CAME +i
14040 NAM2     DEFM     'INPUT FILE NAME '
14050          DEFB    0               ;DELIMITER
14060 OPEN3    LD      HL,BUFFER       ;DISK I/O BUFFER ADDRESS
14070          LD      DE,FCB          ;FILE CTRL BLOCK ADDRESS
14080          LD      B,0             ;256 BYTES PER RECORD
14090          LD      C,10H           ;FILE TYPE DOUBTFUL
14100          CALL    4420H           ;OPEN NEW DISK FILE
14110          RET                     ;RETURN WHENCE U CAPE +i
14120 HOWFAR   LD      HL,53248        ;CALCULATE BYTES TO SAVE
14130 FAR1     INC     HL              ;TO DISK FILE
14140          LD      A,(HL)          ;LOOK
14150          CP      128             ;FOR
14160          JP      NZ,FAR1         ;THREE
14170          INC     HL              ;EACH
14180          LD      A,(HL)          ;DECIMAL
14190          CP      128             ;128
14200          JP      NZ,FARi         ;END
14210          INC     HL              ;OF
14220          LD      A,(HL)          ;MESSAGE
14230          CP      128             ;DELIMITERS
14240          JP      NZ,FARi         ;IN A
14250          INC     HL              ;ROW
14260          LD      (SOFAR+i),HL    ;SAVE THEM IN SOFAR
14270          RET                     ;RETURN WHENCE U CAME +1
14280 DUMP     LD      HL,53248        ;BEGIN DATA LOCATION
14290          LD      DE,FCB          ;FILE CTRL BLOCK ADDRESS
14300 DUM1     LD      A,(HL)          ;BYTE TO SAVE ON DISK
14310          PUSH    HL              ;SAVE BYTE MEM LOCATION
14320          CALL    1BH             ;WRITE TO DISK SUBROUTINE
14330          POP     HL              ;RESTORE BYTE LOCATION
14340          JP      NZ,ERROR        ;Z FLAG SET IF ERROR
14350          INC     HL              ;NEXT BYTE LOCATION
14360          PUSH    HL              ;SAVE IT IN STACK
14370          PUSH    DE              ;SAVE FCB POINTER
'I4380 SOFAR   LD      DE,65535        ;LAST MEM BYTE LOCATION
14390          OR      A               ;CLEAR CARRY FLAG
14400          SBC     HL,DE           ;SUBTRACT HL MINUS DE
14410          POP     DE              ;RESTORE FCB POINTER
14420          POP     HL              ;AND NEXT MEM LOCATION
14430          RET     Z               ;RETURN IF ALL DONE
14440          JP      DUM1            ;GO DUMP NEXT ONE TO DISK
14450 NAM1     DEFM     'REMEMBER 128 DELIMITERS ? HIT BREAK TO ES
CAPE ELSE  <ENTER>'
14460          DEFB    0               ;DELIMITER
14470          ORG     0C840H          ;SAVE FILE MEM LOCATION
14480 SVFILE   CALL    INPNAM          ;INPUT FILE NAME
14490          CALL    HOWFAR          ;CALCULATE BYTES TO SAVE
14500          LD      A,195           ;RESTORE JUMP
14510          LD      (400CH),A       ;TO LOW MEMORY
14520          CALL    MOVDN           ;MOVE DOS BACK DOWN MEM
14530          CALL    OPEN3           ;OPEN OR CREATE DISK FILE
14540          CALL    DUMP            ;DUMP IT TO DISK
14550          CALL    'CLOSE          ;CLOSE TME DISK FILE
14560          LD      SP,29758        ;RESET STACK POINTER
14570          CALL    SETUP           ;REINITIALIZE PGM PTRS
14580          CALL    CLRLO           ;CLEAR OUT DOS LO-MEM
14590          JP      MENU            ;GO FOR INSTRUCTIONS
14600 MS2C     DEFM     'ERROR #       ;DISK I/O ERROR MESSAGE
14610          DEFB    0               ;DELIMITER
14620          ORG     0C880H          ;LOAD FILE MEM LOCATION
14630 LDFILE   CALL    INNAME          ;INPUT FILE NAME
14640          CALL    CLRHY           ;CLEAR HI-MEMORY
14650          LD      A,195           ;RESTORE JUMP
14660          LD      (400CH),A       ;TO LOW MEMORY
14670          CALL    MOVDN           ;MOVE DOS BACK DOWN MEL;
14680          CALL    OPEN1           ;OPEN AN EXISTING FILE
14690          CALL    MULPLY          ;CALCULATE FILE LENGTH
14700          CALL    READ            ;LOAD FILE TO HI-MEMORY
14710          LD      (HIHL),HL       ;SAVE HI-MEM END OF FILE
14720          CALL    CLOSE           ;CLOSE DISK FILE
14730          LD      SP,29758        ;RESET STACK POINTER
14740          CALL    SETUP           ;REINITIALIZE PGM PTRS
14750          CALL    CLRLO           ;CLEAR OUT DOS LO-MEM
14760          CALL    BAKUP           ;CHECK EXTRA FILE DATA ?
14770          JP      MENU            ;MENU FOR INSTRUCTIONS
14780 HIHL     DEFW    0               ;END HI-MEM FILE STASH
14790 BAKUP    LD      HL,(HIHL)       ;! ! VOLUME 3 ONLY ! !
14800 BAK1     DEC     HL              ;THIS
14810          LD      A,(HL)          ;SCINTILLATING
14820          CP      28              ;ROUTINE
14830          JP      Z,BAK1          ;TRIES
14840          CP      128             ;TO
14850          JP      Z,TESAGN        ;BACKUP
14860          INC     HL              ;IN HI-MEMORY
14870          LD      (BEFORE),HL     ;TILL IT
14880          RET                     ;IT
14890 TESAGN   DEC     HL              ;FINDS
14900          LD      A,(HL)          ;3 EACH
14910          CP      128             ;128'S IN
14920          JP      NZ,BAK1         ;A ROW.
14930          DEC     HL              ;WHEN FOUND
14940          LD      A,(HL)          ;IT RESETS BEFOR.
14950          CP      128             ;SO THAT WHEN THE PROGRAM
14960          JP      NZ,BAK1         ;AUTOMATICALLY MOVES THE
14970          LD      (BEFORE),HL     ;FILE DOWN, ONLY DATA IS
14980          RET                     ;MOVED. ! VOLUME 3 ONLY !
14990 NAM1A    DEFM     'INPUT BEGINNING MEM ADDRESS (53248 NOMINA
L) '
15000          DEFB    0               ;DELIMTER
15010 MULPLY   LD      A,(FCB+12)      ;NUMBER RECORDS IN FILE
15020 MUL0     LD      HL,0            ;ZERO OUT BYTE COUNTER
15030 MUL1     LD      DE,256          ;BYTES PER RECORD
15040          ADD     HL,DE           ;ADD THEM UP
15050          DEC     A               ;MINUS ONE RECORD
15060          JP      Z,MUL2          ;ALL DONE, GOTO MUL2
```

```
        JP      MUL1            ;ADD UP NEXT ONE
MUL2    LD      A,(FCB+8)       ;BYTES IN LAST SECTOR
        LD      E,A             ;STUFF IN 'E'
        LD      D,0             ;ZERO OUT 'D'
        ADD     HL,DE           ;ADD THEM UP
MUL3    LD      (LBYTES),HL     ;AND SAVE THEM HERE
        LD      DE,53248        ;BEGIN HIGH MEMORY
        ADD     HL,DE           ;ADD BYTES TO HI-MEM
        LD      (LONG1+1),HL    ;AND SAVE THEM HERE
        RET                     ;RETURN WHENCE U CAME +1

; - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; END OF VOLUME 2 - DISK I/O SUBROUTINES
```

```
;                               FIGURE 7

; IN-PROGRAM EDIT/MODIFY/MONITOR SUBROUTINE - 866 BYTES

; ALSO USED FOR KEYBOARD INPUT PACKET MESSAGES


        ORG     38912           ;SUBROUTINE MEM LOCATION
DISMEM  LD      HL,40960        ;CURRENT PACK LOCATION
DISEM1  LD      (MEMO),HL       ;TOP OF PAGE STASH
DISPLA  LD      HL,(MEMO)       ;BACK TO HL REGISTER
        LD      (LASMEM),HL     ;INC/DEC STASH
        DEC     HL              ;MINUS ONE
        LD      (MEMO1),HL      ;BOTTOM PREVIOUS PAGE
        INC     HL              ;TOP OF THIS PAGE OF MEM
        LD      DE,15360        ;BEGINNING VIDEO MEMORY
        LD      BC,1024         ;BYTES PER PAGE OF VIDEO
AGAIN   LD      A,(HL)          ;CHANGE MODEL III
        BIT     7,A             ;VIDEO DISPLAY
        CALL    Z,SET6          ;TO SIMILAR TO
        BIT     7,A             ;MODEL I
        CALL    NZ,RES6         ;VIDEO DISPLAY
        LD      (DE),A          ;STASH BYTE IN VIDEO
        INC     HL              ;NEXT BYTE FROM MEMORY
        INC     DE              ;NEXT VIDEO DISPLAY MEM
        DEC     Bc              ;BYTES TO MOVE COUNTER
        LD      A,B             ;TEST B
        CP      0               ;IF ZERO
        JP      Z,TESTIT        ;TEST C
        JP      AGAIN           ;ELSE MOVE NEXT BYTE
RES6    RES     6,A             ;ZERO OUT BIT 6
        RET                     ;RETURN WHENCE U CAME +1
SET6    BIT     6,A             ;TEST BIT 6
        RET     NZ              ;RETURN IF SET TO 1
        BIT     5,A             ;TEST BIT 5
        RET     NZ              ;RETURN IF SET TO 1
        SET     6,A             ;IF NOT, SET BIT 6 TO 1
        RET                     ;RETURN WHENCE U CAME +1
TESTIT  LD      A,C             ;BYTES TO MOVE COUNTER
        CP      0               ;ZERO ?
        JP      NZ,AGAIN        ;IF NOT, MOVE NEXT ONE
        LD      (MEMO), HL      ;TOP NEXT PAGE MEMORY
NEXT    CALL    049H            ;AWAIT KEYBOARD INPUT
        CP      1               ;BREAK KEY ?
        JP      Z,7630H         ;IF SO, GOTO FTP MENU
        CP      13              ;ENTER KEY ?
        JP      Z,DISPLA        ;IF SO, DISPLAY NEXT PAGE
        CP      45              ;MINUS KEY ?
        JP      Z,BACKUP        ;IF SO DISPLAY LOWER PAGE
        CP      77              ;'M' KEY PRESSED ?
        JP      Z,MODIF         ;IF SO, GOTO MODIFY MODE
        JP      NEXT            ;ELSE IGNORE IT
BACKUP  LD      HL,(MEMO1)      ;MOVE THE
        INC     HL              ;VIDEO DISPLAY
        LD      (MEMO),HL       ;DOWN A FULL PAGE
        DEC     HL              ;IN MEMORY
        LD      DE,16383        ;LAST BYTE VIDFO MEMORY
        LD      BC,1024         ;FULL PAGE VIDEO BYTES
AGAIN1  LD      A, (HL)         ;CHANGE MODEL III
```

```
05750          BIT     7,A           ;VIDEO DISPLAY
05760          CALL    Z,SET6        ;TO SIMILAR TO
05770          BIT     7,A           ;MODEL I
05780          CALL    NZ,RES6       ;VIDEO DISPLAY
05790          LD      (DE),A        ;STASH BYTE IN VIDEO
05800          DEC     HL            ;NEXT LOWER BYTE MEMORY
05810          DEC     DE            ;NEXT LOWER BYTE VIDEO
05820          DEC     BC            ;DECREMENT BYTE COUNTER
05830          LD      A,B           ;TEST B
05840          CP      0             ;IF ZERO
05850          JP      Z,TESIT       ;TEST C
05860          JP      AGAIN1        ;ELSE MOVE NEXT BYTE
05870 TESIT    LD      A,C           ;TEST C
05880          CP      0             ;FOR ZERO
05890          JP      NZ,AGAIN1     ;IF NOT, MOVE NEXT BYTE
05900          LD      (MEMO1),HL    ;BOTTOM NEXT PAGE DOWN
05910          INC     HL            ;TOP THIS PAGE OF MEM
05920          LD      (LASMEM),HL   ;AND SAVE THIS LOCATION
05930          JP      NEXT          ;GO AWAIT NEXT COMMAND
05940 LASMEM   DEFW    0             ;MEM STASH
05950 MODIF    LD      IX, 15360     ;MODIFY MODE - MODIFY
05960          LD      IY,(LASMEM)   ;BOTH VIDEO & REAL MEMORY
05970 CONT3    CALL    BLINK-9       ;BLINKING CURSOR
05980          LD      A,(LNFEED)    ;LINEFEED AFTER CARRET?
05990          CP      1             ;IF SO
06000          JP      Z,LFEED2      ;STUFF IT IN MEMORY
06010          CALL    BLINKB        ;RESTORE MEM CHARACTER
06020          LD      A, (14400)    ;KEYBOARD ROW PSUEDO MEM
06030          CP      4             ;BREAK KEY PRESSED ?
06040          JP      Z,NEXT2       ;IF SO, RESUME EDIT MODE
06050          CP      32            ;LEFT ARROW KEY PRESSED ?
06060          JP      Z,LEFT1       ;MOVE CURSOR BACK A SPACE
06070          CP      64            ;RIGHT ARROW KEY PRESSED?
06080          JP      Z,RIGHT1      ;MOVE CURSOR AHEAD SPACE
06090          CP      16            ;DOWN ARROW KEY PRESSED 3
06100          JP      Z,DOWN1       ;MOVE CURSOR DOWN 1 LINE
06110          CP      8             ;UP ARROW KEY PRESSED ?
06120          JP      Z,UPONE       ;MOVE CURSOR UP 1 LINE
06130          LD      A, (14464)    ;SHIFT KEY PSUEDO MEM
06140          CP      0             ;EITHER SHIFTKEY PRESSED?
06150          JP      NZ,NOTASC     ;IF SO, TEST NOT ASCII
06160 CONT3B   CALL    02BH          ;KEYBOARD TO 'A'
06170          CP      11            ;SUBTRACT 11
06180          JP      M,CONT3       ;IF MINUS, IGNORE IT
06190          CP      13            ;ENTER KEY ?
06200          CALL    Z,LFEED1      ;SETUP AUTO LINE FEED
06210          CP      32            ;SPACE ?
06220          JP      Z,CK          ;TEST ILLEGAL SHIFT
06230          CP      64            ;@ KEY 3
06240          JP      Z,CONT3       ;IF SO, IGNORE IT
06250          CP      91            ;UP ARROW ?
06260          JP      Z,CONT3       ;IF SO, IGNORE IT
06270          CP      96            ;SHIFT @ ?
06280          JP      Z,CONT3       ;IF SO, IGNORE IT
06290          LD      (HOLDZ),A     ;SAVE BYTE INPUT
06300          LD      A,(UPSIDE)    ;TEST FOR LOWERCASE
06310          CP      0             ;IF SO
06320          JP      NZ,INVERT     ;INVERT IT
06330          LD      A,(HOLDZ)     ;RESTORE BYTE INPUT
06340          CP      65            ;SUBTRACT 65
06350          JP      M,CONT5       ;MINUS JUMP AROUND RESET
06360          RES     5,A           ;RESET BIT 5
06370 CONT5    LD      (IX),A        ;DISPLAY BYTE ON VIDEO
06380          LD      (IY),A        ;LOAD BYTE INTO RAM MEM
06390          CALL    CKAHED        ;NEXT LOCATION IN BOUNDS?
06400          INC     IX            ;OK SO, INCREMENT VIDEO
06410          INC     IY            ;AND MEMORY LOCATION
06420          JP      CONT3         ;GO SCAN FOR NEXT INPUT
06430 LFEED1   PUSH    AF            ;SAVE CARRET BYTE
06440          LD      A,1           ;STUFF 1 INTO
06450          LD      (LNFEED),A    ;AUTO LINE FEED POINTER
06460          POP     AF            ;RESTORE CARRIAGE RETURN
06470          RET                   ;RETURN WHENCE U CAME +1
06480 LFEED2   XOR     A             ;ZERO OUT
06490          LD      (LNFEED),A    ;LINEFEED POINTER
06500          LD      A,10          ;ASCII 10 = LINEFEED
06510          JP      CONT5         ;GO STUFF IT IN MEMORY
06520 LNFEED   DEFB    0             ;LINEFEED POINTER STASH
06530 LEFT1    CALL    SLOWLY        ;SLOWDOWN CURSOR MOVEMENT
06540          CALL    CKBACK        ;CHECK IN BOUNDS ?
06550          DEC     IX            ;OK, MOVE BACK A SPACE
06560          DEC     IY            ;AND DOWN 1 MEM LOCATION
06570          JP      CONT3         ;GO SCAN NEXT INPUT
06580 RIGHT1   CALL    SLOWLY        ;SLOWDOWN CURSOR MOVEMENT
06590          CALL    CKAHED        ;CHECK IN BOUNDS ?
06600          INC     IX            ;OK, MOVE AHEAD A SPACE
06610          INC     IY            ;AND UP 1 MEM LOCATION
06620          JP      CONT3         ;GO SCAN NEXT INPUT
06630 UPONE    CALL    SLOWLY        ;SLOWDOWN CURSOR MOVEMENT
06640          CALL    SLOWLY        ;SLOWDOWN CURSOR MOVEMENT
06650          CALL    CKDOWN        ;CHECK IN BOUNDS ?
06660          CALL    SUB64         ;OK, SO MOVE UP A LINE
06670          JP      CONT3         ;GO SCAN NEXT INPUT
06680 DOWN1    CALL    SLOWLY        ;SLOWDOWN CURSOR MOVEMENT
06690          CALL    SLOWLY        ;SLOWDOWN CURSOR MOVEMENT
06700          CALL    CKUP          ;CHECK IN BOUNDS ?
06710          CALL    ADD64         ;OK, SO MOVE DOWN A LINE
06720          JP      CONT3         ;GO SCAN NEXT INPUT
06730 CONT3A   POP     HL            ;ADJUST STACK
06740          JP      CONT3         ;GO SCAN NEXT INPUT
06750 CKBACK   LD      DE, 15360     ;BEGIN VIDEO MEMORY
06760          PUSH    IX            ;SWAP IX
06770          POP     HL            ;INTO HL
06780          CALL    0A39H         ;COMPARE HL MINUS DE
06790          JP      Z,CONT3A      ;IF EQUAL, THEN IGNORE
06800          RET                   ;ELSE OK.  RETURN
06810 CKAHED   LD      DE, 16383     ;END OF VIDEO MEMORY
06820          PUSH    IX            ;SWAP IX
06830          POP     HL            ;INTO HL
06840          CALL    0A39H         ;COMPARE HL MINUS DE
06850          JP      Z,CONT3A      ;IF EQUAL, THEN IGNORE
06860          RET                   ;ELSE OK.  RETURN
06870 CKDOWN   CALL    SUB64A        ;-64 FROM VIDEO MEM
06880          LD      DE,15360      ;BEGIN VIDEO MEM
06890          CALL    0A39H         ;COMPARE HL - DE
06900          JP      C,CONT3A      ;IF OUT OF BOUNDS, IGNORE
06910          RET                   ;ELSE OK.  RETURN
06920 CKUP     CALL    ADD64A        ;+64 TO VIDEO MEM
```

```
06930          LD      DE,16384        ;END VIDEO MEM
06940          CALL    0A39H           ;COMPARE HL - DE
06950          JP      NC,CONT3A       ;IF OUT OF BOUNDS, IGNORE
06960          RET                     ;ELSE OK.  RETURN
06970 SUB64A   PUSH    IX              ;SWAP IX
06980          POP     HL              ;INTO HL
06990          LD      A,64            ;WE COULD HAVE
07000 AGN64S   DEC     HL              ;USED ADD HL,DE
07010          DEC     A               ;BUT THERE IS MORE
07020          RET     Z               ;THAN ONE WAY TO
07030          JP      AGN64S          ;SKIN A CAT
07040 ADD64A   PUSH    IX              ;SWAP IX
07050          POP     HL              ;INTO HL
07060          LD      A,64            ;WE COULD HAVE
07070 AGN64A   INC     HL              ;USED SBC HL,DE
07080          DEC     A               ;BUT THERE IS MORE
07090          RET     Z               ;THAN ONE WAY TO
07100          JP      AGN64A          ;SKIN A CAT
07110 SUB64    LD      A,64            ;HERE IS ANOTHER
07120 AGNSUB   DEC     IX              ;PLACE YOU MIGHT
07130          DEC     IY              ;WISH TO USE
07140          DEC     A               ;SBC HL,DE
07150          RET     Z               ;HOW MANY BYTES
07160          JP      AGNSUB          ;DID IT SAVE ?
07170 ADD64    LD      A,64            ;HERE IS ANOTHER
07180 AGNADD   INC     IX              ;PLACE YOU MIGHT
07190          INC     IY              ;WISH TO USE
07200          DEC     A               ;ADD HL,DE
07210          RET     Z               ;HOW MANY BYTES
07220          JP      AGNADD          ;DID IT SAVE ?
07230 HOLDIT   DEFW    0               ;HOLDIT STASH
07240 SLOWLY   CALL    BLINKA          ;S   C   M
07250          CALL    BLINKB          ;L    u    0
07260          CALL    BLINKA          ;  0   R   V
07270          CALL    BLINKB          ;   W   S   E
07280          CALL    BLINKA          ;    D   0   M
07290          CALL    BLINKB          ;     0   R   E
07300          CALL    BLINKA          ;      W       N
07310          CALL    BLINKB          ;       N       T
07320          RET                     ;RETURN WHENCE U CAME +1
07330 BLINKA   LD      A,(IX)          ;SAVE VIDEO BYTE
07340          LD      (HOLDIT),A      ;IN HOLDIT
07350          LD      A,143           ;RECTANGULAR CURSOR
07360          LD      (IX),A          ;DISPLAY ON VIDEO
07370          LD      BC,600          ;1/100TH SECOND
07380          CALL    060H            ;TIME DELAY
07390          RET                     ;RETURN WHENCE U CAME +1
07400 BLINKB   LD      A,(HOLDIT)      ;RESTORE VIDEO CHARACTER
07410          LD      (IX),A          ;TO VIDEO MEM LOCATION
07420          LD      BC,600          ;1/100TH SECOND
07430          CALL    060H            ;TIME DELAY
07440          RET                     ;RETURN WHENCE U CAME +1
07450 INVERT   LD      A,(HOLDZ)       ;INVERT UPPER/LOWER CASE
07460          CP      65              ;SUBTRACT 65
07470          JP      M,CONT5         ;NOT ALPHABETICAL IGNORE
07480          CP      123             ;SUBTRACT 123
07490          JP      P,CONT5         ;NOT ALPHABETICAL IGNORE
07500          CP      95              ;SUBTRACT 95
07510          JP      Z,CONT5         ;NOT ALPHABETICAL IGNORE
07520          BIT     5,A             ;BIT 5 SET ?
07530          JP      Z,SET5A         ;IF NOT, THEN SET XT
07540          RES     5,A             ;ELSE RESET IT
07550          JP      CONT5           ;AND DISPLAY IT
07560 SET5A    SET     5,A             ;SET BIT 5 TO DISPLAY
07570          JP      CONT5           ;AND DISPLAY IT
07580 HOLDZ    DEFB    0               ;BYTE STASH
07590 UPSIDE   DEFB    0               ;LOWER CASE POINTER
07600 CK       PUSH    AF              ;SAVE BYTE
07610          LD      A,(14464)       ;SHIFT KEY PRESSED ?
07620          CP      1               ;IF SO
07630          JP      Z,COR           ;IGNORE IT
07640          POP     AF              ;RESTORE BYTE
07650          JP      CONT5           ;CONTINUE ONWARD
07660 COR      POP     AF              ;ADJUST STACK FOR PUSH
07670          JP      CONT3           ;GO SCAN NEXT INPUT
07680 ONE28    CALL    SLOWLY          ;SLOWDOWN AS THIS IS AN
07690          CALL    SLOWLY          ;AUTO REPEAT FUNCTION
07700          LD      A,128           ;END OF MESSAGE DELIMITER
07710          JP      CONT5           ;STUFF IT IN MEM & VIDEO
07720 NOTASC   CP      16              ;ELECTRIC PENCIL CTRL KEY
07730          JP      Z,29760         ;REINITIALIZE PGM POINTER
07740          LD      A,(14352)       ;KYBD ZERO PSUEDO MEMORY
07750          CP      1               ;SHIFT ZERO PRESSED ?
07760          JP      Z,ONE28         ;END OF MESSAGE DELIMITER
07770          LD      A,(14337)       ;KYBD @ PSUEDO MEMORY
07780          CP      1               ;@ KEY PRESSED ?
07790          JP      NZ,CONT3B       ;IF NOT, CONTINUE ONWARD
07800          CALL    CLS             ;C L E A R  V I D E O
07810          CALL    CARETN          ;VIDEO SKIP A LINE
07820          LD      HL,VALMS        ;STACK POINTER MESSAGE
07830          CALL    DIZPLA          ;DISPLAY IT ON VIDEO
07840          LD      HL,0            ;ZERO OUT HL
07850          ADD     HL,SP           ;ADD IT TO STACK VALUE
07860          CALL    0A9AH           ;MOVE IT TO ACCUM
07870          XOR     A               ;ZERO OUT 'A'
07880          CALL    1034H           ;GENERATE
07890          OR      (HL)            ;UNSIGNED
07900          CALL    0FD9H           ;INTEGER
07910          CALL    DIZPLA          ;DISPLAY IT ON VIDEO
07920          CALL    CARETN          ;VIDEO CARRIAGE RETURN
07930          CALL    CARETN          ;VIDEO CARRIAGE RETURN
07940          LD      HL,VALMS0       ;MEMORY LOCATION MESSAGE
07950          CALL    DIZPLA          ;DISPLAY IT ON VIDEO
07960          PUSH    IY              ;SWAP IY MEM LOCATION
07970          POP     HL              ;INTO HL
07980          CALL    0A9AH           ;MOVE HL TO ACCUM
07990          XOR     A               ;ZERO OUT 'A'
08000          CALL    1034H           ;GENERATE
08010          OR      (HL)            ;UNSIGNED
08020          CALL    0FD9H           ;INTEGER
08030          CALL    DIZPLA          ;DISPLAY IT ON VIDEO
08040          CALL    CARETN          ;VIDEO CARRIAGE RETURN
08050          CALL    CARETN          ;VIDEO CARRIAGE RETURN
08060          LD      HL,VALMS1       ;MEM VALUE MESSAGE
08070          CALL    DIZPLA          ;DISPLAY IT ON VIDEO
08080          LD      A,(IY)          ;IY LOCATION MEM VALUE
08090          LD      L,A             ;INTO 'L'
08100          LD      H,0             ;ZERO OUT 'H'
```

```
08110         CALL    0A9AH           ;MOVE HL TO ACCUM
08120         CALL    0FBDH           ;CONVERT ACCUM TO STRING
08130         CALL    DIZPLA          ;AND DISPLAY IT ON VIDEO
08140         CALL    CARETN          ;VIDEO CARRIAGE RETURN
08150         CALL    CARETN          ;VIDEO CARRIAGE RETURN
08160         LD      HL,VALMS2       ;INPUT NEW MEM MESSAGE
08170         CALL    DIZPLA          ;DISPLAY IT ON VIDEO
08180         LD      BC,32000        ;1/2 SECOND
08190         CALL    060H            ;TIME DELAY
08200         CALL    1BB3H           ;INPUT NEW VALU FROM KYBD
08210         RST     10H             ;SCAN STRING SET 'C' FLAG
08220         CALL    0E6CH           ;ASCII $ TO ACCUM RET MIN
08230         CALL    0A7FH           ;CONVERT ACCUM TO INTEGER
08240         LD      A,L             ;NEW MEM VALUE
08250         LD      (IY),A          ;AND STUFF IT IN MEMORY
08260 NOTAS   LD      HL,(LASMEM)     ;BEGINNING MEM LOCATION
08270         LD      DE,15360        ;BEGINNING VIDEO MEM
08280         LD      BC,1024         ;RESTORE VIDEO ALMOST
08290         LDIR                    ;SAME AS BEFORE
08300         CALL    CKAHED          ;TEST VIDEO IN BOUNDS ?
08310         INC     IX              ;OK, SO MOVE CURSOR AHEAD
08320         INC     IY              ;& INCREMENT MEM LOCATION
08330         JP      CONT3           ;GO BACK & SCAN KEYBOARD
08340 VALMS   DEFM    'STACK POINTER = '
08350         DEFB    0               ;DELIMITER
08360 VALMSO  DEFM    'MEM LOCATION IS '
08370         DEFB    0               ;DELIMITER
08380 VALMS1  DEFM    'MEMORY VALUE IS '
08390         DEFB    0               ;DELIMITER
08400 VALMS2  DEFM    'INPUT NEW VALUE '
08410         DEFB    0               ;DELIMITER
08420 MEMO    DEFW    0               ;MEMORY LOCATION STASH
08430 MEMO1   DEFW    0               ;MEM LOCATION STASH -1
08440 NEXT2   LD      BC,24000        ;ABOUT 1/3 SECOND
08450         CALL    060H            ;TIME DELAY
08460         JP      NEXT            ;AWAIT EDIT MODE COMMAND
08470 CARETN  LD      A,13            ;VIDEO
08480         CALL    033H            ;CARRIAGE RETURN
08490         RET                     ;RETURN WHENCE U CAME +1
08500 CLS     LD      HL,15360        ;BEGIN VIDEO MEM
08510         LD      DE,15361        ;PLUS ONE
08520         LD      BC,1023         ;BYTES TO CLEAR
08530         LD      (HL),32         ;WITH SPACES
08540         LD      (16416),HL      ;RESET VIDEO CURSOR
08550         LDIR                    ;MOVE'M RIGHT NOW
08560         RET                     ;RETURN WHENCE U CAME +1
08570 DIZPLA  PUSH    AF              ;SAVE
08580         PUSH    BC              ;EVERTHING
08590         PUSH    DE              ;INCLUDING
08600         PUSH    HL              ;THE
08610         PUSH    IX              ;KITCHEN
08620         PUSH    IY              ;SINK
08630 MORE1   LD      A,(HL)          ;BYTE TO DISPLAY
08640         CP      0               ;END MESSAGE DELIMITER
08650         JP      Z,FINIS1        ;IF SO, ALL DONE
08660         CALL    033H            ;DISPLAY & UPDATE CURSOR
08670         INC     HL              ;MESSAGE MEM LOCATION
08680         JP      MORE1           ;GO DISPLAY NEXT ONE
08690 FINIS1  POP     IY              ;SINK
08700         POP     IX              ;KITCHEN
08710         POP     HL              ;THE
08720         POP     DE              ;INCLUDING
08730         POP     BC              ;EVERTHING
08740         POP     AF              ;RESTORE
08750         RET                     ;RETURN WHENCE U CAME +1
08760
08770 ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
08780 ; END OF EDIT/MODIFY/MONITOR SUBROUTINE
```