Packet Radio - A Software Approach

Robert M. Richardson, FJ4UCI-I
22 North Lake Drive
Chautauqua Lake, N.Y. 14722

Abstract:

A software rather than hardware approach to synchronous Packet Radio communication at 1200 or 2400 Baud using the Radio Shack TRS-80, Model I or Model III microcomputer is described. The program duplicates virtually all the functions provided by the Vancouver Area Digital Communications Group (VADCG) terminal node controller board which requires an 8085 microprocessor, an 8273 synchronous data link controller (SDLC), an 8250 serial I/O, and a number of EPROM and RAM memory chips, plus a separate microcomputer with RS232C interface and a 1200/2200 Hz modem.

The only external equipment required by the software approach, other than a TRS-80, amateur VHF transceiver and antenna, is a port zero encoder/decoder, and two EXAR chips for AFSK keying and demodulation. The program has been extensively tested on the 2 meter amateur band working into southern Ontario and locally in western New York.

Introduction:

Packet radio communications is coming down the amateur radio pike in the near future, and the near future is now upon us. We have closely followed the evolution of amateur packet radio with its asynchronous beginnings in the Montreal area during early 1979 and its synchronous beginnings in the Vancouver area later in 1979/1980. The synchronous packet radio pioneers, including Douglas Lockhart VE7APU et al in the Vancover area developed the rightly famous and widely used VADCG terminal node controller board which was the introduction to synchronous amateur packet radio for the majority of all amateurs actively participating in packet communications today.

Though a number of SDLC controllers in addition to the Intel 8273 are now available, such as those from Western Digital and Zilog, and the price will hopefully be coming down as volume increases, there is yet another approach to amateur packet radio which due to its low cost and simplicity, may greatly broaden amateur participation in this wave of the future. This approach is the software rather than firmware approach using a Model I or Model III TRS-80 with 48K memory and hopefully for convenience, 1 or 2 mini-disk drives. Suffice it to say, assembly language is used which offers nearly 300 times faster execution speed than standard Fortran, Pascal, or Basic high level languages.

Even with the remarkable speed assembly language offers the user, it does require a finite amount of time in the receive mode for the program to serially accomplish what the dedicated SDLC chips are able to accomplish in parallel: i.e., find the last openning flag and address and store the packet bits in memory, convert the serial packet bits to decimal with zero deletion (the opposite of zero insertion), and do the CRC checking for each frame. Nevertheless, this entire process for the average packet only requires 90 milliseconds which is not a significant or even noticeable time delay to the operator at either end of the circuit.

General:

The program is comprised of 5 segments:

1. The transmit mode segment which does the work horse job of converting prepared messages or programs, either keyboard or disk input, into IBM SDLC format, and clocking them out bit by bit serially at the desired Baud rate via port zero. Packet and frame length may be any length desired from 1 to 250 bytes plus address, control and CRC bytes, and may be input from the menu. See figure 1. A total packet or frame length of 256 bytes will be allowed when intra repeater routing comes to pass. The extra two bytes will serve to determine routing. The number of preamble flags sent before the packet may also be programmed which is a courtesy for those with slow transmit and receive switching at the receiving end, A number of prepared messages may selected from the menu as well as keyboard input to memory for a nearly immediate packet. The video display utilizes the split screen format with the top 8 lines for receive and the bottom 8 lines for transmit. See figure 2. Independent sequential scrolling is provided for both transmit and receive modes.

2. The cyclic redundancy check,

CRC16, segment which automatically generates the 2 byte CRC in IBM modulo 2 format and appends these 2 bytes to each frame transmitted. This same subroutine serves the receive mode to check each incoming frame For CPC validity also using the IBM modulo 2 CRC algorithm. which the dedicated SDLC chips utilize.

3. The receive mode segment has a software equivalent of a digital phase locked loop which takes the serial packet input via port zero, and stores each bit in memory either 8 bits per byte for best memory utilization, or 1 bit per byte for instructional purposes (easy to visualize). After the complete packet has been received, it is first converted from binary to decimal, then stored in high memory and displayed on video while the CRC test is being made for each frame if a multi frame packet was received. Depending upon whether the packet is of the unnumbered, supervisory, or information variety, appropriate action is taken automatically. The conventions followed are those used by the VADCG so the program is fully compatible with those amateurs using the Vancouver terminal node controller. The program will receive and decode packets of any length up to the 12,288 byte capacity of the bit store allocation. Average packet decoding time after reception of the packet transmmission is 90 milliseconds, as previously mentioned.

4. The edit/modify segment is a unique and extremely useful utility that allows the operator instant access to any 1 of the 1024 byte, 60 pages of memory in the TRS-80. Each page of memory is displayed at one time on the video display in TRS-80 ASCII or graphics format. Either upper case only, or upper and lower case edit/modify may be selected from the menu. Pressing the 'M' for modify key intiates a flashing cursor that may be moved up, down, left, or right on the displayed page with the arrow keys. Keyboard input ASCII value may be input directly if desired or pressing the shift '@' keys displays the memory location in decimal, memory value at this location, stack pointer value, and the operator asked to input the new value. Upon inputting the new value, the full memory page is displayed again, Pages are moved up or down through memory by pressing the BREAK key (flashing cursor disappears) and then the ENTER key to move up a page in memory or the minus/DASH key to move down a page in memory. Needless to say, though ROM may be examined with this function, only RAM may be modified. To elininate tedious paging by hand, a number of control keys allows instant access to the more frequently used memory locations; i.e., transmit program store, received bit store, and received message store.

5. Morse code I.D. and Morse transmit segment. This minor subroutine called by shift 'I' sends the Morse I.D. of the transmitting station at any speed desired to satisfy F.C.C. requirements. Conversely, Morse code at any speed may be tranmitted from the keyboard via shift 'X'.

Memory Management:

The operating program resides in memory from 29696 through 40959. Three 12,288 byte segments are used exclusively by:

A. 17408-29695 is reserved for the program or data to be transmitted store. Normally, a disk program to be transmitted via packet is first loaded from disk to 40960+ in memory and then moved down to the 17408+ area by pressing shift 'Y'. In the 'connected' mode of operation the program or data fron 17408+ i s automatically transmitted in single frame packets of 254 bytes length (a western New York convention so that a single station may NOT monopolize the packet repeater unfairly). This automatic transmission is called by pressing 'B' from the menu. Acknowledgement (ACK) packets are received automatically and if valid the next packet automatically transmitted, otherwise the previous packet is retransmitted.

B. 40960-53247 is reserved for incoming received bit storage. We prefer to store a bit per byte so that the user may visualize the stored bit pattern, though 8 bits per byte could be used for better memory utilization. Unless directed by the menu command to SAVE bit storage, this area is automatically cleared after each packet is converted to decimal and stored in high memory. Surely all you amateurs operating VADCG terminal node controllers know that it sends two SDLC logical zero 'bytes' AFTER the opening flags, which are then followed by a single flag BEFORE the address byte. Or did you? Packets less than 4 bytes total length, are ignored by the SDLC protocol that is used in this program.

C. 53248-65535 is utilized for storing the converted decimal byte values from received packets. When it is full, an automatic 'not ready to receive' (RNR) = 'wait' is transmitted and when acknowledged, the operator may clear out all address, control, and CRC bytes by pressing '$' from the menu. Pressing shift 'B' then takes the operator to DOS ready where the received program or data may be DUMPed to disk, after which one returns to the program, sends a 'ready to receive' (RR) to 'clear' the previous 'wait' and continues upward and onward if it is indeed a program or data longer than 12,288 bytes bytes in length.

Conclusion:

Amateur radio has room for all types and varieties of individuals with

dramatically differing interests. Some prefer operating and could care less how an electron gets from 'A' location to 'B' location and what is involved in getting it there, There is certainly nothing wrong with that, and to that variety of amateur we highly recommend the Vancouver terminal node controller kit *. With a few hours of soldering you are 'on the air' using their EPROMs. To the variety of amateur who truly wishes to thoroughly understand how the absolutely brilliant IBM SDLC protocol operates, how to modify it for the forthcoming HDLC modes, we recommend the software rather than hardware approach. When you are done, you will not only be an 'instant SDLC expert,' but a better assembly language programmer too.

The software approach to packet radio communications is obviously limited by the clock speed of the microcomputer being used. The Model I and Model III TPS-80 will easily handle 1200 and 2400 Baud packets, with 4800 Baud packets being somewhat marginal but acceptable, without modifying the standard crystal clock of either microcomputer. By installing one of the numerous clock speed up kits available (4 MHz), both 4800 and 9600 Baud packets could be handled.

We are indebted to a number of our Canadian neighbors in the Hamilton and Toronto area for assistance in testing the software approach to packet radio. Most noteable have been VE3MWM, VE3DSP, VE3IUV, and VE3DVV. Their packet repeater located in the southern environs of Toronto with the apt call sign of VE3RPT is now active on 145.650 MHz.

Locally (6.5 miles northeast of our QTH), the major effort to install a packet repeater in the greater Bufffalo area has been borne by W2EUP with considerable assistance from VE 3MWM. It will be linked to both the Toronto repeater to the north and a new packet repeater in the Syracuse area that will be able to link into the greater New York City area to the east, and thence linked up and down the east coast.

We have only touched briefly on a few of the highlights of the software approach to packet radio communications. To go through it in depth would take much more than the allotted space and time. For those who wish to explore the subject in greater depth, at least a few hundred pages worth, we suggest you watch for the new book, 'The Packet Radio Handbook' that will be published spring '83 by Richcraft Engineering Ltd., #1 Wahmeda Industrial Park, Chautauqua, N.Y. 14722 at $22 postpaid to the U.S. and Canada and $30 postpaid overseas.

* Vancouver Area Digital Comm. Group
818 Rondeau Street
Coquitlam, British Columbia
Canada V3J 523

```
                    ENTER OPTION DESIRED ? _

CHANGE ADDRESSEE & NUMBER  = A     W2EUP CONNECT REQUEST CQ   = B
NOT CONNECTED TOGGLE       = C     W2EUP DISCONNECT REQUEST   = D
SEND PACKETS FROM LO-MEM   = E     W2EUP CONNECT ACKNOWLEDGE  = F
USING ONLY SOFTWARE MSG    = G     WORKING ON AMSAT AX, 25 MSG = H
NOT INSERT DPLL BIT TOGGLE = I     SEND HI-MEM CONTINUOUSLY   = J
NOW IN UFPER CASE MODIFY   = K     FILL HI-MEM WITH UUUUU     = L
DISPLAY/EDIT MEMORY PAGE   = M     SET DISCRETE PACKET LENGTH = N
LOAD HI-MEM ALL 11111      = O     CHANGE DPLL TIMING VALUE:  = P
SEND CONTINUCUS FLAGS/126  = Q     ABORT LO-MEM XMIT SEQUENCE = R
MULTI-FRAME PACKET TEST    = S     QUICK BROWN FOX TEST MSG   = T
CLEAR NON-PGM MEM 17K-65K  = u     INPUT/TRANSMIT MESSAGE = V & W
ANY PACKETEERS ABOOT 255/0 = X     MOVE MID-MEM TO LO-MEM     = Y
NORM BIT STASH CLEAR       = 1     MOVE RECV PACKS TO LO-MEM  = 2
SET NUMBER OPENING FLAGS   = 3     WAIT & CLEAR WAIT TOGGLE   = 4
```

Figure 1

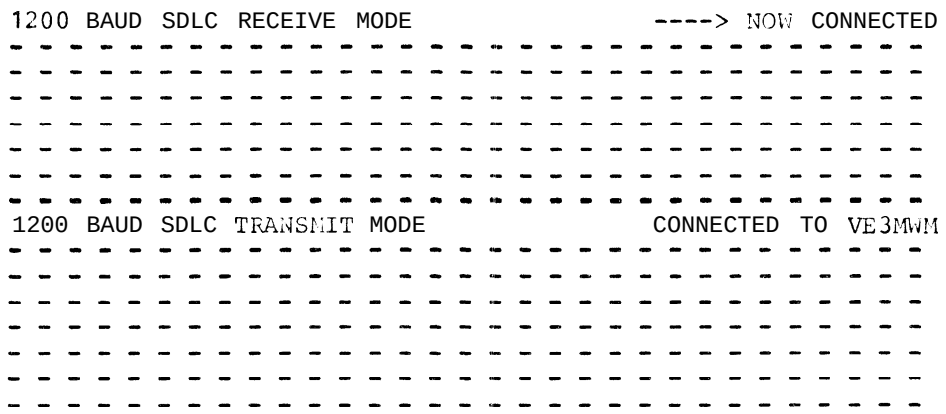

Figure 2