Packet Radio

By Peter J. Eaton WB9FLW

35 Norspur, Route 4
Edwardsville, IL 62025

Radio amateurs in Canada, Sweden, and the U.S. have been experimenting with packet radio, a system of computer-based communications. This new mode can provide high-speed communication that is interference resistant and is efficient use of the spectrum.

What is packet radio?

Packet radio is a communication of digitally encoded data (similar to teletype and ASCII) that includes handshaking and error detection. The error checking is done by including a frame check sequence (FCS) with each transmission (called a packet of data). The receiver acknowledges an error-free packet by sending back an acknowledge (ACK) signal.

If the sending station does not receive an ACK within a certain period of time, it automatically retransmits the packet.

A packet also contains an address, so a packet station will ignore any packets not addressed to it. Since packets are sent in short bursts, many stations can use the same frequency without conflict. On very busy frequencies you might notice some delay in sending data or receiving an acknowledgement, but you never hear the other stations.

Requirements

Each station has to have a terminal, a terminal node controller (TNC), and an amateur radio transceiver.

The terminal can be a simple dumb terminal, a printing terminal, a personal computer, or even a mainframe type system.

Most terminals generate asynchronous characters. These characters have 1 or more "marks" (binary 1's) which indicate where each character begins (start bits) and ends (stop bits). The characters are sent at a specific baud (bit) rate. There is no set time interval between characters.

The TNC

The terminal node controller (TNC) is the heart of the system. It has an asynchronous serial port which connects to the terminal (etc.) and an additional port which connects to the transceiver's microphone line, speaker line, and transmit control line.

The TNC collects the data coming in from the terminal, until it has enough for a packet. It then attaches a header which includes the address of the destination and control information for the network, and it attaches the error checksum and flags to mark the beginning and end of the packet.

The TNC then sends the packet out through the transmitter at the packet channel baud rate. Usually it produces AFSK modulation, which means it sends one tone for a mark and another for a space.

The receiving TNC decodes the audio tones (from the speaker line), removing and checking the address information and the checksum. If the packet is correctly addressed and correctly received then it passes the information to the terminal (at whatever baud rate is appropriate for that terminal).

The modem part of the TNC translates the tones into ones and zeros. Most packet radio modems operate at 1200 baud, and the tones are 1200 Hz and 2200 Hz. This is the same pair of frequencies used by the bell 202 (half-duplex) modem which is available as surplus.

The Transceiver

The transceiver (transmitter and receiver) usually operates on the amateur radio 2 meter (144-148 MHz) band. The main requirement is that the transceiver be able to pass 2200 Hz audio tones adequately. Most 2 meter rigs will do this.

Handling the Protocol

The functions of the TNC which would be difficult to duplicate on a personal computer are the protocol decoding/encoding and simultaneous operator control.

The protocol sets the contents of the packet header and trailer so that the receiving TNCs know the purpose of the packet. For instance, is the packet being used to check into a net? Or is it part of a communication with another station? Or is it simply acknowledging receipt of another packet? Meanwhile, the station operator may want to interrupt the proceedings.

Obviously, a system running under a BASIC interpreter would not keep up, so we've had to write the software in assembly language. If the TNC were replaced by personal computers, we would have to write new software for each different computer.

Since the TNC must be constantly listening to both ports while putting packets together or taking them apart, the hardware of personal computers may not even be capable of handling the task.

Editor's note: Peter is obviously not aware of the incredible feats of engineering taken on by inspired BB owners. The common ham (amateur radio operator) with his hand-held appliance (I have one too) won't know what hit him if we turn the BB group loose on the airwaves. (Legally, of course!)

Packet Details

A packet is the basic message unit. It usually consists of text typed in by the operator and sandwiched between the header and the trailer.

During a typical QSO (conversation on the air) a packet would be put together and sent out each time the operator ended a line by hitting a carriage return. The length of the packet is limited to no more than 128 characters. This limitation helps a single user from hogging the frequency as well as making sure that the sending and receiving TNCs don't get swamped.

The data inside the packet need not be ASCII characters. They could be BCD, EBCDIC, or even binary data such as .COM files.

The TNC uses a bit-oriented protocol called HDLC (high level data link control). This protocol was chosen because it is supported by a single LSI communications chip, which simplifies both the hardware and the software. Also, in this mode, data is transferred faster since individual characters no longer need start and stop bits in this synchronous mode. See Figure 1. *(Editor's note: the Z80 SIO supports HDLC very nicely, handling the CRC and flag generation. It also checks the first byte of the address for a match. See a Z80 SIO manual for details of HDLC which is also called SDLC.)*

The address field contains routing information for the packet. This information may include the destination station, the originating station, and possibly, some intermediate routing instructions.

Identification of the stations might be by network address number or by amateur call sign.

The control field describes the purpose of the packet. It identifies packets which are network check-ins or check-outs, packet acknowledgements, or requests for information from net control. It may also contain a sequence number for a multi-packet message which must be received in the correct order.

The data field contains the message.

The FCS is just another name for a CRC, a fancy checksum.

What is a packet network?

A local area packet network (LAN) is made up of a net control station (station node) and a number of individual stations (terminal nodes). The net may operate through a digital repeater which can be a single-frequency repeater or a standard duplex repeater.

As operators sign on to the net, they are assigned address codes by the net control. An operator wanting to talk to another station logged onto the net can simply address his transmissions to that station.

An operator can choose to listen in on all transmissions or just those addressed to him. Of course, he will only send acknowledgements for transmissions directed to him.

The operator who is acting as net control operates his station just like anyone else; the special net control functions are taken care of by his TNC.

Connecting LANs

Some stations will be able to access more than one LAN. These stations could be members of both groups and serve as communications links through which packets can move between nets.

Plus there are three other ways being considered for transfer of data between LANs.

1. TERRACON would be a high-speed ground-based link using UHF and microwave relays. It could handle most packet radio communications in the U.S. and Canada. It will probably be a few years before this system becomes useful.

2. AMICON would be a satellite-based utilizing one of the special services channels on the AMSAT phase II-B satellite. This system will allow intercontinental linking with isolated areas which would not be accessible by TERRACON.

3. SKIPCON is the projected high frequency network. The nature of HF propagation requires slower data rates (50 to 600 baud) and error correction as well as error detection protocol. Experiments with this long range mode began in 1981.

How to get started.

There are now two TNC designs. The first TNC was designed by the Vancouver (BC) Amateur Digital Communications Group and they sell a bare board with instructions. They also sell a modem kit. This TNC is based on the 8085 and the 8273 HDLC controller. It includes 4K bytes of 2114 RAM and four 2708s.

The Tucson Amateur Packet Radio Group is testing a second TNC design. This TNC has the modem, radio interface, serial and parallel terminal interfaces and power supply circuit on a single board. It is based on a 6809 and can contain up to 48K of RAM and ROM. The 1933 HDLC chip on this board is compatible with the 8273 chip used on the Vancouver group's board.

Editor's note, I don't have the addresses of the two clubs mentioned but hopefully I'll have that information by the next issue. If you can't wait, contact Peter or get on the air and locate folks from these areas who could tell you.

■ ■ ■

Figure 1 - Makeup of the HDLC Packet

FLAG	ADDRESS	CONTROL	DATA	FCS	FLAG

C'ing Clearly

Aztec CII Compiler Ver. 1.05

Review by Bill LaFay

1214 Westridge Circle
Lynchburg Va. 24502

I am new to the world of C programming. I used to use BASIC for all my program needs. BASIC was easy to learn and use. C is quite different from BASIC and not being familiar with structured languages, I decided that the C I would buy must follow "the book" (Kernighan & Ritchie) religiously. Through the urging of a friend, I bought AZTEC C.

Overview

Aztec C comes in two flavors: Integer and Floating Point. Two for the 8080 and two for the Z80. It also comes with its own assembler, linker and librarian. These, according to the user manual, are a sub-set of Microsoft's M80/L80. If you have M80 & L80, you can use them along with your personal external libraries.

The compiler is a one pass compiler so all references must be forward. It has switches which: allow the source text to be added to the assembly language output as comments; define the length of expression lines in the source program (default is 120), and define the size of internal work tables. The output of the compiler is an 8080 assembly listing in the case of the 8080 versions and expanded 8080 for the Z80 versions (not true Z80 mnemonics).

Strengths

1. Compiles for own assembler/linker or for Microsoft's M80/L80.
2. Initializers on declarations.
3. Random access file I/O.
4. Very complete error message definitions.
5. Compilation, assembly, and linking under SUBMIT file.
6. Supports 16 significant digit floating point arithmetic.
7. FP exponent range E+-128.
8. Dynamic storage allocation.
9. .MAC compiler output can be hand optimized.
10. Structures, pointers, casts.
11. Long, Float, Unsigned, double, static, register, extern.

Weaknesses

1. Calls must have same type & number of arguments as the called function.
2. No FCALL function for calls to existing FORTRAN LIBRARY routines.
3. Internal floating point notation is not compatible with the MICROSOFT FORTRAN or BASCOM conventions.
4. No tracer option for single step debugging. ZSID must be used on the .COM file.
5. No built-in utility for easy debugging.
6. No code optimization option.
7. No bit fields.
8. No pipes.

Documentation

The manual is new with this version and is chock full of information. It gives good coverage of all facets of Aztec system operation.

MANX SOFTWARE SYSTEMS sent an update disk that included the needed libraries called for by the linker. This saved a lot of compiling, assembling etc.

The user's guide seems to contain the information needed but it is "impossible" for the uninitiated. I've seen much better and would hope the needed improvement will be made.

Ease of Use

The submit file that comes with the package makes compiling, assembling, and linking very easy. I use the CZII floating point compiler with M80 & L80 and it works very nicely.

Code Size And Quality

The Aztec C is a single pass compiler and does no optimizing but still generates good code. The CLIBZ80 library is large and no doubt accounts for the large object file size.

Conclusions

There were a number of problems with the early versions of the floating point compiler (CZII) but now things seem to be in good shape. The people at Manx have given me prompt and courteous service. The compiler works like the book (Kernighan & Ritchie) says and has all the features except those indicated above. Would I buy it again if I had to do it over? I think I would.

For the benchmark test, I used the same program as shown on page 4 of August 1982 Micro-Cornucopia. I also am running a 4MHz Big Board.

Benchmark Results
4MHZ Z80 Bigboard

Compile Time 16 sec
Assembly Time(M80) 16 sec
Linker Time(L80) 51 sec
Run Time:
-original prog. 32 sec
-static variables 22 sec
-register int var. 22 sec
Object File Size 17K (.COM)

NOTE: I have written my own version of the trig functions I needed. They are written in Aztec C and are accurate to 9 significant digits which is fine for almost all applications. These will work for Aztec C version 1.04 which doesn't have them.

They should also work on any C compiler that handles double precision numbers. I have also interfaced Aztec C with the MICROSOFT FORTRAN library. This speeds up computations by a factor of 3.

Editor's note: Any novice who is writing trig functions has my vote for "novice of the year." Also, version 1.05 not only adds such things as I/O redirection, the scientific math functions, scanf, and relative byte support for unbuffered I/O—it also has a fancy new manual. And still, it is available (to Micro C readers only) at $149.00 Anyone interested in Bill's scientific routines and FORTRAN library interface should contact Micro C. If you are interested we will put them together as a user disk.

Manx Software Systems
PO Box 55
Shrewsbury NJ 07701
201-780-4004 ■ ■ ■

ENCLOSURE

The home for your BIG BOARD that you will be proud of. With a POWER SUPPLY that will run the BIG BOARD and two standard or slim-line eight inch drives. It comes fully wired with all connectors and is pre-wired for disk expansion. The BIG BOARD mounts on the inside of the top cover allowing all cables to dress neatly to the rear of the cabnet and to allow ease of access for repair.

The enclosure comes in single or double wide. The double wide will fit both standard drives or (with the adaptor, $10) the over sized Shugart SA 800-2, from Cascade Electronics, Inc.

Available without connectors and un-wired as a drive cabinet or as a do-it-yourself enclosure for the BIG BOARD or other SBC.

STANDARD FEATURES INCLUDE

* Power supply
  +5.0V @ 4A w/OVP, +24V @ 2.5A, ±12V @ 200mA
  all voltages have over current prtection.
* Fan
* Reset switch
* Key lock power switch
* Bell circuit and speaker
* AC outlets, one switched
* Solid state AC relay
* Composite video jack
* Reverse video switch
* Disk drive exansion pre-wired (50 pin + DC + AC)
* Color- biege and chocolate
* Optional- adapters & plates
* 6"H x 13"W x 16"D or 24"W for the double wide.

The box (La Caja)

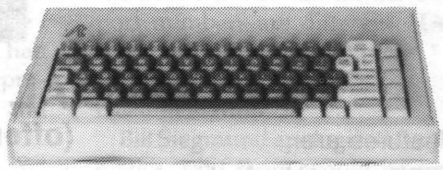
single wide, wired	$379
double wide, wired	$399
single wide, un-wired	$279
double wide, un-wired	$299
adaptor for SA 800-2	$ 10
shipping and handling	$ 10

KEYBOARD

The key board was designed to complement the 'La Caja' enclosure in color, design and function and to be 100% compatible with the BIG BOARD.

FEATURES

* 66 keys
* two-key roll over
* ASCII 8, positive logic
* Delayed negative strobe
* Sculptured key caps
* Five user defined keys
* Power requirments- +5.0V @ 150mA, -12V @ 20mA
* Color, biege and chocolate
* 1.5"-2.5"H x 13"W x 8"D



KEY BOARD w/o cable $159

shipping and handling $ 5

PRINTER

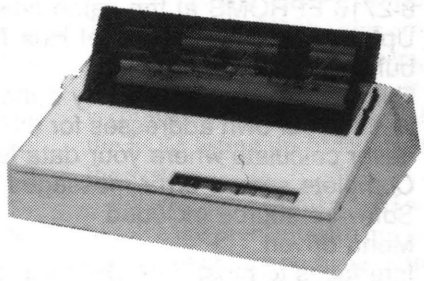
The BROTHER HR-15 daisy wheel printer is a compact printer that will handle paper up to 13.5 inches.

A variety of orerations have been added to increase clarity, precision and to emphasize important points. The HR-15 Prints super-script and sub-script characters with the ability to adjust character spaces proportionally and provides automatic underlining with either bold or red print. For ease of operation both daisy wheel and ribbon are enclosed in casettes making changing trouble free.

In closing, the BROTHER HR-15 makes an exclent choice for word processing and general printing.

FEATURES

* 2K byte buffer
* Graphic printing
* Bold printing
* Proportional
* 13 cps
* Cassette type daisy wheel
* RS-232C or CENTRONICS parallel
* Bi-directional logic seeking head, 1/120, 15, 15, 10 positions per inch.
* Bi-directional friction platen, 1/48, 6, 4, 3, positions per inch.
* Options- Tractor feed, Auto cut sheet feed
* Color- biege
* 6"H x 19.5"W x 13"D



PRINTER, Brother HR-15 $595

shipping and handling $ 10

POWER SUPPLY for the BIG BOARD (+5.0V @ 4A w/OVP, +24V @2.5A, ±12V @ 200mA). All supplys have over current prtection (4"H x 3"W x 11"D, 6.5 lb)

POWER SUPPLY $ 95 +$5 S&H

Transformer for the BIG BOARD as above (3"H x 3"W x 4"D, 5.5 lb)

TRANSFORMER $ 29 +$3 S&H

ASTROTRONICS also salls disks, ribbons, daisy wheels and paper

AstroTronics MICROSYSTEMS

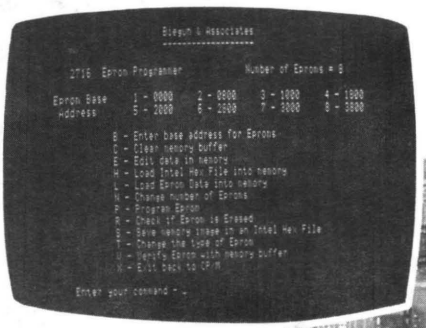
California Residents Add 6% Tax

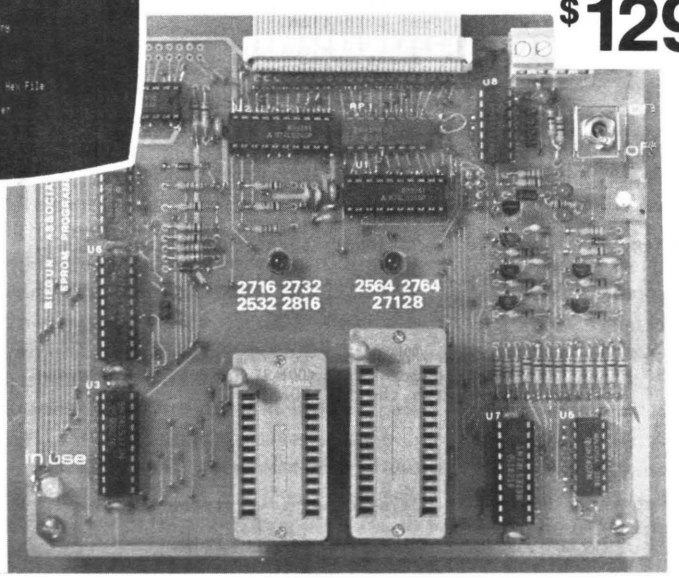
(714) 734-6006

1137 TOPAZ ST.
CORONA, CA 91720

CP/M EPROM PROGRAMMER

interfaces to BB1 parallel port





$129.95

SPECIAL INTRODUCTORY PRICE

(offer expires Sept. 30, 1983)

features are:

- Program, Verify, Load, and check for Erased
  Intel Eproms 2716, 2732(A), 2764, 27128
  T.I. Eproms 2516, 2532, 2564
  Xicor EEprom 2816A
- 16 k byte memory buffer allows you to work with up to 8-2716 EPROMS at the same time
- Upload and Download Intel Hex Files with the memory buffer
- Edit the data in the memory buffer
- Define your own addresses for the memory buffer so you never calculate where your data is in the memory buffer
- Complete screen error messages
- Software source included
- Menu driven
- Interfaces to most Z80 CP/M systems with parallel ports and a TPA = 100H

options (available later)

- EPROM Emulator
- Adapters for single chip processors

Requires +5 v. @ 300 ma., +25 V. @ 100 ma., and interface cable

Software is delivered on a standard 8 inch SS SD floppy disk.

1. Software and schematic	29.95
2. Bareboard and schematic	39.95
3. Software and bareboard	64.95
4. Software and kit (less ZIFS)	89.95
5. Software and full kit	114.95
6. Programmer A + T	129.95
plus shipping	5.00

Still available
BB11 software and source
(uses the BB11 for programming) 29.95

all prices shown are in US Funds

Allow 4-6 weeks for delivery

Send order to:

Biegun and Associates
P.O. Box 4071
Station "B"
Winnipeg, Manitoba
Canada R2W 5K8

Biegun & Associates

CP/M is a trademark of Digital Research, Inc.

Bringing Up The BB II

By Jim Showker

11174 Penrose #C
Sun Valley, CA 91352

Sometime during the first week of December '82 I sold my BB I and the next day ordered a BB II bare board from Cal Tex in San Jose. I had talked to them twice in the previous month to confirm that they could ship from stock immediately, since being without a computer at home for any length of time makes me feel semi-naked. I was concerned because I waited quite a while for my BB I.

However Cal Tex was temporarily out of various parts the day I placed my order. I received my bare board, a "hard to get parts kit," and CBIOS on disk about 2½ weeks later.

Assembly

I very carefully assembled and soldered the sockets and components to the board. I then cleaned the rosin off the back of the board and over a period of two days was able to find 10 errors in soldering. There were 4 connections unsoldered, 5 possible cold or otherwise disreputable looking solder joints and 1 solder bridge.

After carefully installing all the IC's and applying power to the board I was quite happy indeed to hear, after about two seconds, a loud beep from the board. This meant that the CPU, memory, and I/O ports were probably working correctly.

Debugging

I had no video, however, and was able to quickly establish that the CRT section was not working at all. There was a 16 MHz clock, but that was it. It didn't take too long to establish that U45 was bad. After replacing it, I still had no video. I now had all the necessary video signals, but no composite output. I figured that the output transistor had been taken out when U45 went. That was when my troubles began.

I was now trouble shooting from the schematics, of course. I pulled the video output transistor, and put in a 2N2907 like it said on the schematic. (I didn't notice that the component layout sheet listed the same transistor as a 2N2222.)

I now had a completely inverted composite video signal. I assumed the design was faulty and eventually ended up designing a composite video generator on the "breadboard" portion of the PC Board. Only recently did I notice the error and replace the 2N2907 with a 2N2222. This silly error probably cost me 25 hours.

Drives

I then hooked up the disk drives (2 ancient Siemens, that had worked perfectly on SD with the BB I). Installing the CBIOS turned out to be quite easy, unexpectedly so. But, I could not format in double density without LOTS of errors. I got sidetracked with the schematics again, thinking something was wrong on the board as there are many hookups that are not as shown.

After a few days I went out and bought a new drive, to see if that could be the problem. Being not a rich person, I bought another $250 Siemens drive (a mistake). The older ones had worked fine for me on SD and these were specified for double density. Anything else I could buy was $380 or more. I still had the same problems with the format program. Two weeks of hair pulling and chin scratching followed.

On a Sunday, I went to my office and disassembled the computer there (a custom installed rack mount S-100 System) and brought home two Shugarts that I knew worked fine on double density. Voila, no more errors.

I resigned myself to spending $400 a piece on drives and went back to Priority 1 to trade up or get my money back. They convinced me to try another Siemens as they said they were getting very few returns. I took another one home and it worked perfectly.

The next day I went to buy another, for drive B:, and after hooking it up, it had the same problems as the first. I exchanged it and the fourth one did not work at all, either. It made horrible clanking noises and wouldn't load the head. I was able to fix it though and now have two Siemens drives working reliably. (Beware of Siemens drives.)

Modem

I recently tried to hook up my modem. In the assembly portion of the manual that came with the BB II there is a jumper diagram for utilizing SIO A as the modem port. Elsewhere in the manual, under a section called "BETTER BOARD UPDATE" there is a list of jumpers for configuring a modem that is the opposite of that in the assembly manual. In the very next paragraph after this list it reads: "To connect a serial terminal to the Better Board, install the Modem jumpers. To connect the Better Board to a modem, install the Terminal jumpers." Confusing.

As usual, when I get confused about configuring or don't quite understand something, I look through my issues of Micro C to see if I can find a reference. In issue #9 there's a short article called Talking Serially by David Thompson. All became clear. Like most Big Board owners, I am not an expert in this field; it's my hobby. Without Micro C, I wouldn't have had a BB I that worked and would not have bought a BB II.

YAM and MODEM7 configured for the BB I will not run on the BB II. The port numbers are different. (User disk #14 has BB II modem software.)

Documentation

Bill Siegmund apologized for the schematics, said there would be new ones soon. When? The documentation I received has dates on it of 5/25/82 and 8/20/82. Surely there's been enough time.

The BB I documentation was sparse, but everything I needed was there. There is no list of Disk Drive error codes, there's no list of port numbers, and the jumpering info for a modem or terminal is confusing. A list of port numbers is in the ROM listing, included on disk with the system.

Bill was very helpful when I was trying to solve the problem with my drives. He spent a lot of time with me on the phone and his willingness to help impressed me.

Bad documentation is a very common complaint, and perhaps I shouldn't expect too much with a computer kit that sells for this amount of money, but I would have saved at least 30 hours if the schematics had been correct. The other missing or incorrect info would be appreciated also.

Monitor

My NEC 1201 monitor will sync up

(continued on page 26)

Pascal Procedures

By John Jones

6245 Columbia Ave
St Louis, MO 63139

Finally, after what seemed like an endless wait, I got my copy of JRT PASCAL V3.0. It seems that whatever other problems they've solved, JRT Systems still hasn't solved the slow shipment problem. On the good side, my first impression is that it was worth the wait.

This review will just hit the high points since I haven't been able to completely "wring out" the new features.

Improvements

One of the major complaints with earlier versions of the JRT compiler was that it frequently would go "off in the weeds" when it encountered errors in the source. The new compiler seems to be much more resistant to that. Listing control directives and options have been added which allow page-formatted output to the printer, disk file or console. Frequently used routines can be inserted as source with the %INCLUDE command.

Support for file (window) variables through standard GET/PUT statements has been added. Programs written for another compiler will require fewer changes for JRT.

Extensions

Standard PASCAL shows less flexibility for array handling than some other languages since the size of all arrays must be declared at compile time. JRT V3.0 has added the ability to ALLOCATE array bounds at execution time, a significant improvement. The use of dynamic arrays, though, is somewhat restricted.

Utilities

Indexed file support (single key) is provided as a set of external procedures. The demo program in the manual runs properly for me - but it is not very fast. Number output formatting similar to both BASIC's "PRINT USING" and COBOL's "PICTURE" is implemented as an external function. High speed search of memory data is also provided as an external function.

The external procedure source file generator program, CRTMAP, lets you format screen displays with simple, high level commands.

The Manual

The User's Guide has been expanded by about 50 percent. Most of the expansion is coverage of the additional features and utilities. However, there is a new section on common problems (many of which I had to discover on my own with V2.1) and an expanded introduction. Plus, the 16-page reference card makes trips to the manual much less frequent.

Overall, the package is quite improved. All but one of the programs I've been running under V2.1 compiled without error on the first attempt. The only apparent problem with the compiler I've found is that it doesn't seem able to tell when it's out of memory.

The program mentioned above would not compile correctly until portions had been moved to external procedures. Before the program was trimmed down, the compiler would drop back to CP/M or even PFM without an error message—frustrating.

In future articles I'll try to give more details on various portions of the package as space and time permit.

Tutorial

This issue's PASCAL tutorial is on loop and control structures. For all the examples, where the word STATEMENT is used, either a simple statement or a compound statement within a BEGIN . . . END construct can be used.

PASCAL provides three methods of iteration or looping. See Figure 1.

Figure 1

```
FOR I := START TO ENDING DO
  BEGIN
  .
  .
  END;

FOR INDEX := 'P' DOWNTO 'B' DO
  STATEMENT;
```

These should look familiar to anyone who has used BASIC or FORTRAN. The FOR statement is a means to execute a portion of a program a specific number of times. Unlike BASIC, the loop control limits are calculated before the loop is entered so if START **>** ENDING the loop is not executed even once.

The loop control variable, which must be a type for which SUCC and PRED are valid (integer, char, enumerated etc.), cannot be modified within the loop. Nothing equivalent to the STEP clause in BASIC is available. See Figure 2.

Figure 2

```
WHILE BOOLEAN_CONDITION DO
  STATEMENT;

WHILE NOT ( EOF(INFILE) ) DO
  BEGIN
  READ (INFILE; VARIABLE_LIST);
  PROCESS (VARIABLE_LIST);
  END;
```

The WHILE statement is used for iteration with control at the beginning of the loop. When CONDITION evaluates to 'FALSE' the loop is not executed. See Figure 3.

Figure 3

```
REPEAT
  STATEMENT;
UNTIL CONDITION;

REPEAT
  READ(INFILE; V_LIST);
  PROCESS(V_LIST);
UNTIL EOF(INFILE);
```

The REPEAT statement is used for looping when the end condition test is needed at the end of the loop. For both WHILE and REPEAT the boolean value which terminates the loop MUST be capable of alteration within the loop. If it is not, an infinite loop will result.

There are three basic methods available for controlling the execution of a PASCAL program. The IF statement is similar to IF constructs in other languages. See Figure 4.

Figure 4

```
IF BOOLEAN_EXPRESSION
  THEN STATEMENT1
  ELSE STATEMENT2;
```

The THEN clause will be selected if the BOOLEAN evaluates to TRUE and the ELSE (which is optional) will be executed on FALSE. Note that no semicolon follows the statement preceding the ELSE. If it were present, the compiler would interpret it as the end of the IF statement, not desired in this case.

Multi-way branching is accomplished with the CASE statement. See Figure 5.

Figure 5

```
CASE INPUT_CHAR OF
  'A','B','C' : STATEMENT1;
  'Z','X'  : STATEMENT2;
  'Q'   : STATEMENT3;
  'Y'   : STATEMENT4;
  ELSE  ; STATEMENT5;
END;
```

For most PASCALs, the case selector must be an ordinal expression. The ELSE is an extension (for JRT and others), other compilers have no provision for a non-match or use the word OTHERWISE. JRT PASCAL is unique in that expressions may be used as case labels. For example, see Figure 6.

Figure 6

```
CASE BOOLEAN OF
  (SALARY > 0) AND (SALARY < 10000.0) :
    TAX := 0.10 * SALARY;
  (SALARY > 10000.0) AND (SALARY < 20000.0) :
    TAX := 0.20 * SALARY;
  (SALARY > 20000.0) AND (SALARY < 100000.0) :
    TAX := 0.30 * SALARY;
  ELSE  TAX := SALARY - (SALARY / 10.0);
END;
```

Finally, PASCAL does allow the use of GOTO. The destination statement must have a label, and all label values must be declared. See Figure 7.

Figure 7

```
PROGRAM XYZZY;
  LABEL 10;
   .
   .
  IF WIZARD THEN GOTO 10;
   .
   .
  10: PERFORM(MAGIC);
   .
```

I avoid using GOTO statements but am not adamant about it. You'll find that in most cases, programs will be easier to understand and follow without GOTO's.

String Handling

One of the criticisms of JRT PASCAL is that it implements non-standard features in a non-standard way. Leaving the validity of the argument aside, one of the problems with JRT's implementation of the non-standard type STRING is that string variables cannot be read from files, only from the console.

The BOOLEAN function getstring presented here can be used to read a dynamic string from a file. The file is assumed to be a standard ASCII text file with lines terminated with carriage return or linefeed (or both) and end-of-file signalled with cntl-Z. The constants can be changed if your operating system uses different values. The file should be RESET (opened) in binary format.

The function is equivalent to the statement READLN (STRINGVAR); for a console read and will return TRUE on end-of-file. The function could be used in any program where text data needs to be manipulated on a line-by-line basis, see Figure 8.

■ ■ ■

Figure 8

```
(* This type declaration is needed in the main program *)
type text_file = file of char; (*JRT does not recognize 'TEXT'*)

function get_string(var f:text_file; var line:string): boolean;

 const (* constants for normal line & file delimiters *)
       cr = 13;
       lf = 10;
       endfile = 26;

 var
       ch : char;
       i : integer;
       new_line : array[1..256] of char;(* long enough for most *)

begin
 new_line := ' '; (* clear assembly variable *)

 repeat   (* read chars til first non-terminator *)
       read(f;ch);
 until not ( ord(ch) in [cr,lf]);

 i := 1;

(* now read characters until get terminator *)
 while not ( ord(ch) in [cr,lf,endfile]) do
       begin
       new_line[i] := ch;
       i := i + 1;
       read(f;ch);
       end;

 line := copy(new_line,1,i-1); (* assign to dynamic string *)

 get_string := ( ord(ch) = endfile ); (* then assign EOF *)
end;
```

Overbeek Enterprises

OVERBEEK ENTERPRISES is rapidly establishing a broad selection of inexpensive CP/M programs. We will be adding new selections continuously– as fast as we can bring them into the market, while making absolutely sure that each one is a real bargain. Substantial quantity discounts are available. Our average time to process an order is 2 days. If for any reason you cannot make one of our products run on your system, we will refund the purchase price.

$29.95 Menu-Plus

Menu System

You've probably heard about the glories of menu driven systems. This powerful package developed by Capacity Inc. makes the ease of menu driven systems affordable to any CP/M user. Menu-Plus allows rapid, easy configuration of simple or hierarchical menus. You can hide the complexities of CP/M and allow single keystroke invocation of your programs. It's a user's dream. Both beginners and experienced operators will find Menu-Plus a significant enhancement. You do not have to be a programmer to install or tailor Menu-Plus on your system. In just a few minutes, you can easily create whatever new menus you desire with a text editor. Our competitors offer products in the $75-$150 range. We invite comparison of this product with any other menu system in terms of features or user friendliness. In terms of price there is no comparison.

$29.95 WSMX80

Wordstar/Epson Print Processor

Can you print these on your Epson?

$$R = \frac{e^{-2\alpha t}}{(2\pi \sin\theta)^2} \times t_3/\tau \qquad M = \int_{t=o}^{\infty} a \, d\phi$$

WSMX80 has been created for WordStar users that have an Epson MX-80 or MX-100 printer equipped with either the Graftrax or the Graftrax Plus option. By separating the printing function from WordStar, all of the features of the printer can be used to full effect. Now you can use alternate character sets, compressed fonts, italics, and a variety of other features based on the Graftrax capabilities. WSMX80 has been widely used at the University of Kansas for over a year and has proved to be an invaluable tool.

$29.95 Micro-WYL

Text Editor

Tired of trying to use ED under CP/M? Here are just a few unsolicited quotes from our customers:

"I operate a software house in Central California and have used a lot of text editors over the years. Micro-WYL has to be one of the best I have ever used, regardless of price."

"Micro-WYL is undoubtedly the hottest bargain on the market."

"Thank you, thank you, thank you."

"This editor is perfect for writing in nearly any programming language. [I] . . . have no hesitation in recommending it to anyone whose requirements match the capabilities of this inventive piece of software." *– From a review in Infoworld (11/15/82)*

Now you can have the convenience of WYLBUR on any Z80 CP/M system.

$29.95 Disk Inspector

Disk Editor

Have you ever been unable to read a file due to a bad sector? Have you ever erased the wrong file? Disk Inspector acts as a full-screen editor for diskettes. You can simply watch as sectors are displayed on the screen in both character and hex formats. When you wish to make the display pause, touch the spacebar. If you wish to alter a sector, it is a simple matter to move the cursor over the appropriate character, alter it, and have the sector rewritten.

Although Disk Inspector runs only on Z80 CP/M systems, you can inspect and alter normal (non CP/M) Apple diskettes, as well. The disk drives may be single or double density, single or double sided.

Note: Disk Inspector requires an 80x24 screen on your CRT.

☐ 8" SSSD
☐ Apple/Softcard
☐ KAYPRO II 5"
☐ NEC 5"
☐ Zerox 820
☐ ALTOS Series 5
☐ Televideo TS-802
☐ Osborne
☐ Superbrain
☐ Heath/Magnolia
Northstar 5" DD
☐ Advantage
☐ Horizon
☐ Morrow Micro Decision

Amount:

$29.95 for Menu Plus _____ $29.95 for Disk Inspector_____
$29.95 for WSMX80 _____ $2 for postage & handling_____
$29.95 for Micro-WYL _____ **TOTAL** _____

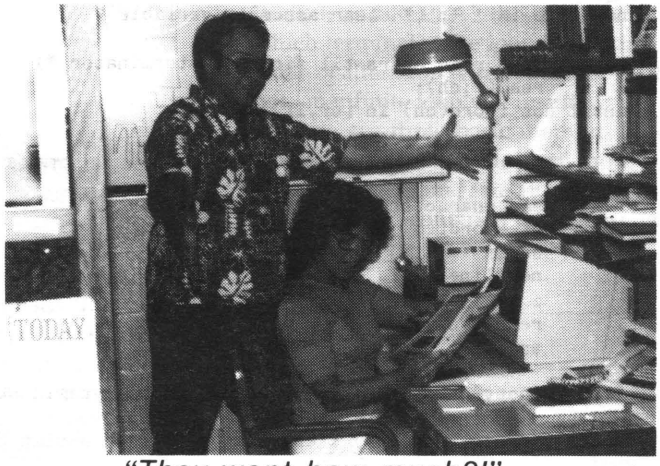
Name_____

Address_____

City_____ State_____ Zip_____

Make your check payable to: **Overbeek Enterprises**
P.O. Box 726
Elgin, IL 60120

To order C.O.D. or with a MasterCard or Visa call
312-697-8420 between 9 am and 5 pm (CST)



"They want how much?!"

On Your Own

By Guest Columnist
Hampton Miller

Computer Consultant
PO Box 816
Carpinteria, CA 93013

Few books are available to help you if you are providing a service. I suppose that this is the case because the books wouldn't be very long.

Contract programming, for example, requires no store front, no inventory, and none of the other trappings of typical business. Your expertise is all the ticket you need for this kind of self-employment.

Self-employment used to be the normal thing and people lived and died on the basis of their own decisions and "Acts of God." However, these days, people figure it's best to be secure so they attach themselves to a large organization. In fact, self-employment is so unusual now that we're called something different—like "Entreprenuers" (which is French for Broke).

Rules

Let's discuss some of the rules found in various guides to self-employment. All of them make sense and after I've broken them (with painful results) they've made even more sense.

1. There are many things you can do for free—other people's work is not one of them. Most people seem to believe that "self-employed" means "unemployed" and "independently wealthy." Their interesting projects which do not pay, DO NOT PAY! You will be very sad, indeed, when the bills arrive.

2. Get paid. Some consultants are embarassed about arranging payment. If you're one of them, change or get out. You know you are worth it—if they don't agree then you don't really want to work for them.

Charging for time and materials (T&M) can be a gold mine but most of my clients have been burned badly this way. So I propose T&M for the up-front analysis which establishes the milestones. This way I root out all the necessary resources in the company, get a good overall picture of the project and get paid for doing it.

With the project clearly laid out, then you and the client can more easily agree on a reasonable fee and time schedule.

Be sure to set partial payments at the milestones with a balloon at the end. This way you can survive along the way while the client still retains control (the balloon) should he not be pleased with the job. If the milestone payments are enough to live on, then you can afford to walk away from a really bad situation.

Be careful if you are performing a service through a broker or service company. Specify that you get paid upon client acceptance. Otherwise, you might wait two months for payment.

3. Don't let them make their problems your problems. Watch out for "we're sorry but after paying all the fixed costs, we don't have enough for you right now" or the ever popular "we should get some money real soon now." Make it abundantly clear to them that you are very much a fixed cost which must be dealt with up front.

4. Put it in writing. You don't have to have a full blown contract, but writing out all the details as you understand them makes it easy to discuss things with the client. It's a lot more pleasant to find misunderstandings early than after you have spent months building something. Of course, if the client agrees to all the points, signs the document, and then disregards portions—it's time to find someone else.

5. Have more than one client. There are three reasons for this.

First, it gives you a lot of freedom in choosing what you want to work on (it's easier to say no to an unpleasant project if someone else has a project waiting).

Second, you can increase your pay by letting clients bid against each other for your time. Let them set your hourly rate.

Third, the IRS can make things pretty tough if they think you are an employee rather than an independent contractor. If you have more than one client, they can't complain.

6. Don't burn any bridges. Your best future clients are your past clients as long as you don't make waves. i.e. don't say anything controversial such as promoting self-employment to your clients' employees. Just do the work, submit your invoice and get out!

Editor's note:

Hampton called me the day he quit his 9 to 5 to begin consulting in earnest. He was ecstatic! Later he called looking for a shoulder to cry on. He had set up milestones for a project but wasn't going to be paid until he had finished (and part of it was taking longer than he had figured). At that point I asked him to do one of the "On Your Own Columns."

When he sent this article, he sent a real bonus. In the margins were scrawled some intriguing comments which I've taken the liberty of excerpting:

"Things are going much better now with money coming in at last! Last week we were down to NO money, NO food, and NO working car (or gas)."

"This is repeat (of) material (in Micro C) but these few points make all the difference between real success and failure. Paper successes can get VERY hungry!"

"I'll be in touch and am looking forward to the big whoop de doo!"

You'll all get a chance to meet Hampton and his wife at the SOG.

■ ■ ■

FORTHwords

A Column by
Arne A. Henden

7415 Leahy Road
New Carrollton, MD 20784
(301) 552-1295

This is going to be a fairly long column, covering the FORTH-83 Proposed Standard, along with a FORTH application. But first, news from the FORTH world.

FORTH Vendor News

Laboratory Microsystems has released their FORTH 2.0 for the Z-80. While 2.0 is not FORTH-79 standard, it has most of the features of that standard, such as the 1024-byte blocks. The user's manual has been expanded and reprinted on a daisy wheel. The real advantage of 2.0 is that Duncan has included a simple I/O-driven multi-tasker, allowing ten background tasks and one foreground task. The best news: the price remains the same: $50 for a multitasking FORTH!

Unified Software Systems has added hashed vocabularies in their latest release, making UNIFORTH the fastest FORTH-79 system when it comes to compilation. Readers who mention Micro Cornucopia are entitled to a 30% discount on any UNIFORTH version.

FORTH-83

I've finally received the draft proposed FORTH-83 standard. There are more changes than I expected, and many of the areas that were begging for standardization were omitted. Here are some of the details.

All truth flags are either 0 or all ones (i.e., -1). This simple change causes all kinds of problems! You can't perform operations such as " 0= VAR +!" to increment a variable by 1, and words such as UNIFORTH's MATCH and CMPS cannot be used as precursors to conditional tests like IF and WHILE.

State smart words are removed. By "state smart" I mean words that have different actions depending on whether FORTH encounters them during compilation or execution. The primary example of this is dot-quote (."). It has now been replaced with two words: ." for compilation mode, and .(for execution mode. Tick (') has been replaced by ' for execution and ['] for compilation, and now leaves a word's code field address (CFA) instead of its parameter field address (PFA).

All arithmetic divide operations are floored. This means that the result of truncating -3.6 will be -4 under FORTH-83, whereas it could have been -3 under FORTH-79 if your system truncated numbers toward zero.

Two words have their names changed for consistency: U* becomes UM*; and U/MOD becomes UM/MOD.

ROLL and PICK now have indices from 0 to n instead of from 1 to n; "0 PICK" is the same as DUP.

LEAVE has immediate action, instead of just setting the DO-LOOP parameters so that the next encounter of LOOP would terminate.

NOT now performs a one's complement of the entire 16-bit value, thereby replacing the current NOT and COM.

EXPECT no longer adds nulls to the end of the input string (yea!). A new variable, SPAN, has been added to provide the user with a count of the characters actually entered with EXPECT. WORD moves a packed string to the dictionary and always adds a blank at the end (FORTH-79 added the delimiter character).

Other new words are: 2/ for an arithmetic divide-by-2; D2/ provides the same function for double precision integers. ABORT" prints the error message following it (like .") and then aborts. #TIB indicates how many characters are present in the terminal input buffer. CMOVE> is like UNIFORTH's -CMOVE, moving a string starting at the end of the string and working towards lower memory. >BODY gives a word's PFA from its CFA.

One unclear aspect of the proposal concerns KEY and EMIT. Of course, 8-bit characters are environmentally dependent, and a transportable program should only use 7-bit characters. However, the proposal makes it sound like KEY and EMIT can only work on 7-bit characters, which would be a gross error.

Overall, the new standard is an improvement from FORTH-79, clarifying and making definitions consistent. I personally don't like the removal of state-smart words, because two words are required to do the work of one. I particularly think the new truth flag definition is abysmal, and will cause a lot of headaches in converting FORTH-79 programs over to the new standard.

The main question I have with the new proposed standard is not what changes were made, but with the areas they overlooked: floating point, strings, data base management, file systems, and multi-programming. They didn't have to define the action of the words, just standardize the names of typical operations in each area. By the time the next standard comes out, there will be such a proliferation of extensions with differing names and actions that it may become impossible to standardize.

Accessing the Big Board Video RAM

If you've read the Big Board manuals, you know that the lower 16K bytes of address space are bank-switched to select between EPROM/video RAM and program RAM. This application shows how you can gain access to the video RAM from FORTH, and gives a screen dump utility as an example.

The bank select is controlled by bit 7 of the general purpose parallel port 1C (hex). You set the bit to select video RAM, and clear it to select program RAM. Simple enough, right? The problem is that while video RAM is selected, you cannot run any program that requires the lower 16K bytes of program space. That happens to be where the FORTH address interpreter (NEXT) and most of the primitives reside. What we need to do is write a CODE word that moves bytes from video RAM space into program RAM space, and then store the CODE word and text buffer somewhere above the 16K lower limit.

The CODE word CRT >PROG shown in screen 1 moves bytes from video RAM to any other memory region. The inverse operation is much harder because of the cursor and character attributes, and is left as an exercise for the reader. CRT >PROG selects video RAM by setting bit 7 (leaving bits 0-6 alone), moves bytes, and then returns to program RAM space.

The video RAM starts at 3000 (hex) and continues for 3K bytes (24 lines of 128 characters, labelled 0 through 23). When first accessed, line 0 of the screen (top) is stored starting at 3000; line 1 at 3080, line 2 at 3100, etc. As you enter lines, a carriage return moves you down on the screen until the bottom line is reached. The next carriage return causes scrolling—the top line disappears, the remaining 23 lines move up, and a blank line with cursor appears at the bottom.

You could perform scrolling in software by moving 23 lines of bytes starting at 3080 down to 3000, and blanking the line at 3A80. Instead, the Big Board uses hardware assist with scrolling. A "register" contains the RAM line which should appear at the bottom of the screen (initially 23), and when decremented, moves lines toward the top with the old top line wrapping around to the bottom. This movement only occurs in the video