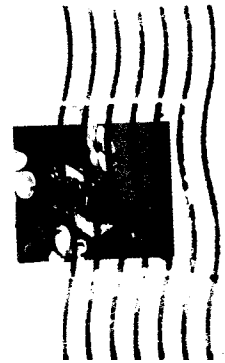
Hawthorne
Technology
1411 SE 31st
Portland, OR 97214



68·KNEWS

Volume 3, Number 1 May 1988

Hawthorne Technology, 1411 S.E. 31st., Portland, OR 97214
(503) 232-7332

Many things have been happening at Hawthorne Technology recently. We have moved Hawthorne Technology and are settled in our new location. We intend to continue putting out a newsletter as regularly as possible and get K-OS ONE articles published in other places. We now have a better area for hardware design and testing. We will continue to announce new products that run K-OS or programs that run with K-OS.

A group in Virginia is involved in porting HTPL and the K-OS ONE system to run on the National 32000 series micro-processor. While the object code produced will not be compatible with the 68000 version the HTPL source code and the manual should be almost identical. If there is enough interest we can convert the K-OS operating system to run on almost any 32/bit processor. This would make it the generic portable operating system for big micros.

We are happy to announce that we now have a full featured BASIC interpreter for the K-OS operating system. It has been written by Custom Computer Products in California. Like the disk format that draws on the MS-DOS world, the BASIC is very similar to MS-BASIC so it is possible to run most PC BASIC or CP/M BASIC programs on any K-OS system. The main exceptions are graphics and sound. See the preview article on it later in this newsletter.

A video display board has been designed to fit on the expansion bus of the HT-68k. It is a very fast programable character oriented display. It also has a generic parallel I/O for scanning special purpose keyboards. This eliminates the need for a full ASCII terminal and is excellent for many process control applications. It is being offered by Haas Automation of California and looks like it would work very well for embeded control usage.

Bill Kibler of California has K-OS running on a Stride micro system. He wrote two articles for The Computer Journal about how it was done. There are also several S-100 systems running K-OS, some of them using 8 inch floppy disks in a native format. We would like more feed back on what kinds of systems have K-OS installed. A library of installations could be built up to make it less effort to get K-OS ONE running on one of a kind systems.

We don't have a C compiler yet. There are several reasons for this. First, the existing assembler, HTPL, and FORTH have proven to be very good and effective for development. Secondly the ease of using a cross compiler that runs on a PC has made that route very convenient for larger projects using the C language. All the people who have started working on C have found that a C compiler is a much larger and more difficult project than they thought it would be. Don't give up hope. If you really want C you are not alone. It still looks like we will get a C compiler. Possibly soon. We haven't worked on a C compiler because we didn't want to duplicate the efforts of those who are working on one.

We have been working on FORTH this winter. We have a new version of FORTH with full floating point support. This will make it easier to use for numerical problems. We also have added a CASE structure to our regular FORTH to make programming easier. Another addition to both versions of FORTH are words for defining abstract data structures. Now you can do things in FORTH as complex as anything in C.

In the next couple of months we plan to have a version of the command processor that has all the wild card characters just like CP/M and MS-DOS. We have part of this done already. This will greatly simplify copying files and managing disks. We will also have some kind of batch processing capability. We haven't done much with batch processing but the operating systems calls for doing it have always been a part of the operating system, it was just a matter of time before we used them.

The big talk in computers this year besides the 80386 has been RISC processors. These are simplified processors that are easier to design and build then the more traditional complex insruction set processors. The language we used for K-OS ONE, (HTPL), is like the software equivalent of a RISC machine. The structure was kept simple but it was powerful enough to do anything that needed to be done.

If you have a piece of code that you have written for your K-OS ONE system and would like to share it with other K-OS ONE users, we want to encourage you to write it up and send it to us for 68-KNEWS or to The Computer Journal for publication.

We are going to be at SOG again this year. That is the Semi Official Gathering (of computer people). It is held in Bend, Oregon and sponsered by Micro Cornucopia magazine. This year SOG will be held July 14, 15, and 16th. We always have a great time. It is worth the trip. We look forward to meeting with K-OS ONE user's and showing our new products.

TinyGiant 68000 SINGLE BOARD COMPUTER

The HT68K is a complete 68000 system on a single board. The K-OS ONE operating system comes complete and installed. The 68000 runs at 8 Mhz with no wait states. There are two EPROMs, a dynamic RAM controller, 4 RAM chips, a 68681 dual serial chip, a printer port, a floppy disk controller and an unbuffered expansion bus.

The 68681 has two serial ports, and a timer. Each serial port has in internal baud rate generator for 50 to 38.4k baud. At start up, the console is set to 9600 baud and the other port is set to 1200 baud. The timer is used for a time of day clock. When the PRN device is selected the parallel printer port is used. The 1770 floppy controller can control up to 4 floppy disk drives, either 5 1/4 or 3 1/2 inch. It has an internally clocked digital data separater. All serial and parallel I/O is done with interrupts.

The board comes with 128k of RAM that can be expanded to 512k by adding more 64kx4 dynamic RAM chips. An expansion board could be added if more memory is needed. The MB1422 dynamic RAM controller takes care of all the timing and refresh. The two EPROMs contain the boot code and a simple hardware monitor.

There are two expansion connectors that provide all the needed address, data, and control lines. The 64 pin connector has all the raw signals just as they are on the 68000. The other expansion connector has a priority interrupt encoder that is connected to autovector interrupts.

To use the TinyGiant, connect a power supply, a terminal, and a disk drive. The board requires only 5 volts at 2 amps and +12 volts for the RS-232. The -12 for the RS-232 is generated on board. The power connector is the same as is used for a floppy disk. The size and shape of the board (5.75 by 8 inch) allows it to be mounted on the side of a 5 inch disk drive.

The PROMs are set to boot K-OS ONE from disk when power is applied if a drive is connected. If no drive is connected then a simple hardware monitor is started.

The purpose of the monitor is to aid in troubleshooting or debugging the hardware and how it is set up. The monitor can examine and modify memory. It can also read individual sectors on the disk. It can boot the K-OS ONE operating system or any other program you want. It can also down load a program in HEX from the Aux serial port. There is also a printer test program included.

SuperBASIC 68K

Custom Computer Products has produced a full featured BASIC for use with the K-OS ONE operating system. The basic is very similar to the Microsoft BASIC that was used on CPM systems, GW BASIC or BASICA used on the PC. This combined with the use of the MS-DOS disk format makes a large number of commercial and public domain programs available for use with any 68000 computer that has the K-OS ONE operating system. If you have a library of BASIC programs on CP/M all you have to do is translate the disk format, if you have BASIC programs already on MS-DOS disks you don't even have to do that. There will probably be small things that need to be changed but most will run directly on the Custom Computer Products BASIC. This form of BASIC was chosen because it is the most compatible with the programs that exist on our disk format.

Most people have expected BASIC for the 68000 to be very large. This one is quite small. Only 32KB of object code. This leaves you with over 32k of work space on a 128k standard HT68K. It is small because it was crafted by an expert in assembly language. You can run fairly large programs on a standard 128K TinyGiant that hasn't been expanded. Even though it is small it has all the features found in much larger BASICs. There are full string functions, file I/O and floating point math. There is a peek and a poke command for direct hardware control.

The floating point math pack is the major difference between the Custom Computer Products BASIC for K-OS ONE and a Microsoft style BASIC. The 32 bit single precision math uses the same format as the HTPL math pack and gives 7 digits with a range of 10E19. The double precision math format is 8 bytes with 14 digits and a range of 10E4900. This is a very long range but both formats were chosen to give fast results on a machine that does not have a floating point coprocessor. All trancendental functions are provided in double and single precision format.

The programs can be saved or loaded in ASCII or a compressed format. The ASCII files can be created or edited with the standard line editor or with the screen editor in the tool kit. The current built in editor is an improved line editor. If you are converting from an older BASIC an ASCII file must be used because while the source is compatible the internal code is quite different.

The sequential disk files that are recorded in ASCII by Microsoft style BASICs are compatible and can be read or updated for with SuperBasic. The random disk files will need to have numeric values converted because the floating point format is different. Integers are also stored in the opposite order from the intel style processors.

SuperBASIC 68K

Supports almost all of the standard MS BASIC syntax including FIELD, GET, PUT etc.

LOAD and RUN standard MS BASIC programs without change.

Reads and writes data files in MS BASIC compatible format so can interchange data with your PC.

Numerical data includes integer, single precision floating point and 64 bit double precision floating point for both arithmetic and called functions such as SQR, SIN, EXP, etc. Automatic type conversion is provided for mixed expressions.

Program variable names can be any length up to 255 characters, all characters are significant.

Strings can be any length up to 255 characters. A full set of string functions is included.

A modern memory manager is used to allocate string and variable storage from dynamic memory to eliminate the need for the garbage collection FRE function to release string space. This can greatly improve throughput for some applications.

IF, THEN, ELSE structures supported as well as WHILE and WEND.

CHAIN function provided to link to supporting programs.

AUTO line numbering and RENUM are provided for easy program entry.

An EDIT function allows WYSIWYG (what you see is what you get) editing of program lines with natural edit keys used throughout.

Easily configurable for most common terminals.

Peek and Poke for direct hardware control.

HT-Debug

When writing or modifying a program, a low level debugging tool is often needed. To make it easier to correct bugs or find them during program development we created the HT-DEBUG program. This can also be used to study or modify programs or debug hardware. It can also do other low level tasks.

HT-DEBUG allows you to make small changes to a program that has been assembled or compiled and then run the program in a controlled manner. Programs can be loaded or saved in three formats, Motorola S records, Intel style HEX records, or binary image .BIN files. A program can be loaded in one format and then saved in a different format. Because of the use of the operating system programs can be save on disk or uploaded to a host computer.

It has commands that fill blocks of memory, move blocks of memeory, or compare blocks of memory to see what has changed. Memory examine and replace covers long words, words and bytes. It is possible to display the contents of memory in hex or in ASCII. You can also look at and change the contents of registers.

The disassembler allows you to see what code is at a particular place in memory. With the single line assembler, you can patch the code using 68000 mnemonics. Breakpoints can be set to stop a program and display what was in the the registers when the breakpoint was encountered. The HT-DEBUGGER also has a trace mode. Individual registers can be examined or set. The memory can be displayed as HEX or ASCII characters.

HTPL Floating Point

The HTPL floating point package is designed to add floating point arithmetic operations to the standard HTPL language. The package contains the floating point routines in a linkable .HEX file that is about 3.5k long. The parameter stack is used to pass all arguments. It does not use any variable space. There are routines to convert to or from integers or ASCII characters.

The floating point numbers are 32 bits long. The format is the same as the Motorola Fast 32 floating point math. This gives 7 digits of precision and a range of E+/- 19. Since the floating point values are 4 bytes and HTPL has long stack items, you can use the same load, store, and variable declarations as long integers.

The HTPL floating point package includes: Sine, Cosine, Arc tangent, exponential, logarithm, and square root functions. A simple RPN calculator program to evaluate the math package is included. Also a sample program that demonstrates the use of the formatted output routine is included.

Screen Editor Toolkit

A very good line editor comes with each copy of K-OS ONE that will work with any terminal without modification. While some people like line editors many people prefer the convenience of a full screen editor. Many programmers like to customise an editor for their own special use. The available editors were written in C or some other language that is not available for K-OS ONE yet. To solve this we created our editor tool kit using HTPL.

There are three programs in the editor tool kit for K-OS ONE: a full screen text editor, a line oriented editor, and a text formatter. A ready to use binary object copy of each program is included. The HTPL source code for each program is included so you can modify any of the programs to fit special needs. The manual has a user section and modification section for each of the included programs. The operation of all procedures is explained. The purpose and use of each variable is also explained.

The screen editor uses Wordstar(tm) style commands and can be configured for almost any type of terminal. It is fast and can handle large files. The output is plain ASCII text. New features can be easily added or the function of existing features can be changed. The command codes can be changed to emulate most editors. Because of the disk format used, it is easy to move text files to or from from any PC.

The line editor is like the editor provided with the K-OS ONE operating system. A line editor doesn't need to be customised for a particular type of terminal. It is easy to specify changes that affect groups of lines. Also many different lines from different parts of a document can be compared at the same time.

The text formatter is a nroff style formatter. The text is prepared with a normal editor and the format commands are embeded in the text. The formatter prepares a disk file or sends it directly to a printer. It is useful in document preparation tasks such as for desk top publishing. It is easy to adapt it to use all the special features of a printer such as special spacing or character fonts.

These are all compact and efficient programs. The source code for the screen editor is only 25k, the line editor 19k, and the text formatter 13k. The small size and modular code make them very easy to understand. The object code for the screen editor is 14k, the line editor is 11k, and the text formatter 8k. This small size gets more done with less memory or disk storage space.

K-OS ONE, 68000 OPERATING SYSTEM

K-OS ONE is a single user operating system. It has all of the standard commands that you need to use a 68000 system. K-OS ONE uses the same disk format as MS-DOS including subdirectories. This makes it easy to exchange data with any PC. The commands are patterned after CP/M and MS-DOS/PC-DOS, so the operation will be familiar to anyone who has worked with either of those systems.

A simple design was used to allow implementation of the operating system on most 68000 hardware. System calls use a parameter block instead of registers. This makes it easy to use system calls from a high level language. K-OS ONE is set up so OEM's can use it with their products or applications. The command processor is a separate part and can be changed to resemble almost any operating system, (even UNIX). The operating system is very small. It can be edited and recompiled in a system with 128k of memory.

A complete package is provided. It includes an editor, a compiler, and an assembler. The source code is provided, making it possible to modify and maintain the operating system with the tools that are provided. The operating system is written in a high level language to keep it managable. K-OS ONE has a RAM disk. A sample BIOS and a boot loader is included to make it easy to install on different systems.

The K-OS ONE operating system package is low priced, even in single piece quantities. Manuals are available for a small fee without purchasing the system.

The standard generic K-OS ONE 68000 operating system package that can be installed on any system includes:

Operating system (HTPL source and patchable binary object)
Command processor (HTPL source and binary object)
HTPL compiler (object only)
HTPLRTL runtime library (ASM source and hex object)
68000 Assembler (object only)
Line Editor (Object only)
Sample BIOS and boot loader (ASM source)
Manual (bound hard copy)

EDITOR

A line editor is included in the K-OS ONE distribution package because it can work with any ASCII terminal without special installation. It provides basic text entry and editing for program development. A find all command acts like a cross refrence. Find and replace commands can reference position or context.

ASSEMBLER

The assembler is a complete, classic, two pass 68000 assembler that runs with the K-OS ONE system. The assembler uses all the standard Motorola mnemonics. Listings can be directed to a printer, or a disk file, or the console. A sorted symbol table and cross referencing is optional. The error messages are descriptive and are included in the listing. If the no list option is selected, error messages will be displayed on the console device unless output is directed elsewhere.

HTPL

HTPL, Hawthorne Technology Programming Language, is a language that was created to write the K-OS ONE operating system. It was designed for writing compact readable code with a small efficient compiler. The runtime library written in assembler is less than 2k long. It is two pass and directly produces a position independent executable binary program.

HTPL uses control structures like those in MODULA-2. Complex programs are easily broken down into small easily understood procedures. Procedures can be forward refrenced. Constructs include IF-ELSE-ELSIF-END, WHILE-END, DO-UNTIL, WHILE-DO-END. There is also a CASE structure. Modules written in assembly language can be linked into HTPL programs.

The expression are handled in RPN (Reverse Polish Notation) which translates into compact, efficient object code using a small compiler, less than 21k. Variables are easy to declare and use, making the source code easy to read. Variables and arrays can be byte, word, or long. Variables can be initialized or allocated at run time.

Using HTPL, a person can create menu driven user oriented applications. The same language can be used to create compact systems programs. Almost all the K-OS ONE operating system, the edit tool kit, the assembler, and the HTPL compiler was written in HTPL.

8086/8088 ASSEMBLER

The HAWTHORNE TECHNOLOGY 8086/8088 assembler can be used to develop 8086 programs on any microcomputer with MS-DOS 2.0. The assembler was developed to easily create .COM programs or for embeded systems. It is optimized for small programs. The two pass assembler creates .HEX files without the need for a link editor. The source file is read from disk. An object file in Intel hex format is written to the disk. The HEX object file can then be converted by a utility program, (included with the assembler) to prduce an executable .COM file.

The assembler is quick and easy to use. It does not require complicated directives such as "ASSUME CS: CodePlace" or "CodePlace SEGEMT AT 0100H". It is not strongly typed so a variable may be referenced as a byte or word with equal ease. This assembler was designed to be an efficient tool for the busy programmer.

68000 CROSS ASSEMBLER

We are now releasing our 68000 cross assembler. Because we are doing a lot of our current programming in assembly language, we have finished the cross assembler. All of our early work with the 68000 was done on a PC until we had the K-OS ONE system going. When we first wrote it we didn't have any time to spare. We didn't need all of the features of a full 68000 cross assembler so the time was not devoted to making it into a completed product. While finishing it we made sure it was compatible with the resident assembler on K-OS ONE. We use a PC to assemble and list many of the programs we develop. The compatible disk format makes this combination work very well.

The cross assembler is written entirely in 8086 assembly language so it is small and fast. It is only 15k of code. All input and output is done with standard MS-DOS calls so it will run on any MS-DOS system even those that are not totally PC compatible. The manual describes the assembler and how to use it. It does not try to explain how the 68000 opcodes work.

All 68000 and 68010 instructions are supported. The cross assembler has conditional assembly. The symbol table is in alphabetical order. Cross refrencing is included. There are include files so it is easy to assemble big programs but edit them in small pieces. An equate file can be produced for PROM based programming.

CROSS ASSEMBLERS

A HAWTHORNE TECHNOLOGY cross assembler allows the creation of assembly language programs for the specified microprocessor on any microcomputer with MS-DOS 2.0 or later operating system. Each assembler is a classic two pass assembler that uses the manufacturers nmemonics. Source files are read from disk. A machine language object file is generated and stored on the disk in Intel Hex format. The resulting file can be burned into PROM or downloaded to another computer through a serial port.

These directives are present in all versions. Some processors have additional directives.

DB	Define Byte	DS	Define storage
DW	Define Word	END	End of input file
PAGE	Start new listing page	LIST	Listing on
SYMOFF	Turn off symbol table	UNL	Listing off
TITL	New title for each page	ORG	Origin
INCLUD	Include file	EQU	Equate

Listings may be directed to a printer or the console. The cross assemblers include the option to print an alphabetical symbol table. A complete cross reference of all symbols and their use can also be requested. Error messages are descriptive and are included in the listing of the line following the occurance of the error. If UNL (listing off) is selected, any error messages will be displayed on the console device unless output is directed elsewhere.

The assemblers generate an Intel style .HEX file as output. Utilities are included to convert the .HEX files into binary files or the Motorola style S records.

Each package contains an MS-DOS format, 51/4 diskette and a complete manual that contains a list of the processors assembly language commands.

Currently available for the following processors:

6502	6800/6801	6805	9900/9995
8080/8085	Z80/64180	8051	

Floating Point HT-Forth

FLOATING FORTH is a special version of HT-FORTH that has single and double precision floating point operation words added to it. All the words that are in the standard HT-FORTH are included in the floating point version. The kernel size is only 18k and is supplied as a ready to run binary. There is no source code of the assembly language portion provided. The utilities, full screen editor, and 68000 RPN assembler are provided in source form and are the same as for the standard HT-FORTH. Like the regular FORTH, programs can be saved in a modifiable form or in a frozen form. When stored in a frozen form, a program you write can safely be distributed without royalty payments.

There are two floating point formats. First is the single precision format that takes 4 bytes of storage. This is like the Motorola fast floating point format. Because HT-FORTH is 32 bit all the normal words can be used with single precision floating point values. The other format is an 8 byte format. This has 48 bits for the fraction and 13 bits for the exponent. This allows about 14 significant digits and an exponent range of several thousand. There are some 8 byte operators added to take care of normal operations like @D !D DOVER DSWAP DVAR. There are words for many double precision constants.

There are the standard floating point operation words. There is also reverse subtract and divide to avoid swapping the top of the parameter stack. There is a complete set of functions for both single and double precision numbers. This includes sin, cos, arctangent square root, log, exponential and comparisons. There are words that can convert single to double and double to single.

A format word makes it very easy to create floating point output for display or writing to disk files. A table of format codes is created and the FORMAT word interprets the result much like FORTRAN uses FORMAT statements with Read and Write statements. This reduces the size and complexity of code needed compared to formatting one output value at a time. Integers and strings can be mixed in the format tables. There are immediate words for convenient use of floating point constants or interactive use.

HT-Forth

HT-FORTH is a full featured, interactive FORTH that works with the K-OS ONE operating system. It uses a full 32 bit stack and 32 bit arithmetic to take full advantage of the 68000 microprocessor. Programs are position independent and are limited in size only by the memory available. Source code compiles to inline macros, JSR or BSR so there is no inner interpreter overhead.

Many major extensions have been added to normal FORTH. There are words for working with strings just like in BASIC. There are words to access the file system in K-OS ONE. There are words that provide pointers like the C language has. There is a CASE structure. There are words for declaring complex data structures. There are about 250 words in the basic FORTH. This is more effective words than it seems because there are no hidden words for use by the compiler.

Standard ASCII files are used. The HT-FORTH package has a Wordstar(tm) style full screen editor and a FORTH style 68000 assembler. A lock program can be used to create .BIN files for distribution without paying royalties. A person using the program does not need to know anything about the language used to produce the program.

Source code is provided. The kernel is 68000 assembler code. This is the normal two pass assembler and not an RPN FORTH style assembler. Because of the structured method of writing the kernel routines and the provision of source code it is easy to add new words or delete existing words. Most words can either be inline macros for speed or subroutines for a more compact size. Many of the utilities like the editor and assembler are written in FORTH and source is provided for them also.

Utility words are provided for using all the K-OS ONE operating system calls. These include make file, open file, read file, write file, close file, and delete file. This makes it possible to easily work with normal ASCII or binary files and share data with programs written in other languages. Using these words results in programs that are portable to any hardware using the K-OS ONE operating system.

FORTH is an excellent choice for a programming language. It is very interactive and easy to debug. It is portable to other computers and operating systems. It can be modified to fit different kinds of programming and programming styles. It is very good for interactive or control type programs.

HTPL ROUTINES

The following routines are useful for programming in HTPL. These routines perform functions similar to some of the 'words' provided with HT-FORTH. Each task is accomplished first in assembly language and then in HTPL. Pointers are a useful way to load or store a value. You then increment the pointer. A library of procedures like these can be added to the run-time library code that comes with HTPL to make programs smaller or faster.

In many programs most of the time will be spent in a very small part of the code. If these small parts are written in assembly code the entire task can run fast but still be as easy to develop as if it were all in HTPL.

In the screen editor, the replacement of a single routine made the editor appear to run twice as fast. In the assembler, the symbol table routines were written in assembly code. When we develop programs at Hawthorne Technology, we do all of the routines in HTPL first because it is a quick way to get the program running. Then, if we want to improve the speed of the program, we recode the routines that are accessed frequently, in assembly language. A routine like one of these can be treated like a component and used in many different programs.

```
;---- L@+       LOAD USING A POINTER
  MOVEA.L  (A4)+,A0      ;GET ADDRESS
  MOVEA.L  (A0),A1       ;GET POINTER
  MOVE.L   (A1)+,-(A4)   ;GET VALUE
  MOVE.L   A1,(A0)       ;SAVE POINTER
  RTS                    ;RETURN
;---- L!+       STORE USING A POINTER
  MOVEA.L  (A4)+,A0      ;GET ADDRESS
  MOVEA.L  (A0),A1       ;GET POINTER
  MOVE.L   (A1)+,-(A4)   ;GET VALUE
  MOVE.L   A1,(A0)       ;SAVE POINTER
  RTS                    ;RETURN
;---- NEXTB     POINTER ON STACK
  MOVEA.L  (A4),A0       ;COPY ADDRESS
  MOVE.B   (A0)+,D7      ;GET BYTE
  MOVE.L   A0,(A4)       ;SAVE ADRES
  MOVE.L   D7,-(A4)      ;SAVE BYTE
  RTS
```

These have the same function but are in HTPL. The word TUCK is an HTPL primitive and is equivalent to SWAP OVER.

```
proc L@+          ( adrs -- value )
  dup  @4 tuck +4 swap !4 @4  end
proc L!+          ( value adrs -- )
  tuck @4 tuck !4 +4 swap !4  end
proc nextb        ( pntr -- pntr byte )
  dup @1 swap +1 swap end
```

PRICE LIST

Single piece pricing 5/88

HT68K Single Board Computer with K-OS ONE	395.00
RAM expansion to 512k (12 64k x 4 chips)	call
K-OS ONE operating system	50.00
HTPL FLOATING POINT	25.00
HT-FORTH	100.00
FLOATING FORTH	100.00
EDITOR TOOLKIT	50.00
HT-DEBUGGER	50.00
SUPER BASIC	150.00
LIZARD LAND Adventure Game	10.00
8086/8088 Assembler	50.00
Cross Assemblers	50.00
Any manual	10.00

We accept VISA, MC, C.O.D. or Prepaid orders.
Please call for OEM pricing and terms.
We ship UPS or US MAIL.

Normal hours are 8:30 to 5:00 pacific time.
We are sometimes available during additional hours.

LIZARD LAND Adventure Game

This is a simple adventure game provided in source and object form. It is a good example of how to do an interactive program in HTPL. Extensive use of string manipulation is demonstrated as well as a natural language command processor. The general format can be adapted to other games or to command processor driven applications.