68-KNEWS

Volume 2, Number 2 September 1987

Hawthorne Technology, 8836 SE Stark, Portland, Oregon 97216
(503) 254-2005

NEWS and NOTES -- Marla Bartel

I think everyone had a great time at Micro Cornucopia's SOG conference in Bend this summer. They host a great gathering. Joe and I packed a couple of TinyGiant systems in the car along with several boards boxed up to sell, a lot of software packages, manuals, literature, clothes for the two of us for a week, and a programmers picnic basket (survival food). Again I was thrilled with the TinyGiant. There was still room for us! (You should have seen the car a year ago, with the system we were using then.)

We checked into our dorm room to find we were right next door to Art Carlson, the Editor of The Computer Journal. Several hours were spent discussing computer magazines in general, the history of some of the biggies such as BYTE, and where the industry is going. A major question was what sort of articles do people want to see. The next was, who is going to write them. Anyone with opinions on the topic is encouraged to let us know. You can send comments (or copy) to us at Hawthorne Technology, or directly to The Computer Journal.

I was 'volunteered' to give a talk at SOG on how to install a TinyGiant in a system. It is such an easy thing to do that I wondered what I could say to a room full of engineers that they didn't already know. I covered some cabling techniques and static protection (which engineers sometimes get sloppy with). We had a good discussion going when they told us the hour was up and we would have to give up the room to the next group. With years of training and experience, there is always more to learn if you are willing to listen to people who have had experiences different than your own. Joe gave talks on HTPL programming and on the K-OS ONE operating system. Both were well attended.

In exchange for speaking, Joe and I were given whitewater raft trips. It was my first raft trip, and I had a great time. I am not big on water sports, but I will jump at the chance to go rafting again.

At the show, we met several of our customers and gained several new ones. Evenings were spent discussing product lines, manual production techniques, the state of the education system (there were several instructors in the group), and deep subjects such as what makes one American beer any better than the next. We checked several brands and I didn't find anyone too concerned with finding an answer. We were all grateful that the our rooms were just down the hall and no one needed to drive.

One discussion that I found particularly interesting had to do with illiteracy and the use of icons in computing. You don't need to be able to read to use some computers anymore, but what are we computing?

As a child, I remember my father coming in from the farm to say the new tractor didn't have numbers on the gear shift knob to identify the different gears. With the large non-english speaking population in farming, this major farm machinery company had decided to use icons for clarity. They had little pictures of a turtle and a rabbit. (You use turtle gear to go slow.) Everyone seemed to adapt to this quite readily. It was several years later I read in a business magazine how this same company ran into a major problem. They started exporting these tractors to countries that didn't have turtles or rabbits. Is it easier to explain a numeral sequence (even in a foreign language), or to explain: what is a turtle, what is a rabbit, and how do they move in relation to one another? I vote for starting off in gear number one. Even chinese numbers are understandable without much trouble.

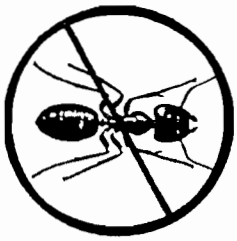
一 二 三 四
one two three four

Anyone developing with icons must be careful to realize that due to life style or experiences, people using the machine might not make the same associations that the designer does. Symbols are something we learn and are not always universally recognized. Another example of this that was mentioned was a popular icon based system that used asparagus spears. If you got 3 to 5 asparagus spears on the screen, the machine locked up. The manuals didn't explain this, but to the author, asparagus spears must have represented BAD things. What does this mean to an operator who likes asparagus?

I want to thank all of the people who responded to Tony Ozrelic's survey. We had a chance to talk with him when we were in Bend. It looks like the Bolo system is not going to be showing up on the market right away. No one computer system will please everyone, but to have a viable market, you need enough people wanting the same thing to support a design (and the company producing it). I haven't seen the actual results, but it sounded like the responses were not concentrated in any area. That and other demands on his time had Tony sounding like the Bolo project has been tabled.

68-KNEWS has gone international. We have sent copies of this newsletter to individuals around the world who have responded to our ads. We have also been contacted by a user group in Australia. They will be reprinting portions of 68-KNEWS in their local club newsletter. If any of our readers in a 68000 interest group would like to use portions of the 68-KNEWS in their newsletter, please contact us.

We have started working on some assembly language programs again. There are times when this is necessary. I am not doing any of the coding so I have an impartial view of the changes this makes around the shop. When we are doing development in HTPL, the programmers are at their development machines, working away. When we switched back to assembly code, I first noticed that there were line printers buzzing away for long periods of time. (When you are trying to talk on the phone you notice these things.) Then you start to see programmers taking time off to read the paper during an assembly or a listing. They never had time for this when they were coding in HTPL. Things just don't take as long. (A case of paper also lasts longer when you use HTPL.)



THE NEW HT-DEBUGGER

A new product that we are releasing this month is one that many people have been waiting for, a full featured debugger. The purpose of a debugger is to allow you to make small changes to a program that has been assembled or compiled and then run the program in a controlled manner.

The debugger has an expanded set of features like those found in the monitor PROMs on the HT-68K. You can fill blocks of memory, move blocks of memory, and compare blocks of memory to see what has changed. The memory examine and replace has been expanded to cover long words, words, and bytes. When examining and changing bytes you can look at all bytes, only odd bytes, or only even bytes. In addition to memory you can now look at and change the contents of registers that a program is using.

Programs can now be loaded or saved in three formats, Motorola S records, Intel style HEX records, or plain binary image .BIN files. A program can be loaded in one format and saved in a different format allowing you to change the format for a program. Because path names are used with the operating system it is possible to down load files from other machines or to list the hex records produced.

A disassembler has been added. This allows you to see what code is at a particular place in memory. Many times only a very small part of a program is critical and needs to be understood to make a program work in a different manner.

To go along with the disassembler there is a single line assembler. This allows you to patch the code in memory using 68000 nmonics and is much easier than hand assembling patches.

For running programs there are now breakpoints that can be set. When a program hits a break point it will stop and display where it stopped and what was in the registers at the time the break point was hit. A trace command has also been added. This allows the debugger to monitor each step of a program as it executes. For the programs that go wild and kill memory this is very useful. You can run the program and stop it before it does any damage.

There are two versions of the debugger. The first version runs with the K-OS ONE operating system and uses all the file handling and device drivers of that system. There is also an OEM version which can be placed in EPROM and will run without an operating system.

The HT-Debugger package with its line assembler and dis-assembler fits in with the Editor Toolkit and the 68000 Assembler that is supplied with K-OS ONE to make a solid development package.

Pre K-OS Products
from HAWTHORNE TECHNOLOGY

for use with
MS-DOS

8086/8088 ASSEMBLER - - - $ 50.00

This assembler is an alternative to MASM. It is smaller (only 14K), faster (this one really moves), and easier to use. This is an untyped assembler. Variables may be referenced as byte or word with equal ease.

Source code is read from disk and an object file is written back to the disk. A utility for creating an executable file is included.

CROSS ASSEMBLERS - - - - $ 100.00

Develop assembly language programs for other processors on your MS-DOS computer. Each cross assembler produces object files on disk, in Intel Hex format.

Available for these processors:

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

Each of the above assembler and cross assembler packages are shipped on an MS-DOS, DSDD, 5 1/4" disk with a manual. They run under MS-DOS 2.0.

There are options for conditional assembly, listing direction, optional object code, sorted symbol table and cross referencing that may be requested at run time or through directives in the source code. Error messages are descriptive and are included in the listing on the line following the error. If Listing Off is selected, messages are sent to the selected output device.

Magrathea

Computer Bookstore

8842 S.E. Stark
Portland, Or 97216
(503) 254-2045

Screen Editor Toolkit

The Editor Toolkit for K-OS ONE contains three programs, a Screen Editor, a Line Editor, and a Text Formatter. Binary and HTPL source code is included for each. A few routines are in 68000 assembly code to make the editors more responsive.

* Screen Editor uses Wordstar(tm) style commands and can handle large files.
* Line Editor is source to the generic editor supplied with K-OS ONE.
* Text Formatter is a nroff style formatter for document preperation.

$50.00

Order Today:
VISA, MC,
Checks, COD
(503) 254-2045

Hawthorne Technology
8836 SE Stark
Portland, OR 97216

INSTALLING K-OS ONE

K-OS ONE was designed to be a very portable operating system that could run on almost any 68000 based computer. One question that is asked is: how do you get K-OS ONE to run on a new or unique machine?

To install K-OS ONE, you should follow these steps:

First, write a console driver and a disk driver and load them into the new machine.

Second, write the OSINIT and TRAP1 routines as described in the manual and load them into memory.

Third, load the binary copy of the operating system and command processor into memory.

Fourth, patch the addresses of the driver routines into the area just below where the operating system has been loaded. You should now be able to start the system.

After the system is started in this manner the next step is to edit the file that contains the BIOS links, assemble it, and re-compile the operating system with the BIOS links. This will provide a better system to work with and all the tools of the K-OS ONE operating system will be available for use.

To create the first drivers to get the system going a system with a 68000 assembler or a cross assembler is very useful. (Without one, the drivers would have to be hand assembled.) Because the disk format is the same, a cross assembler that runs on a PC, like the RAVEN SYSTEMS Cross Assembler, works very well.

What's new in K-OS ONE PROGRAMMING LANGUAGES

To write, you need a language. To write programs, you need a programming language. There is an assembler and an HTPL compiler with every K-OS ONE package shipped. These are the languages we used to create the package and some of our own development tools. We released a FORTH language this summer and people are starting to use it. Many large companies are using cross compilers or cross assemblers on IBM PC machines for developing programs that will later run on the 68000.

We are getting close to having two new languages to work with, C and BASIC.

Several companies have been working on porting C compilers to the K-OS ONE Operating System. In one case, the implementor started using HTPL and found that it was easier to use HTPL than to finish a C compiler. Another had to drop out when another project came off hold. The remaining companies haven't yet finished.

The BASIC that is being ported to K-OS ONE will have many features in common with Microsoft BASIC. There are many programs in MBASIC. Most are on compatible disks. Because of this many of these programs should be available for use with K-OS ONE.

The C that is closest to being done is much more than a small C. The floating point math package that was written for use with the BASIC will be used to produce a floating point C.

Both of these should be announced in the next issue of 68-KNEWS. There will also be written reviews of these and other languages as they become available.

68000/68010 CROSS ASSEMBLER

An inexpensive way to develop 68000 and 68010 assembly language on your IBM PC or Compatible.

* Generates S-Record output files
* Comprehensive listing facilities, labels to 32 characters, sorted symbol table
* Strict syntax check with extensive ERROR reporting
* Supports INCLUDE files, PATH names, both RORG and ORG mode, many other directives (except MACROs)
* NOT Copy-protected

Minimum requirements: 256K, DOS 2.XX, one 5 1/4" DSDD disk

Checks, VISA, MasterCard
MN residents add 6% sales tax

$49.95

(612) 636-0365

RAVEN COMPUTER SYSTEMS

BOX 12116
ST. PAUL, MN 55112

NON-PREEMPTIVE MULTITASKING IN HTPL

In many control applications some form of multitasking is needed. Usually there is a terminal or some kind of keyboard and display to communicate with the operator. While this is going on, a process has to be monitored and/or controlled in the background. This is different from common programs being written for a PC where only one thing is happening at a time.

Problems usually have a complex solution and a simple solution. If all of the features of a full multitasking system are not needed, a special purpose program can do the job at lower cost and with less effort. This means that an existing single user/single task operating system like K-OS ONE can often be used for multiple tasks such as character I/O and disk file management.

If a task can be interrupted at any point, complications start. Any shared routines have to be reentrant. It must be possible for a second task to start executing the routine before the first task has finished. Any shared data areas must have locks to prevent two tasks from trying to update at the same time, corrupting the data. Because the task swapping routine doesn't know anything about the task, it must save all the registers and status.

A nonpreemptive multitask system is one where the tasks cooperate with each other. A task runs for a while, (long enough to accomplish something but not too long,) and then gives up control of the system to the next task. Each task is expected to play by the rules and not hog the system.

If a task decides when to be swapped with another task many things become simpler. You don't need the safe-guards to prevent an update from being interrupted. Shared routines don't need to be reenterant because no other task will enter the routine until the current task is finished. Because a task knows it will be swapped out, it can save any information that needs to be saved. The swap routine needs to save only minimal information.

Now that you are sold on the advantages of the nonpreemptive multitask scheme, how does it work? There are three routines needed in your multitask program. Two HTPL routines, PROGRAM and NEXT, and one assembly language routine, SWAPTSK. SWAPTSK is added to the runtime library HTPLRTL or linked in as an include file. NEXT is used to transfer control to the next ready task.

To save the state of an HTPL routine (in this context) only the parameter stack pointer and the return stack pointers need to be save. The SWAPTSK routine uses two parameters, a place to save old pointers and a place to find new pointers. SWAPTSK returns to the routine that called it leaving the return address of the routine that called NEXT on the old return stack. Because all of the tasks are written in a single HTPL program, all the other registers are either the same or don't need to be saved because computations are done on the stack.

The main PROGRAM allocates space for the parameter stack and return stack for each task. The starting address for the task is placed at the top of the return stack. A pointer to the first task is set and a task number is set. To start the tasks the program calls SWAPTSK with a pointer to TSK0P where its own pointers will be saved, and a pointer to where it will find the pointers for the first task. When the return statement is executed the first task will start.

Each task in the multitasking system needs to call NEXT at regular intervals. Most programs have a major wait loop where they get a character or wait for time to pass. This is a good place to call NEXT. If there is no wait loop then any place in the main loop of a routine will do. In any multitask system it is important not to waste time that could be used by other tasks.

This is a generic prototype routine that needs to be customized for a specific set of tasks. These routines can be used as a starting point for dedicated multitasking applications. They can help simplify what might otherwise have been a complex or unsovable problem. A much longer discussion of this topic can be found in the next issue of The Computer Journal, or contact us for a sample copy of a multitasking program.

```
( HTPL NONPREEMPTIVE MULTITASK )
( these must be contiguous and in order )
long tsk0p tsk0r tsk1p tsk1r tsk2p tsk2r tsk3p tsk3r ;
long tskpn tskid ;

program
  512 allocate 512 + !tsk1p
  512 allocate 508 + #task1 !4 +4 !tsk1r
  512 allocate 512 + !tsk2p
  512 allocate 508 + #task2 !4 +4 !tsk2r
  512 alloacte 512 + !tsk3p
  512 allocate 508 + #task3 !4 +4 !tsk3r
  #tsk0p !tskpn    0 !tskid
  #tsk0p @tskpn swaptsk ( start system )
  return
  end

proc next
  @tskpn #tsk0 swaptsk    ( save old task )
  @tskid +1 dup !tskid
    if 3 > then 1 !tskid #tsk0p !tskpn end
  @tskpn 8 + !tskpn
  #tsk0 @tskpn swaptsk    ( start new task )
  end

proc task1  end
proc task2  end
proc task3  end


;------- HTPL CODE FOR MULTITASK
;------- SWAP TASK  ( OLD NEW -- )
SWAPTSK MOVE.L  (A4)+,A1        ;NEW POINTER
        MOVE.L  (A4)+,A0        ;OLD POINTER
        MOVE.L  (A7)+,A3        ;RETURN
        MOVE.L  A4,(A0)+        ;SAVE PARAM PNTR
        MOVE.L  A7,(A0)         ;SAVE STACK PNTR
        MOVE.L  (A1)+,A4        ;NEW PARAM PNTR
        MOVE.L  (A1),A7         ;NEW STACK PNTR
        JMP     (A3)            ;RETURN TO CALLER
;---------------
```


HT-Forth

HT-FORTH Features:

* Compiles to JSR, BSR
* Uses K-OS ONE Files
* Unlimited File Size
* Position Independent
* Source Code Included
* String Utilities
* File Utilities
* Screen Utilities
* 32 Bit Stack

.BIN files for royalty free distribution.

Software & Manual: $100.00, Manual: $10.00

* * * * * * * * * * * * * * *

From **Hawthorne Technology**

FLOATING POINT for HTPL

We have just finished up a floating point package for HTPL. It will be available for $25.00.

This package adds floating point math to HTPL programs. It consists of a linkable .HEX file that is 3kb long. It uses the parameter stack to pass all arguments. It does not use any variable space, only the stack is used. There are routines that convert to or from integers or ASCII characters.

A complete manual with examples of using each function is included. One example is an RPN calculator similar to an HP calculator. The calculator code can be added to your K-OS ONE command processor to make it a resident utility.

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 is a stack machine with long stack items it is possible to use the same load and store operators as are used by long integers.

QUESTIONS AND ANSWERS **Q & A**

Here are the questions you have asked:

Q: What happened to the bulletin boards that were listed in the May 68-KNEWS?

A: 1) The Computer Journal system is down due to problems with the phone lines in Montana. Line quality is not quite what is needed for computer communications. It looks like hardcopy will be required to access this resource.
2) Our May issue had the wrong phone number for the Micro Cornucopia bulletin board. The correct number is: (503) 382-7643. 24 hours 300-1200-2400 baud, 8 bits, No Parity, 1 Stop Bit.
3) As far as I know Motorola BBS Freeware is still at: (512) 440-3733

Q: If the TinyGiant is only $395.00, how much extra is it for the operating system?

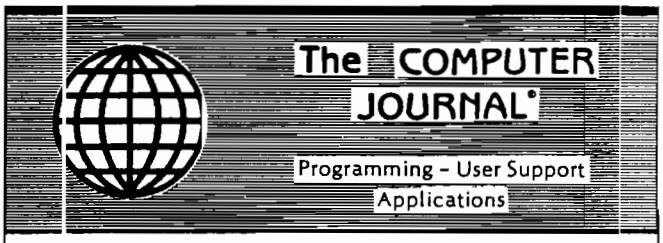
A: $0.00. The K-OS ONE operating system package with command processor and source code, a 68000 assembler, a line editor and the HTPL compiler are all included in the price of the board.

Q: What baud rate does my new TinyGiant board expect the terminal to be set at?

A: 9600 baud.

Q: My TinyGiant powers up and sends a sign-on message to the screen. Then I boot the operating system from disk. I get a prompt on the screen, but then I get no response from the keyboard.

A: The problem is with the terminal. Set it to NO PARITY. The software in PROM takes care of parity for you, but the current version of the operating system that you boot from disk does not. It requires NO PARITY.



68000 TECHNICAL RESOURCE

TCJ is a technical journal for serious microcomputer users, providing a forum for sharing ideas and information. It concentrates on practical articles with useful information on how programs work—how to define problems—and how to design and implement systems to solve those problems, including OEM applications using dedicated controllers.

While we cover several different systems, we are greatly expanding our 68000 coverage with information on K-OS ONE, HTPL, HT-FORTH, assembly language, C and BASIC when they are available, hardware and peripheral interfacing, etc.

TCJ is published for the users, and your articles, tips, and questions are always welcome.

FREE information — Sample issue $3 in the U.S.
6 issues $16 in U.S. — $22 in Canada
MasterCard and Visa Accepted

The Computer Journal
190 Sullivan, Columbia Falls, MT 59912 (406) 257-9119

WHAT IS THE COMPUTER JOURNAL?

The Computer Journal is a bi-monthly magazine that is published in Montana. The information in The Computer Journal is the timeless sort that makes back issues valuable.

Many Computer Journal readers work with industrial electronics. Motorola processors have always been strong in this environment. The 68000 provides a step up from older designs. As the requirements change, more people are switching to the 68000. The Computer Journal is planning to support these people. We will be seeing more articles on using the 68000 in future issues.

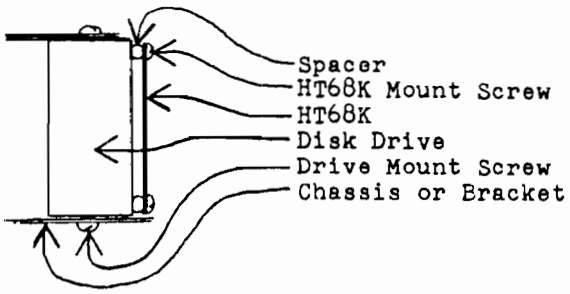
Articles on the 68000 and/or industrial control applications are needed. All good articles are written by people who have experienced the thing they are writing about. Share your knowledge.

If you had problems with a project, write about the problems and how you solved them. If you had no major difficulties, think of the experience you relied on to make that possible.

You can't write? Pretend you are describing your project to someone on the phone. Explain what you did and why so this person can understand without seeing the project. Be complete. Add photo's, drawings or schematics to complement the text. Tell about any special software that was needed. Read it back pretending you have never heard of the project. Edit for clarity. Something that is routine for you may be the solution to someone elses problem.

Send your article to Art Carlson, The Computer Journal, 190 Sullivan Crossroad, Columbia Falls, MT 59912. Contact Art for acceptable disk formats. (406) 257-9119

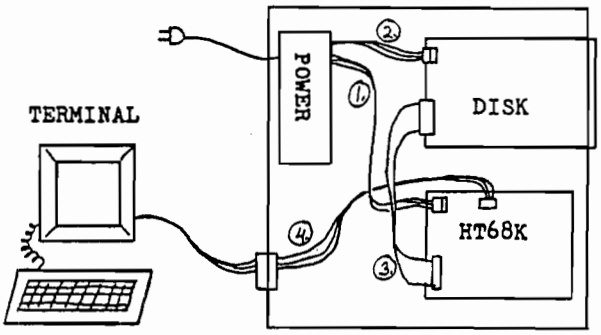
Bringing Up the HT68K TinyGiant.



There are two ways you can mount the HT68K board. The first is to mount it directly to a 5 1/4" disk drive. The notches on the HT68K are spaced properly for this. The edge of the HT68K that has the power and disk control connectors go to the back of the drive making cable lengths minimal. A small spacer should be used between the HT68K and the drive. This is usually necessary to clear all of the components on the disk drive.

The other option is to mount the board directly to the chassis. Spacers may or may not be needed in this case, depending on the chassis, and the type of hardware you use to mount the board.

Once you know how you are going to mount the board, you can figure out how long to make the cables. Cables should always be made as short as possible, but not bind or pull on anything, have a path through the system that is as direct as possible, and avoid draping the cable across any circuit boards. There are always things you have to give in on. A cable that is a little longer is a lot easier to work with but has more chance to pick up noise.



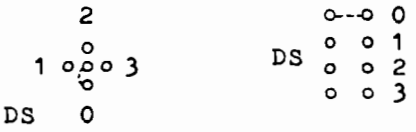
The cables you will need in a HT68K system with a single disk drive are:

1. From the POWER supply to the HT68K POWER connector.
2. From the POWER supply to the DISK DRIVE POWER conn.
3. From the HT68K disk CONTROLLER SIGNAL connector P3 to the disk DRIVE SIGNAL connector.
4. From the RS232 connector P4 to the CONSOLE DEVICE.

Check all your cables and connections. Pin one on the cable should connect to pin one on the board. Make sure there are no loose objects in the way.

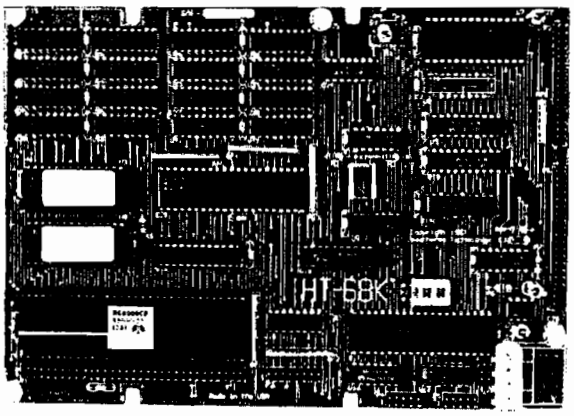
When you turn on the system for the first time, have the disk drive power and signal cables disconnected. This prevents damage to the drive and makes it easier to track down any problem you might have.

Drive Select Option



Have your terminal turned on and set to 9600 baud and No Parity. Power up the board. A sign-on message should appear on the screen immediatly. This is the monitor that is in PROM on the board. It will respond to the commands listed in the monitor portion of the manual.

Now you can power down and connect the disk drive. If you only have one drive, it should be strapped to 0. This is done in different ways on different brands of drives. They are usually labeled DS 0, 1, 2, & 3. Most drives use a strap that is put across two pins next to the number you want selected. Some brands of drives use an IC socket with a plug you put into it with your selections strapped.



With the disk drive connected, turn on the power. Now insert the disk. The drive should be spinning. The sign on message you got earlier will be on the screen and the drive will do some stepping as it loads K-OS from disk. When it is loaded, you will get another message and a prompt. You can now execute your first K-OS ONE command. (Try DIR).

If K-OS ONE will not boot up and the drive continues to spin constantly, check your cable. This is the sympton you get if the drive signal cable is upside down, (all of the signal lines are grounded). Check for pin 1 on the cable mating to pin 1 on the HT68K and pin 1 on the disk drive.