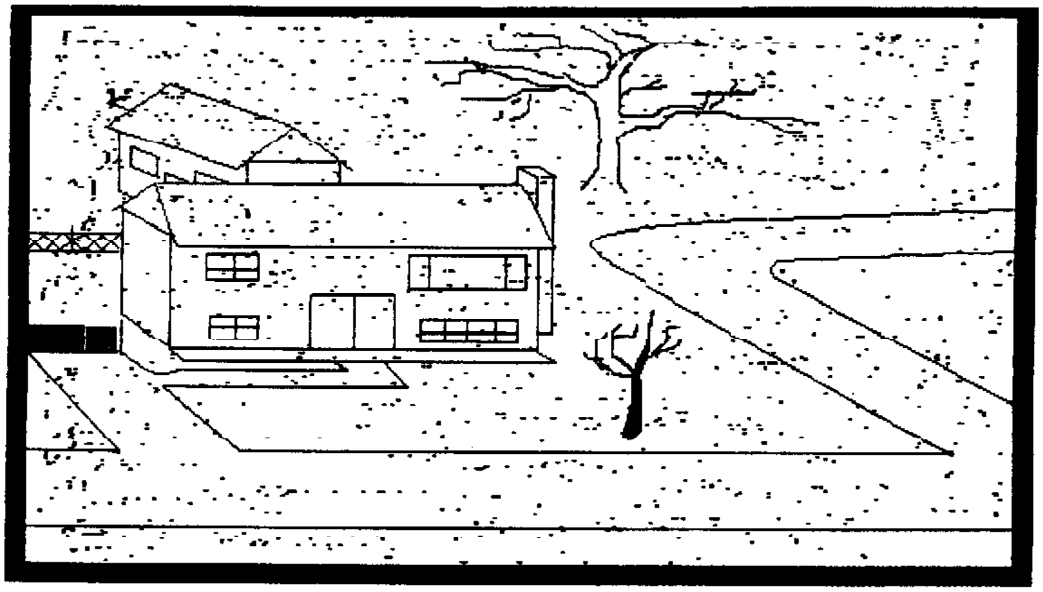
M

The MANNERS Journal of TI Computing

JAN - FEB, 1989



Tlme to stay inside and enjoy Your TI Home Computer!

In This Issue:
- *Part 2 of Al Beard's Fortran Tutorial*
- *Getting the Most From Your Cassette System*
- *C99 articles*
- *and much more!*

In This Issue:

Club Information:

MANNERS Club Officers

President - Jerry Coffey
Membership Chairman - Bill Howard
Newsletter Editor - Dave Ramsey

MEMBERSHIP INFO

Membership in the MANNERS (Mid-Atlantic Ninety Niners) User Group is open to anyone.

Yearly Club dues are $16 per year which includes the bi-monthly newsletter and access to the club library of TI-99 software.

The MANNERS User Group is dedicated to uniting and helping users of Texas Instruments Home Computers and compatibles. These include the TI-99/4A, the TI-99/2, the TI-99/8, the CC-40, and the Myarc 9640. We also welcome users of other 9900-based computer systems to join us in our efforts to use our computers to the fullest.

FOR THE NEXT ISSUE: I am soliciting articles from the members. In particular, we'd like to see more hardware articles. Most helpful of all would be a regular hardware column to teach both the basics and intermediate concepts. I am also looking for a series of articles on beginner's BASIC programming. Anyone who is interested in these areas is encouraged to submit articles BEFORE February 10th to the mailing address indicated on the back cover. Would-be columnists step forward!

The Editor's Page

Dave Ramsey

Welcome to the latest edition of the MANNERS Journal of TI Computing. I'm your new newsletter editor and I hope to get the club newsletter back on track rapidly. But I want to emphasize to everyone that I can't do it alone. We all have to pitch in and lend a hand in club activities. Now, I have the hard part - getting the newsletter ready every two months to be printed, bound, addressed, and mailed. I also have to do the page layouts and the editing of the articles to fit them into the newsletter. What I realy need from our members are articles. I would hope that some of you would volunteer to do a regular column and share your expertise in some area with the rest of us. Other club members may wish to submit single articles on some subject with which you have experience. Please don't be shy about writing - I'll see to it that your article comes through in the best format possible. And remember that while many of us have experience in certain areas, none of us has experience in all of them.

On another note, I want to also ask out members to begin doing two favors for me. The first is to collect clippings and ads of coming computer shows that may be of interest to our membership. Please submit these to me personally at my home address - 1188 Green Holly Drive, Annapolis, MD 21401. The second request is to feel free to write to me, at either the above address or the new club PO Box listed on the back of this issue, and submit classified ads or simply write letters to the editor. If we have the room, I'll try to see that they get printed, especially if they contain corrections or additions to articles that we have run in past issues. These two things can help me quite a bit in improving our newsletter.

One of the reasons that I'm urging everyone to get involved is this: if you do not get involved and submit articles, I'll end up doing most of the writing or looking for articles to reprint. I'm a systems programmer by day and I enjoy programming problems of an advanced sort. If you leave it up to me, you're liable to find this newsletter filled with advanced topics in 'C99' or assembly language. But this would shortchange both our novice members and our hardware enthusiasts. The novices I will try to look out for but I'll still need articles for them, like the series we started in this issue on using a cassette system. As for the hardware hackers, I'd be honored to print whatever you submit but don't expect me to do any of those articles. I'm a walking fire hazard wwhen I have a soldering iron in my hand and have my hands full soldering together a new printer cable!

One of the layout changes that I made was to switch to a double column page format. I did most of this issue on a PC AT at work using Wordstar 5.0. However, parts of it were also done at home using Clint Pulley's ROFF formatter program along with LIST, a program I wrote in C99 to output text in double column format to either the printer or to a disk file. I am hoping to be able to layout the entire next issue on my 9640 here at home. (This editorial was done with ROFF and LIST. Compare it to the Fortran-99 tutorial by Al Beard also in this issue which was done on a PC AT with Wordstar 5.0 using its multi-column mode.)

Finally, as you can all see, I took the liberty of both redesigning our newsletter layout and renaming it. I hope that you all will enjoy it and if you have any other suggestions, including some other name or a proposed layout, please WRITE! I'm looking forward to hearing your ideas about our newsletter.

Understanding 99 FORTRAN, for your TI-99/4A and MYARC Geneve

The following is the second in a series of tutuorials on using the 99 FORTRAN Language. In this section, data types used in 99 FORTRAN are discussed.

Part 2 - Fortran Data Types

A big worry of old time BASIC users is the difficulty in learning a new language like FORTRAN or C. Keep in mind that much of BASIC was based from FORTRAN, as a matter of fact, most modern higher level programming languages (and all procedural type languages) have roots in FORTRAN.

Lets start looking at some FORTRAN statements. The first statement to keep in mind is the ASSIGNMENT statement, which lets you assign a value to a variable, just like the BASIC LET statement:

BASIC	FORTRAN
LET A=5	A=5.0
LET A2=8	A2=8.0
LET ALPHA=10	ALPHA=10.0

So, FORTRAN looks a lot like basic, except the LET keyword is not used (and note that it is not needed in TI-BASIC either, most BASIC's dropped the requirement for the LET statement, to be more "FORTRAN" like).

One item you didn't have to worry about in BASIC (at least TI-BASIC) are data types. TI-BASIC has only TWO data types, string and double precision. 99 FORTRAN has FOUR data types, as follows:

TYPE	SIZE (IN BYTES)
INTEGER	2
SINGLE PRECISION	4
DOUBLE PRECISION	8
LOGICAL	2

Why data types? Your TI-99 microprocesser actually does it computing in integer format (sometimes called INTEGER *2 to indicate the number of bytes needed to store a variable). Writing such things as loops and doing common functions as indexing are MUCH faster in INTEGER than in the DOUBLE PRECISION format used by BASIC. As a matter of fact, benchmarks have shown that a loop and index program in 99 FORTRAN using INTEGERS is over 100 times faster than using TI-BASIC. So, we humans put up with the data types for the performance increase gained.

FORTRAN originally only had two data types, INTEGER and SINGLE PRECISION, so a convention was invented which still exists today for INTEGER and SINGLE PRECISION variables, INTEGER variables start with the letters I, J, K, L, M, or N; and SINGLE PRECISION variables start with the rest. Lets look at the following variables, and their default types:

ICON	type integer
INDEX	type integer
XRAY	type single precision

So where are the other data types DOUBLE PRECISION and LOGICAL? There are no default letters for these types, but there is a way to declare a variable as the other types. More on that later.

Lets write a simple program that computes a Fahrenheit to Celsius table, from 0 to 300 degrees, every ten degrees, using the INTEGER and SINGLE PRECISION variable types as used above.

Load up your 99 FORTRAN, and call in the editor. Type the program from listing #2:

Save the program to a disk, and compile, link, and run the program using the procedures outlined in the previous tutorial (and also using the 99 FORTRAN manual) (NOTE: while compiling, you will get a warning displayed on the screen:

```
CELSIUS = (5.0/9.0) * (IFAHR-32)
** Warning Mixed Mode Arithmetic
```

Ignore the warning. It is just telling you that you have used two data types in a single statement, integer for IFAHR and single precision for the constants 5.0 and 9.0).

Get a listing of the program. Note at the end of the listing, there is an allocation summary. Look at the Local Data Area summary, and you will see two variables:

```
0008 i IFAHR       000A r CELSIUS
```

These are the starting locations in the data area for the variable IFAHR and CELSIUS, and the variable types "i" for INTEGER (variable IFAHR) and "r" for REAL (variable CELSIUS).

Writing Constants:

The way you write a constant in a FORTRAN statement determines its data type. In BASIC, it didn't matter, you could write 1, 1.0, 1.E0, and the data type would all be the same. In 99 FORTRAN, the basic rules on numeric constants are:

o If the constant does not contain a decimal point (e.g. 1, 2, -300, etc.) then the constant type is INTEGER, and each constant takes two bytes of storage.

o If the constant contains a decimal point (e.g. 1.0, 2.0, 3.1415927, -300.123), then the constant is of type SINGLE PRECISION, and takes four bytes of storage.

o If the constant contains an exponent starting with the letter "E" (just like TI-BASIC, 1.0E10, .31415927E01, etc.), then the constant type is SINGLE PRECISION, and takes four bytes of storage.

o If the constant contains an exponent starting with the letter "D", (e.g. 1.0D10, .31415927D01, etc.) then the constant type is DOUBLE PRECISION, and takes eight bytes of storage (just like a TI-BASIC numeric variable).

In the example program, we wrote the line:

```
CELSIUS    = (5.0/9.0) * (IFAHR-32)
```

with two single precision constants and an integer constant:

```
5.0  -  SINGLE PRECISION REAL
9.0  -  SINGLE PRECISION REAL
32   -  INTEGER
```

Now, lets write the same line with two double precision constants and an integer constant:

```
CELSIUS = (5.0D0/9.0D0) * (IFAHR-32)
```

Note that 5.0D0 and 9.0D0 are both double precision constants.

Why do we want double precision? Isn't it just confusing?

Single precision offers a savings of two to one over double precision in terms of memory requirements. An array of 500 single precision elements would take only 2000 bytes (or characters) of memory storage, whereas an array of 500 double precision elements would take 4000 bytes (or characters) of memory storage.

However, single precision is limited to five or six digits of precision. This is enough for a simple program like this temperature conversion program, but certainly not enough for some big "number cruncher" type programs which need to squeeze every bit of precision possible into the final result.

Declaring other Data Types

Remember I said that variables which start with the letters I, J, K, L, M, N are considered to be integer, whereas all others are to be considered of type SINGLE PRECISION REAL. How can the user use the other data types, namely DOUBLE PRECISION and LOGICAL? Quite easily, using the type declaration statements.

The four declaration type statements are:

INTEGER	declares 2 byte integers
REAL	declares 4 byte single precision
DOUBLE PRECISION	declares 8 byte double precision
LOGICAL	declares 2 byte logical

Declaration statements can be explicit or implicit, i.e. you can give the program a list of variables to be explicitly typed, or give the program a range of letters to be declared implicitly.

Lets look at the same program coded with explicit data types in Listing 3:

Note that the variable IFAHR (implicitly INTEGER) has been recoded to use an explicitly declared variable FAHR. The variable CELSIUS is now of type double precision (just to show off the usage of the data type).

IMPLICIT type statements look similar to the explicit types, except they are preceded by the word IMPLICIT, and instead of variable names contain letters or a range of letters to be declared implicitly. For example, to define the FORTRAN standard of integers starting with the letters I,J,K,L,M,N, you could write:

```
IMPLICIT INTEGER (I-N)
```

and the single precision real's as:

```
IMPLICIT REAL(A-H,O-Z)
```

Usually, when I code a program, I use integers more often then other variable types, so I code the first statement of a routine as:

```
IMPLICIT INTEGER(A-Z)
```

Some modern books of programming (and many new languages such as C, PASCAL, and PL/I) demand that every variable be explicitly declared. A famous story in the early 1960's was of a NASA Venus probe that missed Venus due to a mis-typed variable in a FORTRAN program.

Lets re-write that example program one more time using IMPLICIT type declarations: (see listing #4)

Logical Data Types:

Logical data types are generally not represented in implementations of BASIC, which in my opinion, is a great tragedy. Logical constructs are VERY powerful, and VERY simple. A logical constant only has two forms:

.TRUE. and .FALSE.

For example, we can compare two numbers, and define the result of that comparison in a logical variable, as follows:

```
LOGICAL FLAG
REAL  A,B
FLAG  =  A .GT. B
```

There are six relational operators, as follows:

.GT.	Greater Than (>)
.GE.	Greater Than or Equal (>=)

.LT.	Less Than
.LE.	Less Than or Equal To (<=)
.EQ.	Equal To
.NE.	Not Equal To

So, for the following expressions:

```
3.LT.10    is .TRUE.
20.GT.30   is .FALSE.
```

There are several other operators which can be combined to build more powerful logical expressions, as follows:

.AND. - True if left and right operands are true
.OR. - True if either left and right operators are true
.EOR. - True if either left and right operators are true, but NOT both
.NOT. - True if operand following is not true

Let's write that program one more time, using a logical variable to control the operation of the loop: (see listing #5)

Many programmers find this program structure (with the structured DO WHILE loop and logical constructs) to be easier to understand that the somewhat more cryptic DO statement.

Where is??????

Note here we have left out a discussion of character data (BASIC string data) and other data types used in more powerful FORTRAN compilers, such as COMPLEX.

Some of these data types will be added in an upcoming release of 99 FORTRAN. Character data can be stored in the current version of 99 FORTRAN as Hollerith variables, but that is another tutorial.

In Summary

This tutorial introduces the FORTRAN data types INTEGER, LOGICAL, REAL, and DOUBLE PRECISION. These variables can be declared using the implicit typing of FORTRAN (integers start with letters I,J,K,L,M and N; and all others are single precision); they can be declared explicitly using the explicit INTEGER, LOGICAL, REAL, or DOUBLE PRECISION statements; or they can be declared implicitly using the IMPLICIT INTEGER, IMPLICIT LOGICAL, IMPLICIT REAL, or IMPLICIT DOUBLE precision statements.

Variable typing gives you flexibility in your program to decide which data types to use where, for the ultimate in performance from your TI-99 computer.

The next tutorial will discuss I/O handling and formatting, one of the most powerful features of 99 Fortran.

Enjoy!

FORTRAN 99 Tutorial Program Listings

(Ruler lines are supplied to help readers in learning about proper alignment of FORTRAN source code. Labels exist in columns 1-5, Fortran statements are in columns 7-72, comments are denoted by a C in column 1, and continuation marks are placed in column 6.)

Listing #2

```
+---+----1----+----2----+----3----+----4----+----5----+----6----+----7-+
      DO 100 IFAHR=0,300,10
         CELSIUS    = (5.0/9.0) * (IFAHR-32)
         WRITE ( 6, 9100 ) IFAHR, CELSIUS
100   CONTINUE
      STOP
9100  FORMAT ( 1X, I6, ' DEGF IS ', F6.2, ' DEGC')
      END
```

****

LISTING #3

```
+---+----1----+----2----+----3----+----4----+----5----+----6----+----7-+

      INTEGER FAHR
      DOUBLE PRECISION CELSIUS
      DO 100 I=0,300,10
         CELSIUS    = (5.0D0/9.0D0) * (FAHR-32)
         WRITE ( 6, 9100 ) FAHR, CELSIUS
100   CONTINUE
      STOP
9100  FORMAT ( 1X, I6, ' DEGF IS ', F6.2, ' DEGC')
      END
```

****

LISTING #4

```
+---+----1----+----2----+----3----+----4----+----5----+----6----+----7-+

      IMPLICIT INTEGER(I,F)
      IMPLICIT DOUBLE PRECISION(C)
      DO 100 I=0,300,10
         CELSIUS    = (5.0D0/9.0D0) * (FAHR-32)
         WRITE ( 6, 9100 ) FAHR, CELSIUS
100   CONTINUE
      STOP
9100  FORMAT ( 1X, I6, ' DEGF IS ', F6.2, ' DEGC')
      END
```

****

LISTING #5

```
+----+----1----+----2----+----3----+----4----+----5----+----6----+----7-+

      INTEGER FAHR
      DOUBLE PRECISION CELSIUS
      LOGICAL DONE
      DONE = .FALSE.
      DO WHILE ( .NOT. DONE )
         CELSIUS    = (5.0D0/9.0D0) * (FAHR-32)
         WRITE ( 6, 9100 ) FAHR, CELSIUS
         FAHR = FAHR + 10
         DONE = FAHR .GT. 300
      ENDDO
      STOP
9100  FORMAT ( 1X, I6, ' DEGF IS ', F6.2, ' DEGC')
      END
```

Below are listed some vendors who still supply products to the TI Home Computer community. We will be listing others in future issues but if you have a particular vendor you'd like mentioned, let us know!

SOFTWARE:

Asgard Software
P.O. Box 10306
Rockville, MD 20850

Databiotics
P.O. Box 1194
Palos Verdes, CA 90274

E&M Software
Box 551
Oscoda, MI 48750

Genial Computerware
P.O. Box 183
Grafton, MA 01519

Great Lakes Software
804 E. Grand River Ave.
Howell, MI 48843

Harrison Software
5705 40th Pl.
Hyatsville, MD 20781

Nameloc Software
3971 S.E. Lincoln
Portland, OR 97124

Texaments
244 Mill Road
Yaphank, NY 11980

Trio+ Software
P.O. Box 115
Liscomb, Iowa 50148

HARDWARE:

Bud Mills Services
166 Dartmouth Drive
Toledo, OH 43614

Corcomp, Inc.
2211-G Winston Road
Anaheim, CA 92806

Dijit Systems
4345 Hortensia Str.
San Diego, CA 92103

Myarc, Inc.
241 Madisonville Road
Basking Ridge, NJ 07920

Rave 99
112 Rambling Road
Vernon, CT 06066

News, Rumors, and Scuttlebutt Of Goings-on in the TI Community

Dave Ramsey

Recently, Al Beard has given me the great pleasure of beta-testing his Fortran-99 compiler for MDOS. While it has been a long time in the making, Fortran-99 will be a welcome addition to the 9640 programmer's toolkit. I have had the opportunity to port some interesting applications to the 9640 running in its native MDOS mode. These have included an AVL-tree package, a circular-linked list management package, and I am currently working on porting a cross-reference utility.

So far, Fortran-99 seems to have handled all of these Fortran subroutines and functions without so much as a whimper. And remember, these packages were developed on an IBM 4341 mainframe running VM/CMS and using the IBM FORTVS Fortran compiler. This says alot for the quality of Al's work to date. He has planned many other features, the most prominent of which will hopefully be a symbolic debugger! But none of this is finalized yet and Al continues to work on the package. If you think you might be interested in obtaining Fortran-99 for your Geneve, write Al a letter at LGMA Products, Box 210, RD 4, Coopersburg, PA 18036. As soon as it is ready, I'm sure Al will let you know.

If you don't have a Geneve, you can already obtain Fortran-99 from a number of sources right here in the DC area including Asgard Software, run by Chris Bobbitt, and Quality-99 Software, run by Larry Hughes. Both of these fine TI software houses would be happy to take your order for such a fine product. And since Fortran-99 compiles to machine code, you could even consider developing commercial programs for the 4A with it. In any case, I do strongly recommend Fortran-99 to anyone who regularly uses TI Extended BASIC but who wants to get more speed out of their programs without resorting to assembly language.

In other areas, change continues to encompass the TI world. Myarc has released its combined hard disk/floppy disk controller and is getting consistent praise for the new peripheral card. This card replaces the floppy disk controller in the expansion box. It then provides the capability to hook up 3 floppy drives in 40 or 80 track mode and double or single density. It also allows the user to hook up any standard ST-506 hard disk. Up to four of these hard rives can be attached. The drives can be of any capacity up to 40 megabytes of storage. (That's 40,000,000 bytes of storage!) As a final touch, Myarc has added a streaming tape backup capability to the card. Unfortunately, with the initial release, the streaming tape backup is not yet operational. There is no software to support it at this time. But give TIers a chance and we'll be seeing streaming tape backup utilities for this card very shortly.

Back on the 9640 front, Myarc has released MDOS version 1.14 and it looks like MDOS is beginning to really stabilize in all areas except the video XOPS. These are still in flux as Chris Faherty and Lou Phillips continue work on Advanced BASIC.

Additionally, there is a rumor of a new Z80 coprocessor board coming out of Canada, possibly in mid-1989. Details on this co-processor board are not yet available but we'll keep you informed as events progress.

Many other things have been occurring in the TI world also. Our own Tom Wible has released a new version of his C99 optimizer. If you program in C99 and don't use Tom's optimizer, you aren't getting optimum machine code! His optimizer will squeeze out the last bit of space and speed from your C99 programs. If you are interested in it, talk to Tom and see if it has been placed in the UG library yet.

Charles Earl has released version 2.3 of TELCO which corrects additional bugs and provides for both hard disk support on the 4A using the new Myarc controller and for 9640 support. TELCO is becoming the premier TI-99/4A telecommunications program in use today. With windows, program overlays, multiple terminal emulations, and several file transfer protocols, there is not another TI communications program that can compare to it.

Asgard continues work on Press, their new word processor, also written by Charles Earl. Press will bring some of the power of IBM-PC type word processors to the 4A world. Press has not yet been released but it is now approaching completion. Watch for it if you want more word processing power than TI-Writer currently gives you.

Additional rumors indicate that prices of hard disk drives may soon follow the prices of RAM chips. Many hard drive makers are considering shutting down some of their capacity due to the current abundance of hard drives on the market. Such a move would drive prices up in the long run. If you are considering a hard drive to go with the new Myarc HFDC Card, now may be the time to purchase.

One of the more tenacious of TI software and hardware distributors, Triton, has been sold to Ashton-Tate. This could be either a sign of better things to come with continued TI support or it could signal the end of Triton's support for the 4A market. If you have any purchases that you planned to make through Triton in the near future, do it now. This will show the TI community's continuing interest in new products. And besides, if they do close their doors to TI support, you'll at least have bought those items you wanted!

That about wraps up my current report on the goings on of the TI world. If you know of anything new or exciting for TI-99/4A owners, drop us a note at our new post office box and we'll see to it that your message gets out to the rest of the MANNERS members! Take care and happy computing!

LINK TIME CONSTANTS IN c99

Tom Wible

Link time constants are not unique to c99; in fact they are a feature of the underlying assembly code and the TI's linking loader. The trick is to DEFine a symbol whose value is the desired constant in one object file, preferably a small, quick-to-assemble one. Then by REFering to the constants and using them just like you would any constant, your main c99 modules don't have to be recompiled and reassembled every time you want to change the size of, for example, an array.

The enclosed files, ;DATA and MAIN, are taken from my disassembler post-processor that I uploaded to BBBB last month. While developing this program, I often had to change the size of the label arrays, and I got tired of long recompiles.

In the ;DATA file, note that I have made these defines conditional: in the MAIN file, I have #defined the (compiler) symbol PSEG (for program segment), so that the c99 compiler will see the extern variable definitions. When the ;DATA file is then #included in the MAIN file, the compiler will understand and use the variables, but will not allocate space for them. When the ;DATA file is compiled, space will then be allocated. This also means that the variables don't have to be stored with the program, or included in the saved program file: just load C99PFEND after your last program object file and before your ;DATA object. As long as the addresses are defined (by the loader), your program can access them. Unless you are storing predefined data (initialized arrays, tables, etc.) you don't have to save the space in the program image file, thus making it smaller.

The only drawback is 2 dimensional arrays: the second dimension must be known to the compiler. This is illustrated in the character array pname: the second dimension is #defined first(NAMLEN).

C99 Listing for "Link Time Constants in C99"

```
/* ;DATA */
#define NAMLEN   7

#ifndef PSEG

#asm
 DEF PMAX,LMAX,XMAX  * link time constants

PMAX         EQU  80 * these constants are
LMAX         EQU 120 * passed to PSEG
XMAX         EQU  40 * at link
#endasm

#define PMAX      80
#define LMAX     120
#define XMAX      40

entry pname,paddr,addr,xaddr,ix,pu;
char pname[PMAX][NAMLEN];
int  *paddr[PMAX], *addr[LMAX], *xaddr[XMAX];
int  ix[LMAX], pu[PMAX];

#else

#asm
 REF PMAX,LMAX,XMAX     * link time constants
#endasm

extern char pname[][NAMLEN];
extern int  *paddr[];
extern int  *addr[];
extern int  *xaddr[];
extern int  ix[];
extern int  pu[];

#endif

/* MAIN */
#define PSEG yes
#include dsk.c_source.;data

extern printf();
int    i;

main()
{
  for(i=0,i<PMAX,++i) {
     printf ("\nenter a %d char string:", NAMLEN-1);
     scanf ("%*s", NAMLEN-1, &pname[i][0]);
     printf ("this is the %dth string:%s", i, &pname[i][0]);
  }
}
```

First Impressions of the Myarc Floppy Disk Controller

by Dave Ramsey

Wow! Now I know that alot of you have been using Corcomp and Myarc floppy disk controllers for a long time. And I know that alot of you wrote about how wonderful they are. But that was a few years ago and alot of new folks have come into the group since then. Also, alot of old-timers like myself just never got around to upgrading.

But during December I finally did it - I bought a Myarc floppy disk controller card. Now don't confuse this with the new combined Hard disk/Floppy Disk controller card. No, this is the older floppy-only controller card. But am I ever impressed!

This thing makes it a pleasure to use a floppy disk system again. (I've gotten spoiled by both the mainframe at work and the AT's we have on the office network.)

Well, before I gush all over everyone with uncontrolled praise, how about if I simply talk about this card. First off, the Myarc card is a direct replacement for the TI floppy controller card. It has the same type of connectors on board - an IDC 34 pin male to support the P-Box drive and a 34 pin edge card connector on the back to support the external drives. But in addition to that, it comes with a set of user settable switches on the card itself.

I've been told that the settings on these switches can mean various things depending on which version of the floppy controller EPROM is installed in the card. Specifically, the switches can either control drive step rates or can signal that the drives are 40 or 80 track. In my case, since I am using a Geneve 9640 with MDOS, coupled with a Myarc 80-track EPROM, the switches tell the DSR routines that I have 1 40-track and 1 80-track drive attached to the system.

Normally, the card is shipped with a 40-track only support EPROM and in that case the switches simply control drive step rates. But an 80-track EPROM is available at additional charge from Myarc if you want to run quad density drives on your TI.

An additional feature of the Myarc card is the excellent Disk Manager V software provided with it. It allows the user to do far more with his disk system than the TI Disk Manager cartridge ever did. This includes setting format interlace and various other functions.

But the two biggest thrills are just in watching this disk controller in action! The Myarc card reads and writes to a floppy faster than the TI controller. And when you format a disk with the system, it's quite a pleasant surprise to see 1438 sectors free on a 40-track floppy or 2878 sectors free on an 80-track floppy.

By just reorganizing my personal floppy backups and library disks, I should be able to store the same volume of programs and files on only 1/4 as many disks. And I have the added option with the Myarc card of adding not only a third drive but also a fourth floppy drive. (The CorComp Disk Controller Card also provides support for 4 floppy drives instead of the TI standard of 3.)

In any case, if you are thinking about possible upgrades to your TI system and are wondering where to go next, consider a Myarc floppy controller card. It seems to be quite reliable and enhances a floppy disk system from the "just usable" stage to the "fun-to-use" stage. I rate this card with a solid "A" grade and recommend it highly.

The Myarc controller card is currently available from a number of sources for prices ranging from $169.99 to $189.99. Be sure to shop before you buy and call all the major TI hardware suppliers.

****

First Impressions of the Myarc Floppy Disk Controller

by Dave Ramsey

Wow! Now I know that alot of you have been using Corcomp and Myarc floppy disk controllers for a long time. And I know that alot of you wrote about how wonderful they are. But that was a few years ago and alot of new folks have come into the group since then. Also, alot of old-timers like myself just never got around to upgrading.

But during December I finally did it - I bought a Myarc floppy disk controller card. Now don't confuse this with the new combined Hard disk/Floppy Disk controller card. No, this is the older floppy-only controller card. But am I ever impressed!

This thing makes it a pleasure to use a floppy disk system again. (I've gotten spoiled by both the mainframe at work and the AT's we have on the office network.)

Well, before I gush all over everyone with uncontrolled praise, how about if I simply talk about this card. First off, the Myarc card is a direct replacement for the TI floppy controller card. It has the same type of connectors on board - an IDC 34 pin male to support the P-Box drive and a 34 pin edge card connector on the back to support the external drives. But in addition to that, it comes with a set of user settable switches on the card itself.

I've been told that the settings on these switches can mean various things depending on which version of the floppy controller EPROM is installed in the card. Specifically, the switches can either control drive step rates or can signal that the drives are 40 or 80 track. In my case, since I am using a Geneve 9640 with MDOS, coupled with a Myarc 80-track EPROM, the switches tell the DSR routines that I have 1 40-track and 1 80-track drive attached to the system.

Normally, the card is shipped with a 40-track only support EPROM and in that case the switches simply control drive step rates. But an 80-track EPROM is available at additional charge from Myarc if you want to run quad density drives on your TI.

An additional feature of the Myarc card is the excellent Disk Manager V software provided with it. It allows the user to do far more with his disk system than the TI Disk Manager cartridge ever did. This includes setting format interlace and various other functions.

But the two biggest thrills are just in watching this disk controller in action! The Myarc card reads and writes to a floppy faster than the TI controller. And when you format a disk with the system, it's quite a pleasant surprise to see 1438 sectors free on a 40-track floppy or 2878 sectors free on an 80-track floppy.

By just reorganizing my personal floppy backups and library disks, I should be able to store the same volume of programs and files on only 1/4 as many disks. And I have the added option with the Myarc card of adding not only a third drive but also a fourth floppy drive. (The CorComp Disk Controller Card also provides support for 4 floppy drives instead of the TI standard of 3.)

In any case, if you are thinking about possible upgrades to your TI system and are wondering where to go next, consider a Myarc floppy controller card. It seems to be quite reliable and enhances a floppy disk system from the "just usable" stage to the "fun-to-use" stage. I rate this card with a solid "A" grade and recommend it highly.

The Myarc controller card is currently available from a number of sources for prices ranging from $169.99 to $189.99. Be sure to shop before you buy and call all the major TI hardware suppliers.

****

The MANNERS Percom Project

Jerry Coffey

MANNERS has been working on a cheap 99/4A expansion project using surplus (as-is) Percom components. Even writing off bad pieces, the cost for a SSSD stand-alone disk system is coming in around $40 less case. Add one of the 32k mods and you have a system that will run most 4A software. This means you can put together a LOGO engine for the kids for less than $100. We have about 10 systems in various stages of completion. I have been sending the bad boards to John Willforth and Richard Roseen who have gotten several back in service already.

The following notes give some of the details of getting one of these systems running.

PERCOM NOTES

The Percom controller board has its own rectifiers and voltage regulators. It requires only a center-tapped AC transformer for power.

If only the board is to be powered (separate power for the disk drive), then a transformer as small as 12.6 volts will suffice. If the drive is also to be powered from the board, use a transformer of 18 volts (e.g. the Radio Shack 2 amp 273-1515B). In either case run the center wire to the middle pin of the 3-pin connector near the 12 volt regulator (7812), then connect the outer wires to the outer pins. There is a standard connector for this pin spacing -- if you want to make the board removable, use the connector on the transformer outputs and plug into the board.

Both regulators stand up on the board so they can be bolted to the cabinet or other suitable heat sink. The 7805 (5 volt) regulator does not have a lot of reserve so unless you are using a low-power drive, you may need to parallel the input and ground pin of the 7805 regulator to a second 7805 (also heat sinked) and move the 5 volt wire of the drive power plug to the output of this second regulator.

The board is connected to the console I/O port by a 44 line cable. A suitable cable can be made from a surplus 50 line cable for 8" floppy drives and a Commodore 44 pin card edge connector (use a right angle or long tailed wire wrap version with .100 spacing NOT the .156 spacing). Use the 50 pin card edge at the Percom board (jam a short piece of pc board into the connector to block off the last three pairs of contacts). Then bend the pins of the 44 pin connector so they line up with sockets 1-44 of the connector on the other end of the surplus drive cable. You will need to shield this cable with to avoid TV interference -- copper mesh or foil will work. Since shielding and tape wrap will make the cable stiff, lay it out and make any permanent bends before you wrap it.

At this point the board will be ready to run a drive connected with either type of TI 34 pin connector. Be sure pin #1 is in the location marked on the board. The termination pack at the rear edge of the board should be retained for single drive applications (the drive may ALSO require a termination pack or a 200 ohm resistor for the drive select line. Remove the pack from the board if second and/or third drives are to be used.

ALTERNATIVE POWER SUPPLIES: The board can be successfully powered from a console power supply installed near the board. This requires removing the 7805 and 7812 from the board and feeding the power into the 4 pin disk drive connector. Other approaches have been used but may require a jumper to complete the circuit.

SECTOR LAYOUT OF TI DISK

Byte numbers in HEX(dec)

SECTOR 0

0--------9	A-B(10-11)	C(12)	D-E(13-15)	10(16)	11(17)
disk_name 10 chars fill out with spaces	total_sectors 360=0168h 720=02D0h 1440=05A0h 2880=0B40h	sectors/track 9=09h 16=10h 18=12h	"DSK" : formatted flag	prot flag P=50h U=20h	tracks/side 35=23h 40=28h 80=50h

12-13(18=19)	14-37(20-55)	38-FF(56-255)
sides/density 01 01 02 02	reserved	sector allocation bit map - each byte represents 8 consecutive sectors; a binary "1" for each used sector, zero otherwise - low order bits represent low numbered sectors, i.e. CFh=11001111 in byte 38h means sectors 0-3 (rightmost 1's) used, 4&5 unused, and 6&7 used. *for the Myarc quad system each bit represents two sectors*

SECTOR 1

Up to 127 pointers (each one word = two bytes) giving the number of the sector (in hex) of a "File ID Block". Pointers are arranged in the alphabetic order of the corresponding file names. The end of the directory is marked with the word 0000 (a fresh formatted disk contains all 00's).

EXAMPLE: If sector 2 is the FIB for "MYFILE", sector 3 is the FIB for "PROGRAM1", and sector 4 is the FIB for "LOAD", then this sector would show the sequence: 0004 0002 0003 0000 ..... all zeros (or pointers to files overwritten on the disk). NOTE: File ID Blocks are also called File Descriptor Records=FDR

SECTORS 2-21(2-33)=FIB's < 2-19(2-31) for early Myarc controller >

0--------9	A-B(10-11)	C(12)	D(13)	E-F(14-15)
file name	not used =0000	file type byte -see below-	records/sector at maximum record size	no. of sectors allocated to file

10(16)	11(17)	12-13(18-19)	14-1B(20-27)	1C-FF(28-255)
EOF offset in last sector unless FIXED	maximum record size	record count (byte reversed)	reserved for date stamp (used on 9640)	block links, 3 bytes each -see below-

FIB NOTES:

1.____MEANING OF BITS IN FILE-TYPE BYTE

```
                   bit number
          8   7  6  5   4    3  2    1
         ------------------------------
 bit    1 :VAR  -  -  -  PROT - INT PROG :
value     :                              :
        0 :FIX  -  -  -  UNP  - DIS  -   :
         ------------------------------
```

THUS:

00001001=09h is protected program

10000010=82h is internal/variable

2.____Block Link is three bytes as follows <f2 f3><n3 f1><n1 n2> where the hexadecimal number f1 f2 f3 is the first sector of the block and the hexadecimal number n1 n2 n3 is the number of sectors in the block, there is one block for each "fracture". Thus an unfractured file of 0CA(202) sectors beginning at 22(34) would have a single block link specifically: 22 A0 0C.

3.____After the disk has 32 files, the next FIB is written in the next available sector (usually the sector preceding the body of the file).

****

PROGRAM LISTINGS FROM
"Getting the Most From Your Casette System"

Listing 1.

```
100 REM *****************************************************************
110 REM *                                                               *
120 REM *     program listing for a cassette tape catalog in t.i. basic *
130 REM *                      created by                               *
140 REM *                    mickey schmitt                             *
150 REM *                                                               *
160 REM *****************************************************************
170 CALL CLEAR
180 PRINT "cassette title:_____________________________________":: 
190 PRINT "cassette number:___________":: 
200 PRINT "cassette side:_____________"::: 
210 PRINT "counter reading:___________":: 
220 PRINT "language used:_____________":: 
230 PRINT "peripherals needed:________________________________":: 
240 PRINT "program name:______________________________________":: 
250 PRINT "program description:_______________________________"::: 
260 GOSUB 10000
270 CALL CLEAR
280 REM to catalog more than one program - follow the same format as used in
290 REM line numbers 210 - 270.  continue using this same format till all of
300 REM your programs have been cataloged.
310 REM caution:  after your final entry - remember to use an "end" statement
320 REM right after your final "CALL CLEAR" statement.
330 REM following this format will help keep all of your programing
340 REM information uniform and easier to follow on your monitor or tv screen.
350 END
10000 PRINT "Press: any key to continue"
10010 CALL KEY(0,K,S)
10020 IF S=0 THEN 10010
10030 RETURN
```

Getting The Most From Your Cassette System

by Mickey Schmitt

[Ed. Note: This is a multipart series that has been condensed into a series of 3 articles for republication in the MANNERS Journal of TI Computing.]

I. Getting Started

Before you try to do anything with a cassette system you need to start with the right equipment. There are many different models of standard cassette recorders available which will work with your TI computer. (Besides the official TI program recorder). However, for best operation and alot less mental aggravation, you should use a cassette recorder with the following features:

1. volume control

For best results this should be set between mid-range and maximum settings.

2. tone control

For best results this should also be set between mid-range and maximum settings.

3. microphone jack

This jack is needed in order to receive information from your computer.

4. earphone or external speaker jack

This jack is needed in order to send information to your computer.

5. remote jack

This jack makes it possible for your computer to control your cassette recorder's drive motor - thus your tape recorder will run by pressing the "enter" key on your computer console.

6. digital tape counter

This is a very important feature as it will save you alot of unnecessary aggravation. This feature enables you to easily locate the correct tape position of your program or data file. This is especially useful when you want to store more than one program on the same side of the cassette tape.

Next, you will need to have the TI cassette interface cable which is used to connect your recorder to your computer. Although this cable comes with the official TI program recorder, it must be purchased separately if you are using another type of cassette recorder. If you are having trouble finding this cable, I would suggest trying the Computer Bug (412-882-3374) 5075 Clairton Boulevard, Pittsburgh, Pennsylvania 15236.

The following instructions will guide you through the process of connecting your cassette recorder to your computer using the TI cassette interface cable:

1. Locate the nine-pin plug at one end of the cassette recorder interface cable. Insert this plug firmly into the jack on the right rear of the computer.

2. Locate the set of three plugs at the other end of the cable. The wires that lead to these plugs are color-coded: red - white - black.

3. Locate the jacks labeled: mic - ear (or external speaker) and rem on your cassette recorder.

4. Insert the plug with the red wire into the recorder's microphone jack (labeled mic).

5. Insert the plug with the white wire into the recorder's earphone (or

external speaker) jack (labeled ear).

6. Insert the plug with the black wire into the recorder's remote jack (labeled rem).

That's all there is to it! Your cassette system is now ready to go.

II. Loading and Saving Programs:

While loading and saving programs with the use of a cassette recorder is not a difficult process in itself - reading and understanding the instructions for the very first time can be quite confusing. With that thought in mind I have tried to keep the instructions as simple as possible.

Instructions for loading programs:

```
1.  Type:  old cs1
2.  Then:  press enter
3.  Follow the directions as they
appear on your monitor or tv screen:
  3.1  *  rewind cassette tape   cs1
          then press enter
  3.2  *  press cassette play    cs1
          then press enter
  3.3     computer displays message:
       *  reading
  3.4     computer displays message:
       *  data ok 3.5
       *  press cassette stop    cs1
          then press enter
4.  Wait for the flashing cursor to
appear in the lower left-hand corner
of your monitor or tv screen
5.  Type:  run
6.  Then:  press enter
```

Instructions for saving programs:

```
1.  Type:  save cs1
2.  Then:  press enter
3.  Follow the directions as they
appear on your monitor or tv screen:
  3.1  *  rewind cassette tape   cs1
          then press enter
  3.2  *  press cassette record  cs1
          then press enter
  3.3     computer displays message:
       *  recording
  3.4  *  press cassette stop    cs1
          then press enter
```

4. Your program is now saved - but you should get into the habit of checking all your programs to be sure that they were saved without error.

```
5. Continue to follow the directions
as they appear on your monitor or tv
screen:
  5.1     computer displays message:
       *  check tape (y or n)?
  5.2     type:  y
  5.3     then:  press enter
  5.4  *  rewind cassette tape   cs1
          then press enter
  5.5  *  press cassette play    cs1
          then press enter
  5.6     computer displays message:
       *  checking
  5.7     computer displays message:
       *  data ok
  5.8  *  press cassette stop    cs1
          then press enter
```

6. Your program is now saved - safely and without error. That's all there is to it!

III. Keeping Your Cassette Tapes Organized - Part I

How many times have you wanted to find a specific program that you had but...

1. You can't remember which cassette you put it on.

2. Or, you can remember which cassette you put in on, but now you can't remember whether you put it on side a or b.

3. Or, you can remember whether you put it on side a or b, but now you can't remember what the counter reading was for the beginning of the program.

4. Or, you can remember what the counter reading was for the beginning of the program, but now you can't remember if the program was written in BASIC or Extended BASIC, or maybe it was that you needed TEII, or was it

Mini-Memory?

If all of this sounds way too familiar to you, don't panic. You are not alone! The same situations have happened to all of us who use a cassette recorder - at least at one point of time or another.

```
**********************************
* The solution - get organized!  *
*    Stop wasting all of your    *
*     valuable computer time     *
*     hunting for a program!     *
**********************************
```

Now that you see the need for some "organization" - let me be one of the first to tell you that there are alot of different ways in which to go about organizing your programs. Keep in mind that while one method may seem to work the best for you, it may not be the best method for someone else. Only you know what method will best meet your own needs!

If you are not using any system right now, I would suggest organizing your programs with the use of 3 x 5 index cards, using the following information as a guideline:

1. Cassette title and/or cassette number
2. Cassette side
3. Program name
4. Counter reading
5. Language used
6. Peripherals needed
7. Program description

That should be enough to get you started and keep you quite busy for awhile. I know that it all sounds like alot of work, but it will be appreciated in the long run, when you need to find a specific program and you don't have all day to hunt for it!!!

III. Keeping Your Cassette Tapes and Programs Organized - Part II

In Part II, I am continuing with the topic of keeping your cassette tapes and programs organized - using the information generated by Part I's 3 x 5 index cards - as the foundation for the following program.

Although this program will work as written - you are encouraged to make any changes that you may want - in order to meet your own personal needs. Don't be afraid to do a little experimenting. It can't hurt and you just may learn a thing or two in the process.

This particular program was created with the intent of giving you the following two options:

1. You may type in the program from Listing 1 as listed - filling in the blanks as they appear or

2. You could just type in the information that would appear in the blank area and forget about typing in all of the "formal titles".

Personally, I like the latter choice myself, as it saves alot of unnecessary repetitive typing, and it keeps my screen information down to a bare minimum when I run the program.

[This series will be continued in future editions of The MANNERS Journal of TI Computing - The Ed.]

NetNotes - Around the TI BBS Network

Dave Ramsey

As the MANNERs representative on Delphi, I try to keep up with events in the TI communcations network. But in addition to Delphi, there are other sources of information and activity for 4A owners. The most important of these, in my own opinion, are the local BBSs that are run by individuals just like you or me, in their own home with a TI-99/4A computer.

One of the finest of these boards is Bob Fowler's and Bill Cavanaugh's "BBBB" Bulletin Board in Clinton, Maryland. BBBB stands for Bob and Bill's Bulletin Board and it is run on a TI-99/4A with four floppy drives. Bob and Bill have added a Hard Disk to the system using the new Myarc Hard and Floppy Disk controller. The addition of a hard disk has expanded the capacity of the BBBB in both messages and programs that you can download. The results have been fantastic!

For information purposes, the BBBB has been operational for about three years now. It has undergone many small changes over the years as Bob and Bill have fine-tuned it to meet their expectations. Because of this, the BBBB is one of the finest and most active BBSs in the TI community. Bob regularly has a new set of programs for you to download, at no cost to you, every month. All he asks is that each user try to upload two new programs each month. Additionally, Bob and Bill are happy to accept donations from anyone who wants to help them defray the costs of operating the BBBB for the Washington area TI community.

Bill Cavanaugh has worked hard over the last three years fine-tuning the software that makes up the BBBB BBS. Bill has also corrected numerous small bugs that existed in the original package.

The BBBB carries apporximately 100 messages at any one time. When the message base fills up, older messages are deleted and new ones take their place. This rollover feature helps to keep the BBBB from overflowing its floppy disks. But it has another effect as well - because the BBBB is so active, it is quite possible for the entire message base to rollover in just a few days. This means that active TIers who are members of the BBBB should call at least twice a week in order to not miss anything. The messages on the BBBB range from questions and problems that TIers at all levels want to solve to technical discussions on both hardware and software. MANNERs hardware gurus, Richard Roseen, Ed Hall, and Richard Cole are all active members and can help you out with hardware problems. Micropendium columnist Mike Dodd is also a participating member and can answer software questions. In addition to Mike, MANNERs member Tom Wible, one of the better C99 programmers in the TI world today, is a contributing member.

All in all, the BBBB BBS is one of the finer and more interesting bulletin boards available to TIers today. And MANNERs members have the advantage that the BBBB is a local telephone call to the DC area. So if you have questions or just want to keep up to date on activities in the TI world, get a modem and become a member of the BBBB. You won't regret it!

Bob and Bill's Bulletin Board is located in Clinton, MD. The phone number is (301) 292-1482. This is a local call for the Washington metropolitan area. Additionally, the BBBB is accessable through Telenet's PC Pursuit service. PCPursuit members can access the BBBB through the DCWAS outdial area. The BBBB accepts calls at 300 or 1200 baud and requires the caller to use 8 data bits, no parity, and one stop bit. After connecting to the BBBB, always press the ENTER key to notify the board that you have connected. Happy modeming!

Unpacking the HFDC

Jim Kwiatkowski

The first week: Nothing but frustration. Not new for me. Let me tell you about my system. Expansion box with foundation 128K card, 512k HRD, Triple Tech card, Gram Kracker, 2 TEAC half height drives. Now a HFDC card with a Seagate 225-20 meg. No box for the seagate, powered by a borrowed power supply from an *#$ 63.5W, both sitting by themselves. Not the best config for a hard drive setup.

I had some problems the first couple of days. Didn't know how to configure the hard disk. I was using MDM5 1.27 that I downloaded from GENIE and reading the manual for 1.21. So there were some differences, the only problem that I had with that was it seemed to take too long. Just around 10 min. to format WDS interlace of 7 and what ever the other one is at 7 also. I think it's interleaving, maybe not. Whatever it is both at 7. Seemed too slow so I tried some others. Right now it's set at 4,3 and I don't seem to be having any problems.

So what kind of problems was I having. Well copying floppy to hard I was getting errors on the hard. Some of that may have been my fault when it would give me a write error I would press (C) to continue. (big mistake). The real frustration came when I tried to use the (EMULATION) feature of the HFDC. It emulates DSK1 (that's great). No it isn't, not if your only copy of MDM5 loads from DSK1. Think about it. It wasn't pretty, I'm glad I was alone. Well I won't use Emulate for a while. I'm one of those guys that likes to try all the features of a piece of software to make sure they work. How did I fix it? Well it is probably unorthadox but this is what I did. Power everything up and then turn off the Hard Disk, loaded MDM5 from DSK1 and then turned the hard disk back on. It probably shouldn't have worked but it did.

The second time I crashed MDM5 while it was formatting (big mistake). Took the better part of 2 hrs. to get the hard disk back up and running. I was really sweating by this time. I kept saying to myself (got to screw around don't you.) How did I do it? Well the way I fixed it the first time I got MDM5 loaded but every time I tried to format the hard disk MDM5 would give me a not found error. So I ended up turning off the hard drive until I was all the way through the formatting sequence. Until you type in format and press enter, of course. It worked - don't ask why.

Now that it is up and running, more problems!!!! Not with the hard disk. Now I have problems with the HRD and floppies. Real problems trashing disks. Maybe using DM1000 has something to do with it. Hate to throw out DM1000. MDM5 is kind of cumbersome, not like DM1000. Especially when formatting using box format. MDM5 is slow in formatting. It automatically verifies. (Rats) Hate to spend my life waiting for verification. I used MDM5 1.22 once. I broke it at 12000 sectors. (life is too short and nothing is that important.) Besides my backup system only has 1 ssdd drive. I'm sure glad MDM5 1.27 arrived so soon. As far as operation the HFDC works great. Fast! Fast! Fast! Can't really tell the difference between it and the HRD. The HRD isn't as noisy.

All I need now is a Power supply. I've thought about buying the Ryte Data 99AT Expansion System. I'll have to look into that a little more. I move like a snail sometimes and after I've made my mind up I want it here yesterday. But isn't that the American way?

Calling MDOS XOPs Directly from C99

Dave Ramsey

On both my Z80 CP/M machine and on the PC AT clones we have at work, I use many of the Borland languages. In particular, I use Turbo Pascal and Turbo C. Both of these languages have functions available to the programmer that make writing extensive assembly language code to interface to the operating system totally unnecessary. After making good use of Clint Pulley's C99 compiler for MDOS, I decided that such a function would be useful on the 9640 also. Since no one else had written such a function, I opted to write it myself.

I elected to name this function mdos(), in keeping with the Turbo Pascal and the Turbo C functions that I use on other machines (the bdos() and the msdos() functions). This is also mnemonic enough to allow easy recall by the programmer.

In order to call any MDOS operating system routine, the application programmer must load certain registers with information and then execute XOP 0 with a particular parameter which indicates which function library is being called. Because the mdos functions use a varying number of registers, I elected to pass an array of 16-bit integers to the function. These integer values would then be loaded into registers R0 through R7 (which are unused by C99). Also, the XOP function code would be passed by the caller and used to index into an array of function codes.

If you examine the program listing (on page 23), you will see that the first order of business within the mdos() function is to validate the XOP function code. The routine tests to ensure that the code is between 1 and 10. The values of 1 through 10 represent the currently valid XOP functions within the MDOS operating system.

Barring an erroneous function code, the routine then obtains the address of the "register array" from the calling program. It then uses this address as a pointer to obtain the eight values to be loaded into registers R0 through R7.

After loading the registers, the routine once again fetches the xop function code from the program stack, doubles it (to make it an effective WORD index value), and then uses that value to make the XOP #0 call.

Upon return from the XOP call, the mdos() gets the pointer to the integer register array from the program stack, reloads the values from the actual registers to the register array, and sets the return code to TRUE (-1, a non-zero value).

A word of caution - because the mdos() function returned a TRUE value does not mean that your XOP call executed as you expected. All that a TRUE value means is that the XOP code was valid and the XOP call was made. The results of the XOP call must be interpreted by the calling program through the examination of the register array. The values of R0 through R7 are available to the caller in that array for manipulation.

In the original version of this routine, I used a zero value to represent a completed function call but that wasn't consistent with the C programming language traditions where 0 = FALSE and NOT 0 = TRUE. If you downloaded an earlier version of the mdos() function from a BBS, you may wish to change the source code as indicated below. Enjoy!

```
/* ***********************************************************

    mdos function - to access mdos system calls from c99
        programs.

    author: dave ramsey              date july 20, 1988

   ************************************************************ */

/* no entry spec for xoptbl - this is private to the mdos function. */
int xoptbl[11] = {0,1,2,3,4,5,6,7,8,9,10};

entry mdos;     /* make it available to other routines */
/* the function call itself */
mdos(xopcod, xopreg)
int xopcod, *xopreg[];
{
#asm
        LI  8,0           * 0 = default return value (FALSE)
        MOV @4(14),4      * get value of xopcod
        CI  4,1           * is xopcod < 1?
        JLT MDOSER        * if yes, take error exit
        CI  4,10          * is xopcod > 10
        JGT MDOSER        * if yes, take error exit
        MOV @2(14),8      * get pointer to register array
        MOV *8+,0         * load register array values into R0 - R7
        MOV *8+,1
        MOV *8+,2
        MOV *8+,3
        MOV *8+,4
        MOV *8+,5
        MOV *8+,6
        MOV *8+,7
        MOV @4(14),8          * get requested XOP code
        A   8,8               * double value for correct indexing...
        XOP @XOPTBL(8),0      * execute system call
*
        MOV @2(14),8      * get pointer to register array
        MOV 0,*8+         * save R0 - R7 into register array
        MOV 1,*8+
        MOV 2,*8+
        MOV 3,*8+
        MOV 4,*8+
        MOV 5,*8+
        MOV 6,*8+
        MOV 7,*8
        LI  8,-1          * return TRUE since function executed
*
MDOSER  EQU $
#endasm

} /* end mdos function */
```

COMING EVENTS

For January-February 1989

Some upcoming events in and near the Washington, DC area have been passed to the newsletter. These events are listed below.

Computer Show, Sale, & Flea Market
January 28-29, 1989
Saturday 10-4, Sunday 10-3
Howard Johnson's Plaza
New Carrolton, MD
(North of DC on the beltway)

Philadelphia Computer Show
George Washington Convention Center
Willow Grove, PA
(Turnpike exit #27)
January 7-8, 1989

If you have information on other computer shows or events of interest to other MANNERS members, please forward them to the newsletter editor. Thanks!

The next MANNERS monthly meeting will be held on Thursday, January 12, 1989 at the Fairfax High School. The meeting will begin at approximately 7:30 but some members may be present prior to that for informal discussions and sales or TI hardware and software. See you there!

NOTE: The mailing address listed below is temporary and will change next issue!

The MANNERS Journal of TI Computing
1188 Green Holly Drive
Annapolis, MD 21401

Dallas TI Home Computer Grp
P O Box 29863
Dallas TX 75229

12/31/99 C