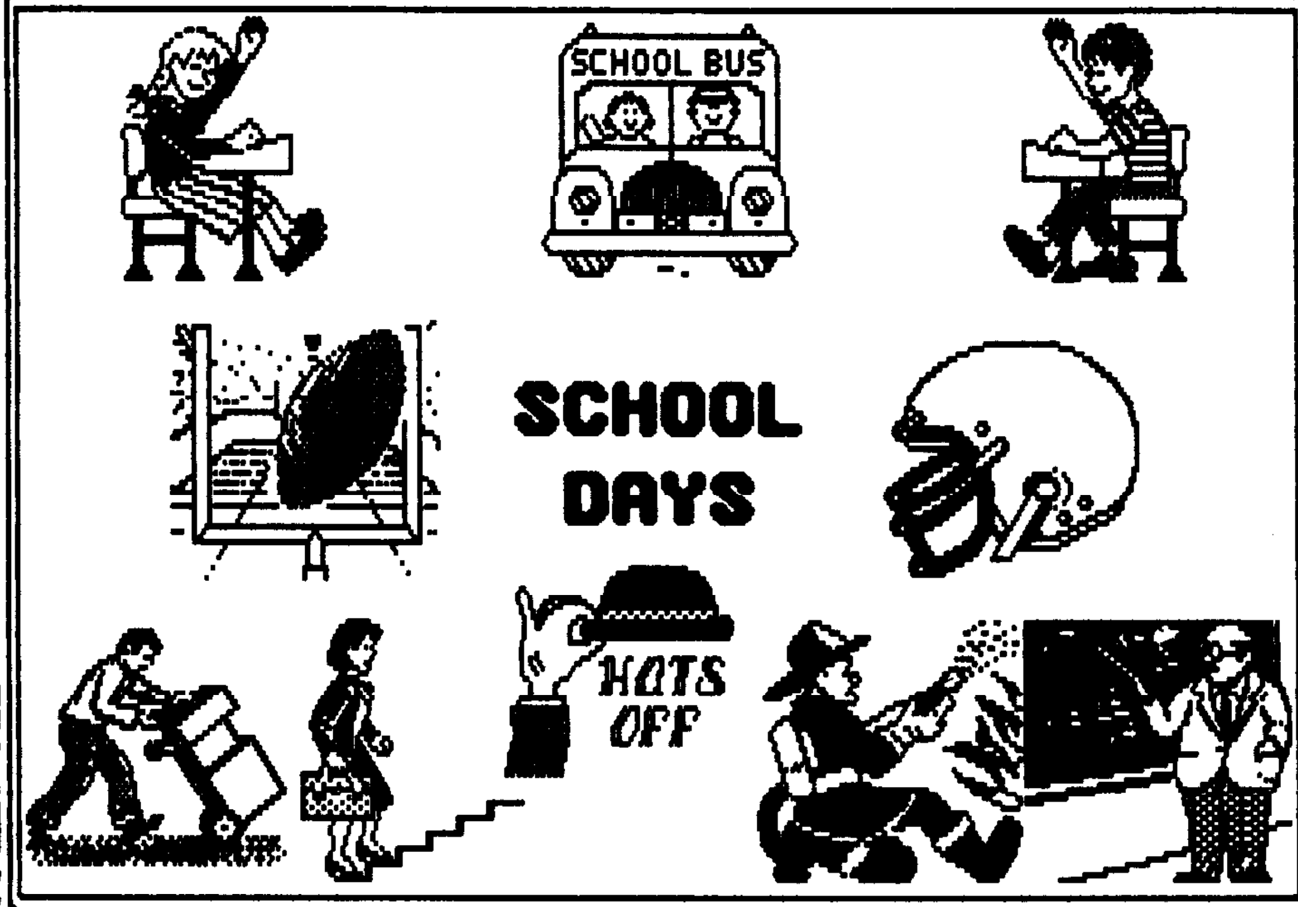
NEWJUG 99ER'S NEWS

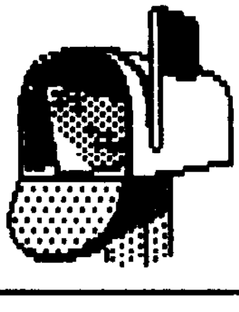
NEWJUG 99ER'S NEWS

SEPTEMBER 1990



HIGHLIGHTS:

HOTBUG VERSION 1; JOKE OF THE MONTH; EDITOR'S FORUM; 99/4A AND 9640 VENDORS; THE EXCHANGE; COMING EVENTS; ACCESSING INTERNAL FILES IN C99



NEWJUG 99ER'S UG
P.O. BOX 1463
SAYREVILLE, N.J.
08871-1463

HOTBUG VERSION 1.0

REPORT CARD	
PERFORMANCE	A+
EASE OF USE	A+
DOCUMENTATION	A-
VALUE	A
FINAL GRADE	A

How many times have you discovered a utility or a program which you had in your possession for awhile but never got around to taking a good look at the product? For me, HOTBUG was that product. This is not the type of program that everyone will have a use for. Heck it took me close to a year to experience a programming problem serious enough to warrant a debugger. With c99 being my primary programming language, I was usually able to debug most of my programs by putting print statements in the body of the program. Not very efficient, but it worked for me.

The authors of HOTBUG realized that the debugger should be flexible enough to permit one to debug just about any program. To resolve this situation, you have a choice of three different debuggers (TI99/4A, Geneve and Remote) which can be located in low memory (>2000), high memory (>E000) or cartridge ram (>6000). I'm sure you won't be able to take advantage of all nine debuggers, but I doubt that you'd ever require all of them. I have found the supercart version to be the most versatile for me, simply because I haven't had to debug anything which loads into (>6000) that memory.

With the software comes 14 pages of documentation which cover all the commands available with the debugger. The debug process is rather simple to initiate. You first load the version of the debugger that you plan to use. I choose the supercart version to insure that the program I want to debug, will not overlay HOTBUG. However, if you do not have a supercart device, the bounds of the actual program to be debugged, can be mapped out by viewing the first six bytes of each memory image file. The first two bytes indicate whether additional image files follow. The next two bytes indicate the length and the last two bytes indicate the start address of where to load the software. For example, the first six bytes of the file UTIL1 (FunnelWeb v4.21 shell) contains >0000 >1FD2 >E006 which means that there are NO additional modules, and the program resides in memory between >E006 and >FFD8.

To load the program you wish to debug, use the LOAD command at the HOTBUG prompt ">". For example, to debug the file UTIL1 which resides on DSK3, you would enter the following command: LOAD DSK3.UTIL1. When the prompt comes back, you are now ready to debug your program.

If you have a program to debug, you'll need a hard copy of the assembled source code to be able to set break points. Setting breakpoints indescriminately without regard to instruction bounds, can create console lockups. If you don't have the source available, you can always use the starting address of the first module to set a starting point. If you aren't quite sure about address bounds, you can have HOTBUG display disassembled code on the screen by entering the UB command. If you'd like to see things change on each instruction, just type G without a count value (it will act like a STEP instruction). The neat thing about this debugger is that it puts everything you'll need pertaining to registers, breakpoints, program counter, status word, etc. right on the screen. The debugger comes with some nifty calculator functions so you won't need a calculator by your side either.

As you would expect, it has the ability to alter memory, registers, workspace, status word, or the program counter. You can set up to 5 breakpoints in the program. As a breakpoint is reached, it will disappear from the list of active breakpoints. This gives you the option to continually set new breakpoints without worrying about exhausting the number you can set. If the breakpoint is within a loop and you desire to force the program to break at the same location, just step to the next instruction via the G command and set the breakpoint again.

If you accidently resume the program without setting a breakpoint, you can get back to the debug screen by hitting the "hot" key (Ctrl and Fctn keys simultaneously for the 4a). The docs mention this and recommend that you enter the LINK command if your software doesn't enable interrupts. \

If you reach a breakpoint and would like to see what the application screen looked like at the moment of the breakpoint, simply type T at the HOTBUG prompt. This will trade screens for you and let you view the application. Pressing another key returns you back to the HOTBUG screen.

There are quite a few other commands available which are quite intriguing. The ability to assemlbe code at an absolute address lets you write and test patches on the fly. This would be something that only the more experienced user may take advantage of, but it's nice to know that it exists.

Normally, most articles or reviews would end here. I'd like to add my endorsement for this program. For the past few months, I've been experiencing problems with a memory image loader that runs in a c99 environment. Programs which load correctly from the E/A cartridge would lock my console when my c99 loader attempted to load them. The problem programs where the ones which loaded into low memory. USing this debugger, I was able to isolate the problem and resolve it by writing routines which emulate the video RAM routines in the E/A cartridge.

If this software was sold as a commerical product, I'd most likely wouldn'] have bought it. This software is FAIRWARE and well worth the registration price of $20.00. If you haven't tried it and have a need for such a program, I suggest you give it a whirl. If it's worthwhile to you, please register it so we can inspire people like Charles Earl to continue writing programs for the TI community.

Accessing Internal Files in c99

By Dan Gazsy

When Clint Pulley first released the c99 compiler, an individual by the name of Tom Bentley had the foresight to provide library functions for the floating point accumulator (FAC) and a general purpose I/O library called TCIO. Initially I didn't find much of a need for the FLOAT library other then to implement SIEVE type programs.

Two months ago, I decided to write some c programs to perform file maintenance on a BBS message base. My first course of action was to check around to see what c code had been written along these lines. I found the FLOAT and TCIO libraries were all I needed to implement my application. As an added bonus, I ran across a disk catalog function written by Tom Bentley which used both libraries.

Before you write any applications to process internal type files, you should have an understanding of how internal records are structured. When the file is created, you allocate the number of bytes for each record. The fields associated with each record can be either numeric fields or string fields. Numeric fields use 9 bytes; the first byte contains the length of the field (always 8 bytes) used to represent the value in float format. String fields are stored in the same manner as DV80 or DF80 strings. The first byte contains the string length; immediately followed by the string. The fields are stored in the record in the sequence they appeared in the print statement (or however the file was created). All unused characters are filled with nulls (0). For example if I created a file of 70 characters and wrote 3 fields (name, address and age) to it, they would look like the following:

```
 byte 01     - length of name string
 byte 02-10  - Name string ("Dan Gazsy")
 byte 11     - length of address string
 byte 12-24  - Address string ("22 6th Street")
 byte 25     - length of age field (always 8)
 byte 26-33  - Age field ("37")
 byte 34-70  - Null characters
```

Great you might say! Now how do I read these records from a file into my c program and make decisions based on the values of the fields? Well to start, you open the file with "topen" and read the records with a "tread". In the event you are not sure of the size of the input record, set up an array of 256 bytes to be on the safe side. Once you complete the "tread", the entire record is in the array you specified as the buffer argument. Now all that's left is to parse the record into fields. To do this, you must know how many fields are in each record, what type field they are (string/numeric) and the order they appear in the record. Below appears six functions which we will use to parse the records. The first two functions (getstr and getnum) will be used to parse the input record and the next two functions (putnum and putstr) are used to parse the output record. The last two functions (strlen and getfn) are used to size strings and to open TESTFILE for input or update mode. We lumped all of these into this section because it's what I'd consider a utility. In the interests of good programming practice, we'll compile the following 79 lines of code as you would create a support library. In addition to compiling this program, you'll also have to assemble it. For test purposes, let's call these 79 lines UTILITY:C and the output of the c compiler will be called UTILITY:S. As expected the input file for the Assembler will be called UTILITY:S and the output will be called just plain old UTILITY. Congratulations, you've just created a support library!

```
entry getstr, getnum, putnum, putstr, strlen, getfn;
#include "dsk1.floati"
#include "dsk1.tcioi"

getstr(buff, t)
     int *buff;
     char *t;
     { char *b;
       int j, siz;
       b=*buff;                        /* set up b with byte
                                     address of input rec */
       j=siz=*b++;                     /* set up j & siz with
                                        # bytes in string */
       *buff=*buff + j + 1;  /* push to next input
                                                   field */
       while(j--)
          *t++=*b++;      /* transfer string from input
                                           rec to string */
       *t = '\000';       /* terminate with null byte */
       return(siz);       /* return size of string */
     }

getnum(buff, t, f)
     int *buff;
     char *t;
     float *f;
     { char *b;
       b=*buff;                        /* set up b with byte
                                        address of input rec */
       ++b;                            /* point data past the
                                        float size indicator */
       *buff = *buff + 9;
       fcpy(b, f);                     /* copy the input field
                                               into the float */
       ftos(b, t, 0, 0, 0);            /* convert field to
                                              string variable */
     }

putstr(buff, t)
     int *buff;
     char *t;
     { char *b;
       int j, siz;
       b=*buff;                        /* set up b with byte
                                        address of output rec */
       j=siz=strlen(t);                /* set up j & siz with
```

```
                                                   # BYTES IN STRING */
          *BUFF=*BUFF + J + 1;  /* PUSH TO NEXT INPUT
                                                          FIELD */
          *B++=J;                     /* PUT STRING LENGTH IN
                                            1ST BYTE OF FIELD */
          WHILE(J--)
             *B++=*T++;          /* TRANSFER STRING TO OUTPUT \
                                                              REC */
          RETURN(SIZ);           /* RETURN SIZE OF STRING */
       }

    PUTNUM(BUFF, T, F)
       INT *BUFF;
       CHAR *T;
       FLOAT *F;
       { CHAR *B;
          INT I;
          B=*BUFF;           /* SET UP B WITH BYTE ADDRESS \
                                                 OF OUTPUT REC */
          *B++=8;            /* PUT FLOAT ID IN OUTPUT REC */
          *BUFF = *BUFF + 9;
          WHILE(*T==' ') /* GET RID OF LEADING SPACES */
             T++;
          STOF(T, B);       /* COPY THE STRING FIELD
                                       INTO OUTPUT RECORD */
       }

    STRLEN(S)
       CHAR *S;
       { INT N;
          N=0;
          WHILE(*S++)
             ++N;
          RETURN(N);
       }

    GETFN(TEXT, M)
       CHAR *TEXT;
       INT M;
       { INT UNIT;
          UNIT=0;
          IF(M==0)
             UNIT=TOPEN(&TEXT[0],
                              INPUT+RELATIVE+INTERNAL+FIXED, 0);
          ELSE
             UNIT=TOPEN(&TEXT[0],
                             UPDATE+RELATIVE+INTERNAL+FIXED, 0);
          RETURN(UNIT);
       }

```

TO DEMONSTRATE THAT THIS ISN'T MAGIC AND IS COMPATIBLE WITH FILES CREATED BY OTHER PROGRAMMING LANGUAGES, I'VE CREATED THE INPUT FILE FROM A BASIC PROGRAM!

```
100 OPEN #1:"DSK1.TESTFILE",RELATIVE,INTERNAL,OUTPUT,
    FIXED 70
110 A$="DAN GASSY"
120 B$="22 6TH STREET"
130 C=37
140 PRINT #1,REC 0:A$,B$,C
```

150 CLOSE #1

What follows now is simple C program to read the file we created in the above Basic program and display the fields. We will be referencing the library functions we created earlier and code will be included in our main program which will demonstrate how to reference these functions.

TESTREAD;C listed below:

```
extern getfn(), getstr(), getnum();
#include "dsk1.tcioi"
#include "dsk1.floati"
#include "dsk1.stdio"

main()
  { int fp, b_ptr, size, i;
    char buff[256], sdum[81];
    float fdum[FLOATLEN];
    putchar(FF);
    for(i=0;i<256;i++)
       buff[i]=0;
    fp=getfn("DSK1.TESTFILE", 0);
    if(fp>0)
       {tread(buff, 0, fp, &size);
        b_ptr=buff;
        getstr(&b_ptr, sdum);
        puts("Name: ");
        puts(sdum);
        getstr(&b_ptr, sdum);
        puts("\nAddress: ");
        puts(sdum);
        getnum(&b_ptr, sdum, fdum);
        puts("\nAge: ");
        puts(sdum);
       }
    else
       puts("Error, file could not be opened!");
    tclose(fp);
    puts("\n\nStrike any key to continue");
    getchar();
  }
```

The goal of the function called "main()" has a rather simple purpose. It is to open the file called "TESTFILE", read record 0 into a buffer, parse the buffer into known fields and finally display them. As you can see, I named the program TESTREAD;C, the output code of the compiler was called TESTREAD;S and the assembler output code was named TESTREAD. To execute the program you will have to specify the following files TESTREAD, UTILITY, TCIO, FLOAT and CSUP.

The last part of this column will display a method to combine fields entered on the screen into a record; then write the record back to the file TESTFILE. Like the earlier test, I called this one TESTWRIT;C, the output of the c99 compiler was called TESTWRIT;S and the assembler output was called TESTWRIT. To execute this program you'll need to load the following files: TESTWRIT, UTILITY, TCIO, FLOAT and CSUP. Again remember these are load and run

(OPTION 3) FILES.

TESTWRIT;C LISTED BELOW:

```
EXTERN GETFN(),PUTSTR(),PUTNUM(),STRLEN();
#INCLUDE "DSK1.TCIOI"
#INCLUDE "DSK1.FLOATI"
#INCLUDE "DSK1.STDIO"

MAIN()
  { INT FP,B_PTR,SIZE,I;
    CHAR BUFF[256],SDUM[81],NDUM[81];
    FLOAT FDUM[FLOATLEN];
    PUTCHAR(FF);
    B_PTR=BUFF;
    FOR(I=0;I<256;I++)
       BUFF[I]=0;
    WHILE(1)
       { PUTS("ENTER NAME (MAX 20 CHARS):");
         GETS(NDUM);
         SIZE=STRLEN(NDUM);
         IF(SIZE<21)
            BREAK;
       }
    PUTSTR(&B_PTR,NDUM);
    WHILE(1)
       { PUTS("ENTER ADDRESS (MAX 20 CHARS):");
         GETS(NDUM);
         SIZE=STRLEN(NDUM);
         IF(SIZE<21)
            BREAK;
       }
    PUTSTR(&B_PTR,NDUM);
    WHILE(1)
       { PUTS("ENTER AGE (MAX 3 DIGITS):");
         GETS(NDUM);
         SIZE=STRLEN(NDUM);
         IF(SIZE<4)
             BREAK;
        }
    PUTNUM(&B_PTR,NDUM,FDUM);
    FP=GETFN("DSK1.TESTFILE",1);
    IF(FP>0)
       {PUTS("WRITING RECORD");
        TWRITE(BUFF,0,FP,70);
        TCLOSE(FP);
       }
   }
```

TO INSURE THAT THE FILES WRITTEN IN C99 ARE COMPATIBLE WITH OTHER LANGUAGES, WE'LL READ THE FILE AND DISPLAY THE FIELDS WITH A BASIC PROGRAM.

```
100 OPEN #1:"DSK1.TESTFILE",RELATIVE,INTERNAL,INPUT,FIXED 70
110 INPUT #1,REC 0:A$,B$,C
120 PRINT "NAME: "&A$
130 PRINT "ADDRESS: "&B$
140 PRINT "AGE: "&STR$(C)
150 CLOSE #1
```

JOKE OF THE MONTH

As he conducted his investigation of a deadly five-car accident, Detective Cook spotted a monkey sitting on the hood of a wrecked car. When he was ready to leave, he put the animal in his car and drove toward the county zoo. "I wish you could tell me what happened back there," the cop mused. The monkey nodded its head. "OK, what happened?" Cook asked. The monkey raised its hand to its mouth in a drinking motion. "So they were drinking, is that all?" The monkey shook its head and brought its hand to its mouth, pretending to smoke. "So they were drinking and smoking. Is that all?". The monkey shook its head and brought its hands together to indicate that intercourse occurred. "Ah, they were drinking, smoking and engaging in intercourse," Cook said. "And what the heck were you doing?" The monkey raised his hands in a driving motion and craned its neck over its right shoulder.

THE EXCHANGE

Our list of TI User groups that we exchange newsletters with continues to grow. Our list of 25+ UG's only took 5 months to compile. If your club exchanges with any UG's not on our list, we be very interested in hearing from you via the comments and suggestions on the cover page.

CLUB 99
Mail Stop 1-21
34 Forest Street
Attleboro, MA 02703

North Jersey TI IBM UG
16 Judith Ann Dr.
Ringwood, NJ 07456-1863

Central Garden State UG
61 Country Lane
Hamilton Square, NJ 08690

QB99er's User Group
c/o Frank Crotty
Queensborough Comm College
Bayside, NY 11364

LITI 99er's UG
93 Myers Avenue
Hicksville, NY 11801-2424

Twin TIers UG
c/o R. Sass
RD 01
Rock Stream, NY 14878

Pittsburg User's Group
P.O. Box 8043
Pittsburg, PA 15216

Erie 99'er User Group
2812 West 33rd Street
Erie, PA 16506

Nittany Users of TI
625 Wiltshire Drive
State College, PA 16803

Hampton Roads TI'ers
4701 Atterbury Street
Norfolk, Va 23513

CONNI
101 Heischman Ave
Worthington, OH 43085

Greater Akron 99er's
P.O. Box 3201
Cuyahoga Falls, OH 44223

Lima 99/4a Users Group
P.O. Box 647
Venedocia, OH 45894

Great Lakes Computer Group
P.O. Box 152
Roseville, MI 48066-0152

Milwaukee Area Users Group
4122 N. Glenway
Wawatosi, WI 53222

Siouxland 99er's
4604 Bluestem Circle
Sioux Falls, SD 57106

Kansas City TI99/4a UG
P.O. Box 12591
No. Kansas City, MO 66416

Dallas TI Home Computer
P.O. Box 29863
Dallas, TX 75229

Net99er HCUG
P.O. Box 534
Hurst, TX 76053

JSC TI99 User Group
c/o John Owen
2321 Coryell Street
League City, TX 77573

TI SLaVes
3810 W. 6540 So.
West Jordan, UT 84084

Southwest Ninety Niners
P.O. Box 17831
Tucson, AZ 85730

Southern Nevada UG (SNUG)
P.O. Box 26307
Las Vegas, NV 89126-0301

Northern Nevada 99'ers
5554 Mark Circle
Sun Valley, NV 89433

North County 99ers UG
P.O. Box 2500
Escondido, CA 92025

The ROM Newsletter
Users Group Orange County
17161 Edwards Street
Huntington Beach, CA 92647

99/4A & 9640 VENDORS

Aldoes Computer Supplies
6298 Hamilton Rd.
36 Main Street Village
Columbus, GA 31909
(404) 327-4900

Asgard Software
P.O. Box 10306
Rockville, Md 20850
(703) 255-3085
Catalog Available

Braatz Computer Services
719 East Byrd Street
Appleton, Wisc 54911
(414) 731-3478
(414) 731-4320 after 6pm
Catalog $2

Bud Mills Services
166 Dartmouth Dr
Toledo, Ohio 43614
(419) 385-5946

CaDD Electronics
52 Audubon Rd.
Havermill, Ma 01830
(603) 895-0119

Competition Computer
2629 W National Ave
Milwaukee, Wisc 53204
(800) 242-7902 in Wisconsin
(800) 662-9253 all others
Catalog $1

Computer Shopper
P.O. Box F
Titusville, FL 32781

CorComp
2211-G East Winston Rd.
Anaheim, Ca 92806
(714) 956-4450

Dijit Systems
4345 Hortensia St
San Diego, Ca 92103
(619) 295-3301 voice
(619) 278-8155 bbs

Disk Only Software
P. O. Box 244
Lorton, Va 22079
(310) 340-7179

Genial Computerware
835 Green Valley Dr.
Philadelphia, Pa 19128
(215) 483-1379

Great Lakes Software
804 E. Grand River Ave.
Howell, Mi 48843
(517) 546-0566

Harrison Software
5705 40th Place
Hyattsville, Md 20781
(301) 277-3467

Hunter Electronics
4 N. 370 Pine Grove
Bensenville, Il 60106
(312) 766-9503

Inscebot Inc.
P. O. Box 29160
Pt Orange, Fl 32029

Jim Lesher
722 Huntley
Dallas, Tx 75214
(214) 821-9274

Joy Electronics
P. O. Box 542546
Dallas, Tx 75354-2526
(800) 422-3892 in Texas
(800) 527-7438 all others

JP Software
2390 El Camino Real #107
Palo Alto, Ca 94306
Catalog #1

LaFlamme & Wrigley Wholesale
5480 Canotek Road
Unit #16
Glouchester, Ontario K1J9H6
(613) 745-2225

L. L. Conner Enterprise
1521 Ferry St.
Lafayette, In 47904
(317) 742-8146

McCann Software
P. O. Box 34160
Omaha, Ne 68134

Micropendium
P. O. Box 1343
Round Rock, Tx 78680
(512) 255-1512

Midwest Engineering
203 Arcadia Dr.
Vernon Hills, Il 60061
(312) 362-9034

Myarc Inc
2624 Ranier Drive NE
Birmingham, Al 35215
(205) 854-5843

Not Polyoptics
P. O. Box 4443
Woodbridge, Va 22191
(703) 499-5543

Pilgrims Pride
5 Williams Ln.
Hatboro, Pa. 19040
(215) 441-4262

Quality 99 Software
1884 Columbia RD #1021
Washington, DC 20009
(202) 667-3574

Queen Anne Computer Shoppe
6102 Roosevelt Way NE
Seattle, Wa 98115
(206) 522-6558

Ramcharge Computers
6467 E. Vancey Dr.
Brookpark, Oh 44142
(216) 243-1244 evenings

Rave 99
112 Rambling Road
Vernon, Ct 06066
(203) 871-7824

Tenex
P. O. Box 6578
South Bend, In 46660
(800) 348-2778
(217) 259-7051

Texaments
53 Center St
Patchogue, N. Y. 11772
(516) 475-3480
(516) 475-6463 bbs

Tex-Comp
P.O. Box 33084
Granada Hills, CA 91344
(818) 366-6631
Catalog #2

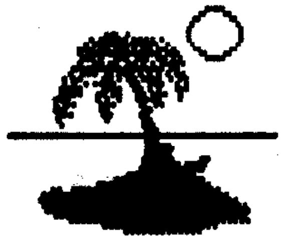
The Bunyard Group
P.O. Box 62323
Colorado Springs, CO
80962-2323
(719) 488-2572

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213
(614) 235-3545

Trio+ Software
P.O. Box 114-A
Lisbon, IA 50148

Triton Products Company
P.O. Box 8123
San Francisco, CA 94128
1-800-227-6900
Catalogs Available

9640 News
c/o Beery Miller
5455 Marina Cove #1
Memphis, TN 38115

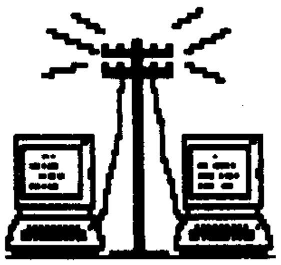


EDITORS FORUM

In this month's Forum there are two items I'd like to devote some attention. The first item concerns my long awaited subscription to Vulcan Computer Monthly (aka. Computer Buyer's Guide). Well the subscription delivery finally started and to my dismay, there was no TI Column in the August issue. Checking into the classified ads, I did find a small section of TI vendors. I'm hoping right now that the TI columnist simply missed the deadline. The periodical does dedicate some of it's space to the Classic computer, so if you have more than one, it's worth picking up a copy at your local newstands. The second item of discussion is the lack of an apparent successor to Myarc. Up till now, many of the third party vendors had been willing to sit back and watch Myarc develop major hardware components (Geneve, HFDC, etc.). Other vendors like RAVE 99 and Bud Mills were satisfied delivering specific items like the RAVE keyboard, Horizon Ramdisk, etc.

The community had become accustomed to the Myarc promised delivery dates which came and went. For most, it was the norm to wait for months for promised hardware. Recently, we have seen an renewed interest on the part of vendórs and developers to supply alternative hardware solutions to the community in a responsible time frame. If you were considering the clone world as an alternative for this reason, please give the new breed (Asgard, Rave 99, Bud Mills Services, OPA, Ron Walters - MEMEX) a chance to change your mind.

Till next month---ZZZZ



BBS SCENE

This month's column will be replaced by the editor's vacation. To insure that a timely newsletter would be available, I had to make some sacrifices. Too be continued next month...

COMING EVENTS

SEPTEMBER

SEPTEMBER 15

NORTH JERSEY - WM PATTERSON COLLEGE - REC. CTR. 400 TABLES WAYNE, NJ - ALL INDOORS - NEAR I80, RTE 23 AND RTE 46 - ABOUT 30 MIN. TO NYC SAT. 10 AM TO 4 PM.

SEPTEMBER 22

SEATTLE TI CONVENTION CALL QUEEN ANNE COMPUTER SHOPPE TIBBS (206) 5461865 FOR ADDITIONAL INFO

SEPTEMBER 23

PHILADELPHIA AREA - VALLEY FORGE CONV. CENTER - OVER 400 VENDOR TABLES. AT PA TURNPIKE EXIT #24, RIGHT AFTER TOLL - 1 MILE ON RIGHT - NEXT TO SHERATON IN KING OF PRUSSIA, PA SUN. 10AM TO 3PM.

OCTOBER

OCTOBER 7

4TH ANNUAL CPUG COMP./ELECT. EXPO COCOA AVE PLAZA, 605 COCOA AVE, HERSHEY, PA. PRE-REGISTRATION THRU AUG 3RD. WRITE TO CENTRAL PA 99/4A USERS GROUP, P.O. BOX 14126, HARRISBURG, PA 17104-0126 OR CALL D RATCLIFFE (717) 238-5414 OR THE DATA FACTORY BBS(717) 657-4992 OR 4997 (24 HOURS 8-N-1 300/2400. SUN 7AM TO 3:30PM

OCTOBER 27-28

COLUMBIA N'WEST TI COMPUTER FAIR, JANTZEN BEACH RED LION INN, PORTLAND OR. SPONSORED BY NOVA (NINETYNINER OF THE VANCOUVER AREA), WASHINGTON AND PUNN(PORTLAND USERS OF NINETY NINES), OR. CONTACT MICHAL CALKINS, 1215 S.W. CEDAR ST., LAKE OSWEGO, OR 97034 OR (503) 636-1839.

NOVEMBER