******* SUPER BUGGER VERSION 3.1 19-FEB-1983 ********

SUPER-BUGGER is a stand alone program that may be loaded by the Editor/Assembler LOAD AND RUN option, or either TI-BASIC or EXT-BASIC CALL LOAD Options. NO special hardware is required, but this program will operate only on the TI99/4A with MEMORY EXPANSION,and a DISK CONTROLLER. The RS232 may optionally be used to get a hard copy printout on some operations.

The SUPER-BUGGER is a very sophisticated and powerful debug tool which can provide funtions usually only available on very expensive development systems requiring special hardware. SUPER-BUGGER allows you to actually step through your machine language program, executing each machine instruction one at a time. Enabling you to examine the logic of your program as it is being run. As each instruction is executed, the SYMBOLIC interpretaion is displayed on the screen in the same format as it occurs in your assembley source listing, providing a trace of instruction execution.

SUPER-BUGGER has a built in DISS-ASSEMBLER which you can use to decipher machine code to it's symbolic assembly language representation. It will interpret any instruction and show all types of operand uses. even displaying the JMP address of jump instruction.

Operation of the SUPER-BUGGER is syntactally identical to the TI-DEBUGGER program, however there are features provided by the TI program that are not supported by SUPER-BUGGER due to memory size limitations. It is recommended you become familiar with the TI-DEBUGGER and it's documentation prior to using SUPER-BUGGER.

The following is a summary comparison of the two debuggers. Both the TI-DEBBUGER and SUPER-BUGGER can be used to compliment each other to provide the best development tool in the industry.

CAMMAND	TI-DEBUGG	SUPER-BUGGER
A	Load Memory with ASCII	DISS-ASSEMBLE machine code to Nmunonic
B	Breakpoint Set/Clear	Same except always two word Breakpoints
C	CRU Inspect/Change	** Not Supported
D	** Not Supported	Dump memory to HARD COPY DEVICE
E	Execute	Same
F	Find Word or Byte	** Not Supported
G	GROM Base change	** Not Supported
H	Hex Arithmetic	** Not Supported
I	Inspect Screen Location	** Not Supported
K	Find Data Not Equal	** Not Supported
L	**Not Supported	Hard Copy List device Toggle on/off
M	Memory Inspect/Change	Same
N	Move Block	** Not Supported
P	Compare Memory Block	** Not Supported
Q	QIUT Debugger	Same
R	Inspect/Change WP,PC,& SR	Same
S	Step with special Hrdw.	Single Step on any TI99/4A
T	Trade Screen	Trades user screen with SBUG screen
U	Toggle Basic offset on/off	Same
V	VDP Base change	Run till VALUE = entered number
W	Inspect/Change Register	Same
X,Y,or Z	Change BIAS	Same
>	Hex to Decimal convert	** Not Supported
.	Decimal to Hex convert	** Not Supported

SUPER-[illegible] PROCEDURES

The syntax for SUPER-BUGGER is the same as the TI-DEBUGGER. The following conventions are used. Items surrounded by <angle brackets> represent mandatory data to be provided. Items surrounded by {braces} indicate you must choose between the two or more items included. Items surrounded by [brackets] indicate optional data. The elipsis (...) indicates the previous item may be repeated.

The SUPER-BUGGER is located on the diskette in two versions. The version named "[illegible]" is stored in DISPLAY/FIXED object code format. The Version named "[illegible]" is stored in condensed format which can be loaded by the Editor/Assembler loader. The condensed format cannot be loaded by TI-BASIC or EXTENDED BASIC loaders.

STEP 1. [illegible] SUPER-BUGGER

The "[illegible]" program is relocatable and can be loaded the same way the TI "DEBUG" program is loaded, ie; the LOAD AND RUN Option on the Editor/ Assembler or with CALL LOAD From TI-BASIC and EXT. BASIC. The name of the file to load is "DSK1.SBUG", (if the diskette is in disk drive 1). Enter the SUPER-BUGGER exactly as you would enter TI-DEBUGGER.

If entered by the Editor/Assembler Load and Run option, then Select LOAD AND RUN from the menu and at the promp:

```
FILE NAME?
```

Enter the file name of Your DEVELOPING object code program in the format (DSKn.Filename).

After the file is loaded, the filename is erased from the screen and you may load other program modules. The SUPER-BUGGER module should be the last program module loaded. Enter the following:

```
DSK1.SBUG     or whatever disk drive has SUPER-BUG
```

after SBUG is loaded, you may proceed by pressing <ENTER> without entering a filename.

The prompt

```
PROGRAM NAME?
```

appears and you must enter the SUPER-BUGGER Program first in order to gain control over your developing program. So, enter,

```
SB[illegible]
```

When [illegible] entered, the ID is displayed

```
*** SUPER-BUG VER 3.1 ***
```

STEP 2. Select User screen type.

The message will ask you to select the type of USER screen you are using.

  * BIT MAP SCREEN (Y OR N)

Press <N> if using the TI graphic screens.
When using this option SBUG will share the graphic screen
with the users program screen.

press <Y> to set up screen address which allow the user screen
to be automatically restored to BIT MAP Mode upon exit from SBUG into the
user program. This is extreemly helpful when developing software that uses
different screen modes than the standard TI system screens.
You may use the <T> option to flip to the user screen when in SBUG.

*** IMPORTANT *** YOU CANNOT USE BITMAP SCREEN IF RUNNING UNDER BASIC.

VDP Screen Memory Map for BIT MAP MODE under SBUG

SUPER-BUG	USER	contents
>3800	>0000	PATTERN DESCRIPTOR TABLE
>3C00	>1C00	SCREEN IMAGE TABLE
>3B00	>1F00	SPRITE ATTRIBUTE TABLE
>NOT USED	>1800	SPRITE DESCRIPTION TABLE
>3B80	>2000	COLOR TABLE
>3F00	????	PAB'S
>3F20	????	PAB BUFFER TO >3FFF

The eight bytes of VDP WRITE ONLY Register values are located at
>0132 (relative to loading address of SBUG).
Use the M option to alter these locations and provide
your own VDP screen locations if you want to change screen addresses
in your program. (The entry point of SBUG is located at >0000 relative to load
address.)

STEP 3. Enter the list device.

The message will instruct you to enter the list device.

  ENTER LIST DEVICE

Press <ENTER> to use the default LIST device of;

  RS232.BA=4800

Or, Enter any output device such as a DSKn.Filename
or an RS232 device. (When using BIT MAP Screen you cannot use a DISK
device for output as the VDP addresses used for SBUGGER will be used
by the disk routines).

This device is then used for output on the
D(ump) and A(ssemble) commands only. With the use of the L(ist) commannd
you can select to print a hard copy of memory dumps and diss-assembly
listings. If no hard copy is desired. You may skip this prompt by pressing
return with out entering a device name, but you must also turn the List
device OFF By using the L(ist) command.

If BIT MAP screen is not selected, then
SUPER-BUGGER will not save the screen upon entry.
Therefore, the user screen is used by the SBUG program to display it's
messages. However, The screen offset will be automatically set for BASIC
when "SBUG" is loaded by the BASIC loaders.

STEP 4. Find the entry point of your program.

To Find the entry point of your program in memory, you can inspect the
REF/DEF table beginning at >3FB0. The 8 Byte table has the name of your
program as defined by the DEF entry address. The First 6 bytes of each
table entry contains the REF/DEF Name, and the WORD following is the
entry point for that name.

Use the <M> option to scan through the REF/DEF table and locate the
address of where your program is entered.

You must also determine where your program has been loaded. If you loaded
your program with the LOAD and RUN Option from the Editor/assembler, and
it was the first program, it will be placed at address >A000. Therefore,
it will be handy to enter an offset or relocation BIAS into one of the
X, Y, or Z BIAS locations.

Before you can actually ENTER your developing program, you must first set
up the WP, PC and SR must be set for your program.
Locate these values from your listings, then add your BIAS to these addres
and enter with the <R> option.

At this point you may enter your program by the <S> option or the
to single step through your program.

*** IMPORTANT ***

Prior to using the <E> Option you must place a Breakpoint in your program
at a place where you expect to stop. otherwise, there is no way to
re-enter the SBUG program.
Breakpoints are not necessary when using the <S>ingle step or <V>alue option to
step through your program.

L -- LIST DEVICE TOGGLE

* To Toggle the List device between the Hard copy device entered, and the user screen display.

ENTER:
```
      L
```

The current disposition is displayed on the screen. To change conditions, simply re-enter this command.

S -- SINGLE STEP

* To Step throuh machine instructions one at a time.

ENTER:
```
      S
```

The instruction located at the user's PC is executed and displayed on the screen next to it's memory address. The effective jump address is also shown for all JUMP instructions.

NOTE:
Care must be taken when stepping through VDP RAM accessing. You should avoid this by placing a breakpoint following the VDP access instructions. You should also avoid stepping through GROM code.

T -- TRADE SCREEN (BIT MAP MODE ONLY)

* To toggle or Trade the user screen for the SUPER-BUGGER Screen.

ENTER:
```
      T
```

The SBUG screen will be swapped with the user screen ONLY if BIT MAP mode was selected upon entry to SBUG. Each time the <T> is pressed, the screen will toggle from user screen to SBUG and reverse.

V -- VALUE EQUAL

* To execute in slow speed untill the value at address entered is equal to the value entered

ENTER:
```
      V<Address><space> or <,><Value>
```

The User program will be executed in interpretive mode (slow-speed) until the value contained at the address specified is equal to the value entered on the command line. When the value is equal, the SBUG screen is restored and the program will be halted with the message "NO BREAKPOINTS SET".

IMPORTANT

DO NOT EXECUTE TI OPERATING SYSTEM CODE UNDER THIS OPTION.

If User program makes any branches into SYSTEM utilities such as SCAN or GPLLNK routines, then results are unpredictable and your computer may go into a catatonic state requiring you to power OFF then back on to recover.

This option is useful only for executing code which will not leave your program area.

THIS PRODUCT WAS DEVELOPED BY NAVARONE INDUSTRIES. IF YOU ARE INTERESTED IN ANY OF THEIR OTHER HARDWARE OR SOFTWARE DEVELOPMENT TOOLS, WRITE TO:

NAVARONE INDUSTRIES, INC.
510 LAWRENCE EXPRESSWAY, #800
SUNNYVALE, CA 94086

OR CALL (408) 866-8579.

COMMANDS

A -- DISS-ASSEMBLE MACHINE CODE

* To Diss-assemble code at current User PC Location

ENTER:
```
      A<return>
      [<number>]{<return> or <space>}
```

Diss-assembles the machine instruction/s beginning
at the user PC Address. If a single paramenter is entered
it is treated as the Number of instructions to display or
print.

* To Diss-assemble code between locations specified

ENTER:
```
      A[<Start Address>{<space> or <,>}<stop address>]
```

Diss-assembles the machine instruction/s beginning with the
start address entered and continuing to the stop address entered.

NOTE:
Output from this command will be directed the HARD COPY DEVICE
if the L(ist) has been turned ON. Otherwise the listing will be
displayed on the users screen.

D -- DUMP MEMORY

* To Dump code at current User PC Location

ENTER:
```
      D<return>
      D[<number>]{<return> or <space>}
```

Dumps the memory Data in HEX and ASCII beginning
at the user PC Address. If a single paramenter is entered
it is treated as the Number of locations to display or
print.

* To Dump code between locations specified

ENTER:
```
      D[<Start Address>{<space> or <,>}<stop address>]
```

Dumps the memory Data in HEX and ASCIIs beginning with the
start address entered and continuing to the stop address entered.

Note: Output from this command will be directed the HARD COPY DEVICE
if the L(ist) has been turned ON. Otherwise the listing will be
displayed on the users screen.