Debug users manual

1.0 Introduction

The purpose of this manual is to acquaint the user with the new atom, DEBUG, implemented by Computer Concepts Corporation. This atom allows the user to look into the internal architecure of the machine, and view or change various paramaters.

The first section describes that actual atom, while section II describes the program DEBUG, which shows examples of how to use the new atom.

DEBUG is an extremely powerful atom, and can do severe damage to your OS if one does not know how to use it.

NEVER use this atom on a live system. One mistake could cause a system crash that may cost loss of data at the least.

1.1 Atom DEBUG

The atom DEBUG, can lay on either side of the equal sign, and can be executed in either local or program mode.

DEBUG(1,0)= (DATA)

This version of DEBUG allows the setting of the memory bank assignment that is used when fetching Data memory locations. DATA may be a hex value or an ALPHANUMERIC variable. Only the first byte is used.

```
EXAMPLE:  DEBUG(1,0)=HEX(40)        Sets second Bank
          DEBUG(1,0)=A$             STR(A$,1,1)=Bank
```

DEBUG(2,<address>)= (DATA)

DEBUG 2 sets a Data Memory location pointed to by address to the value of the first byte of DATA.

```
Example:     DEBUG(2,256)=HEX(55)   Sets location 0100 to 55
             DEBUG(2,V)=A$
```

DEBUG(4, address)= DATA

DEBUG 4 allows us to set Control memory at the specified address to the value of the next three bytes of data. Note that the user must have set parity correctly.

Example: DEBUG(4,VAL(HEX(5C03),2))=HEX(DC0050)
DEBUG(4,A)=STR(Z$(),240,3)

These are all the left side functions. On the right side, the following functions exist.

A$=DEBUG(1,0)

Reads the currently set default bank select bits to A$

A$=DEBUG(2,<address>)

Reads one byte of Data memory from the currently selected bank , to A$

A$=DEBUG(3, offset , <partition, number of bytes >)

Reads in n number of bytes from the selected partition, starting at the base address of that partition offset by the value of the offset.

Example: A$=DEBUG(3,0,<2,400 >) Reads in 400 bytes starting at offset 0 of partition 2.

A$=DEBUG(4,<Address>)

Reads in three bytes, one control word, from Control memory at the location pointed to by Address.

A$=DEBUG(8, Address , <1,number>)

Reads in n number of bytes (256 bytes default) to A$

Example: A$=DEBUG(8,VAL(HEX(3000),2),<1,1000>)

Reads in 1000 bytes from location 3000 hex. Note that you must put a one (1) in the partition number so the DEBUG command thinks it has a valid partition reference, though partition references have nothing to do with this form of DEBUG.

2.0 Description of DEBUG program

The DEBUG program was designed as a tool to allow us to view the internal workings of the machine. When loaded from disk, the following is displayed:

```
Master menu    Basic DEBUG    Revision 4.1          #############
                                                    ## COMPUTER ##
                                                    ## CONCEPTS ##
                                                    #############
             Partitions Sysgened =  5

'00 - Print Master Menu             '08 - Wake Partition Up
'01 - Dump Data Memory              '09 - Force load Program to Partition
'02 - Inspect Control Memory        '10 - View Partition Registers
'03 - Set Mask for Searches         '11 - Map Spare CM locations
'04 - Search Control Memory         '12 - Dump Partition Data - Offset
'05 - Set Memory Bank bits          '13 - Map Spare file locations
'06 - View Last File OPENED
'07 - Put Partition to Sleep

             '31 - Load ASM routines

STOP 140
```

2.1 Mode 1 - Dump Data Memory

Mode 1 allows us to dump 256 bytes of Data memory at the requested address to the CRT. When invoked, the following is displayed:

Enter Address:

Enter any valid hex address. The system will load and display the 256 bytes of data. An example of the display follows:

```
Internal Data Memory Dump    SL = 40  SF'1

0900  02 02 56 00 00 00 00 00 08 00 08 00 30 30 30 38   ..V.........0008
0910  01 60 00 00 00 00 00 00 00 00 00 01 00 00 00 00   .`..............
0920  01 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00   ................
0930  01 22 48 77 65 90 40 44 00 80 00 00 00 00 00 00   ."Hwe.@D........
0940  00 00 40 00 00 80 00 00 00 00 00 00 00 00 00 00   ..@.............
0950  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0960  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0970  00 30 20 30 30 28 28 28 28 28 28 28 28 28 28 28   .0.00(((((((((((
0980  08 08 08 08 08 00 00 00 00 00 00 00 00 00 00 00   ................
0990  0C 7C 20 20 20 00 00 00 00 00 00 00 00 00 00 00   ................
09A0  C0 FF 0C 00 00 00 02 01 03 10 13 10 00 FF FF FF   ................
09B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
09C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
09D0  00 00 01 04 00 00 02 99 00 00 00 00 00 00 00 00   ................
09E0  00 00 40 80 C0 00 00 00 00 00 00 00 00 00 00 00   ..@.............
09F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

The user has control of the SL register, Bank select via mode 5. The 256 bytes displayed use the defaulted bank select.

2.2 *Inspection of Control Memory*

Mode 2 allows us to inspect, but not change, locations in control memory. Simply enter the address to be viewed, and the system will display the contents. If a return is pressed, the next sequential address is brought up.

```
Enter Starting Address ? 1000
1000 81901F

Enter Starting Address ?
1001 711F0A

Enter Starting Address ?
1002 56CE0C

Enter Starting Address ?
1003 540D10
```

2.3 *Set Mask for Searches*

The user may alter the search mask used during Mode 4 through this mode. This mask is used to "and" against the read control memory and the searched for data. Normally, the mask is set to $7FFFFF to allow us to match words without regard to parity.

2.4 Search Control Memory

This mode allows us to search Control memory within specified areas for the occurance of a user supplied word. This word is "anded" with the user set mask, and compared against Control memory, which is also "anded" with the mask. Matches are printed as addresses on the CRT.

```
Enter Starting Address ? 0000
Enter Ending Address   : ? 4FFF

Enter data to search for : ? 8B800F

Current Mask = 7FFFFF

0173 02C1 02EA 034B 0450 0460 0465 0468 048E 04C2 04CB 0577 0888 08A9 08AD 08C3
09F8 0BD9 0D55 0D79 0DD2 0DE4 0DEB 0EB6 1025 107D 1099 10AA 1108 1116 1140 115B
11B3 1210 1218 1325 1360 1413 1429 14DD 16AB 16E7 1756 175E 1878 1B81 1C99 1C9D
1CB9 1CE0 1D25 1D2A 1D2E 1D59 1D6F 1E73 2009 20DA 20E4 210A 23D5 23DD 2542 25F6
2645 2775 279D 27A2 28F7 2DE7 2F83 2F98 3023 30C2 30DA 3249 324F 3255 32DE 3713
3820 393F 394D 3A71 3B7E 3ED1 3EE5 3EEB 3EF4 3EFD 3F06 3F09 3F2E 3F55 3FDB 40F1
4257 425D 4264 4403 4575 4585 45CD 4A33 4A4F 4A85 4A88 4A94 4A97 4AA9 4AB8 4B6F
4B74 4B87 4B89 4C46 4DEC 4E72 4FD4 4FEA 4FF0
 121 matches have been found

Enter Starting Address ?
```

2.5 Set Memory Bank bits

Mode 1 requires that we had previously set the SL bits for memory bank selection. This SF key allows us to alter which memory bank we wish to view.

2.6 *View last file opened*

An interesting insite to the machine. One can view all currently enabled partitions and view which Data/Program file was last looked up or loaded by the various partitions in real time. Since this is a continous update, the user must press halt to exit.

Note that paramaters are available to allow the user to display the time of the change. With this feature, one could formulate a program which analyzes the disk usage of various partitions!

```
Number of partitions enabled is  5

Part#  File OPENed    Time
------------------------------
 1  **    DEBUG
 2
 3        SYM.224
 4
 5
```

2.7 *View Partition Registers*

Breaks down into plain english the contents of various registers used by a partition. When called, simply type the partition number to view. DEBUG will update the contents of the screen at intervals determined by system usage. We cannot view too often what occurs within a partition slice, but we can see how things move through the registers. An interesting exercise is to set up a partition to run Wangs' diagnostics, and view all the activity that has occurred.

In the event of a partition crash, this mode may be helpful to find out exactly what that partition was trying to do at the time of error.

To view another partition, just press the number of the partition.

```
Base Address 0C00  Bank 0   00    Partition 1   Status 00  30 MXD 00 CRT 00 # 1

   Screen format courtesy of                              Waiting for CRT
   Southern Data - Bob Drew                               CRT not attached
                                          Enabled
              R0 R1 R2 R3 R4 R5 R6 R7
Registers     00 01 00 02 00 02 AA 10


Aux     00 3C56  01 0089  02 084C  03 0005  04 1100  05 1025  06 9030  07 2034
Reg     08 2036  09 0000  0A 110A  0B 0752  0C 0DB6  0D 0755  0E 0000  0F 10AA
        10 1047  11 1139  12 1032  13 785C  14 204E  15 0F52  16 782C  17 0020
        18 0001  19 FFCF  1A 0005  1B 6000  1C 0015  1D 4020  1E 1139  1F 0000

 HW Status SH 1A    SW Status SL 02        Status  0000  Break Status 00
        Carry 0      64K Bank    00
      CPB-IBS 1      Execute pass          Waiting on Ready from device  00
  SF KEY (1B9) 0                               Requested IO from device  00
 IO Ready/Busy 1
  Part Timeout 1
 Halt/Step Key 0                           Last File  DADIAG01
         PEDM 0
         DMPI 0
```

2.8 Map Spare locations in file

Whenever we want to modify Basic, we need to know where the spare memory locations are. In most cases, we can run mode 13 to do this.

The file is opened, and all locations containing 800000, which is the No-operation code, are accumulated. Five continous locations containing the NOP code will result in a display.

At the end of the file, a summary is printed containing the accumulated spares per map.

```
- Spare Location Map for 22 -

0C47 to 0C5B for 20  locations
12EC to 12FF for 19  locations
1DA0 to 1DFF for 95  locations
27F9 to 27FF for 6  locations
33F2 to 33FF for 13  locations
3D4B to 3D5F for 20  locations
3FF8 to 3FFD for 5  locations
47F2 to 47FF for 13  locations
55FC to 57FF for 515  locations
5904 to 5BFF for 763  locations
5CEF to 5FFD for 782  locations
```

Map of Spare CM locations

Page

	0000	0400	0800	0C00
0000	0	0	0	25
1000	21	0	0	97
2000	4	7	0	5
3000	14	1	2	27
4000	0	14	2	0
5000	14	549	764	783

Total Spare locations = 2329