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 architecture of the machine, and view or change various parameters.

The first section describes the 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$

Copyright © 1983 by Computer Concepts Corporation, Shawnee Mission, Ks
No part of this document may be reproduced without the expressed written permission of Computer Concepts Corporation
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

Partitions Sysgened = 5

'00 - Print Master Menu
'01 - Dump Data Memory
'02 - Inspect Control Memory
'03 - Set Mask for Searches
'04 - Search Control Memory
'05 - Set Memory Bank bits
'06 - View Last File OPENED
'07 - Put Partition to Sleep

'08 - Wake Partition Up
'09 - Force load Program to Partition
'10 - View Partition Registers
'11 - Map Spare CM locations
'12 - Dump Partition Data - Offset
'13 - Map Spare file locations

'31 - Load ASM routines

STOP 140

Copyright © 1983 by Computer Concepts Corporation, Shawnee Mission, Ks
No part of this document may be reproduced without the expressed written permission of Computer Concepts Corporation
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 SF1

0900 02 02 56 00 00 00 00 00 00 00 00 00 00 00 00 30 30 30 30 .U.........000B
0910 01 40 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 46 77 65 90 40 44 00 80 00 00 00 00 00 00 00 00 00 00 00 .Hex.20
0940 00 00 40 00 00 80 00 00 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 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 ........
0980 00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..........0.000
0990 00 7C 20 20 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..........'
09A0 00 FF FF FF 00 00 02 01 03 10 13 10 00 FF FF ..................
09B0 00 00 00 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 00 00 00 ..................
09D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
09E0 00 00 40 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..........0.000
09F0 00 09 09 09 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 01991F

Enter Starting Address ?
1001 71FFFA

Enter Starting Address ?
1002 54CE0C

Enter Starting Address ?
1003 540D18

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 occurrence 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? 04FF

- Enter data to search for? 8B800F
- Current Mask = 7FFFFF

/173 02C1 02EA 0348 0450 0460 0468 048E 04C2 04CB 0577 0688 0BA4 0B8A 0BC3 0FD8 0B89 0055 0579 00D2 00E4 00EB 05E6 1025 1070 1099 1044 1108 1116 1146 1158 1183 1210 1218 1325 1340 1413 1429 1490 1668 1667 1756 175E 1878 1891 1C98 1C99 1C9A 1C9D 1D92 1D96 1D9A 1D9C 1D9E 1DAA 1DAB 1DAD 1DAB 1DAD 1DAB 1DAD 1DAB 1DAD 1DAB 1DAD 1DAB 1DAD 1DAB 1DAD 1DAB 1DAD 1DAB 1DAD 1DAB 1DAD 1DAB 1DAD 1DAB

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 insight 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 continuous update, the user must press halt to exit.

Note that parameters 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!

<table>
<thead>
<tr>
<th>Part#</th>
<th>File OPENed</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DEBUG</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SYM.224</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
**DEBUG - A window into the OS**

<table>
<thead>
<tr>
<th>Base Address</th>
<th>Bank</th>
<th>Partition</th>
<th>Status</th>
<th>CRT</th>
<th>00</th>
<th>CRT 00</th>
<th>01</th>
<th>11</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen format courtesy of</td>
<td>Southern Data - Bob Drew</td>
<td>Waiting for CRT</td>
<td>CRT not attached</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R0 R1 R2 R3 R4 R5 R6 R7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aux</td>
<td>02 0356 01 0089 02 084C 03 0005 04 1100 05 1025 06 9830 07 2034</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reg</td>
<td>00 2036 09 0000 0A 110A 0B 0752 0C 0086 0D 0755 0E 0000 DF 100A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1D 1047 11 1139 12 1032 13 785C 14 204E 15 5F52 16 782C 17 0020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18 0001 19 FFCF 1A 0005 1B 8000 1C 0015 1D 4020 1E 1139 1F 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW Status SH IA</td>
<td>SW Status SL 02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carry 0</td>
<td>Execute pass</td>
<td>Status 0000</td>
<td>Break Status 00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPB-18S 1</td>
<td>64K Bank 00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SF KEY (1B9) 0</td>
<td></td>
<td></td>
<td></td>
<td>Requested IO from device 00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO Ready/Busy 1</td>
<td></td>
<td></td>
<td></td>
<td>Waiting on Ready from device 00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part Timeout 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Halt/Step Key 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEDM 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWSI 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 1983 by Computer Concepts Corporation, Shawnee Mission, KS
No part of this document may be reproduced without the expressed written permission of Computer Concepts Corporation

8
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 continuous 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.

<table>
<thead>
<tr>
<th>Map of Spare CM locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0000</td>
</tr>
<tr>
<td>0000</td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>2000</td>
</tr>
<tr>
<td>3000</td>
</tr>
<tr>
<td>4000</td>
</tr>
<tr>
<td>5000</td>
</tr>
</tbody>
</table>

Total Spare locations = 2329

Copyright © 1983 by Computer Concepts Corporation, Shawnee Mission, KS
No part of this document may be reproduced without the expressed written permission of Computer Concepts Corporation